

國立交通大學

資訊科學與工程研究所

碩士論文



一個同儕式網路環境中基於完善差異圖與集合
之高效率查詢機制

An Efficient Query Mechanism Based on Perfect Difference
Graph/Set in P2P Networks

研究生：王振維

指導教授：陳耀宗 教授

中華民國 101 年 7 月

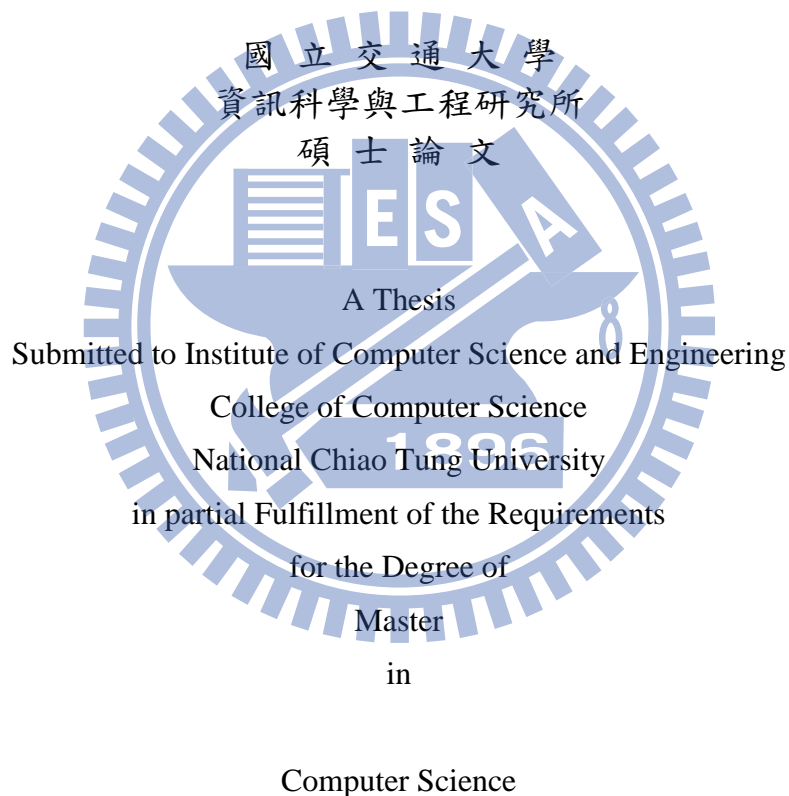
一個同儕式網路環境中基於完善差異圖與集合之高效率查詢機制
An Efficient Query Mechanism Based on Perfect Difference Graph/Set in
P2P Networks

研究生：王振維

Student：Chen-Wei Wang

指導教授：陳耀宗

Advisor：Yaw-Chung Chen



July 2012

Hsinchu, Taiwan, Republic of China

中華民國 101 年 7 月

一個同儕式網路環境中基於完善差異圖與集合之有效率查詢機制

學生：王振維

指導教授：陳耀宗 博士

國立交通大學資訊科學與工程研究所

摘要

在本篇論文中，我們提出一個嶄新並且有效率的查詢機制用於多層式非結構化同儕網路。此外我們的查詢機制不但具備可靠性，例如若某些資源存在系統中，我們能夠更有效率地送出廣播查詢訊息並且能夠成功地找到所有檔案資源；而且具備延展性，例如網路中查詢訊息的流量能夠被超級同儕所限制住。此外在我們多跳數多階層的架構下，可以藉由更強大的超級同儕來連結其他的完善差異網路。就我們調查過多數研究所知，幾乎沒有研究針對非結構化同儕網路來提出同時兼具可靠性和延展性的查詢機制。我們採用多跳指標復製造(Multi-hop Index Replication)和完善差異圖傳遞演算法，可使得每一個超級同儕不會收到額外的廣播查詢訊息；此外藉由查詢 Multi-hop Index，使得每一個超級同儕得以判斷檔案資源來自於何方，更進一步有效的減少網路流量。

我們設計一套模擬實驗，數據結果顯示我們所提出的方法不僅能有效的降低非結構化同儕網路中廣播查詢訊息量，平均傳送延遲，並且有效改善現有查詢成功率。實驗的結果也證明我們所提出的系統在超級同儕變動的網路中，不僅具備延展性而且在溝通訊息的負荷和成功率都可以表現出很好的查詢效能。

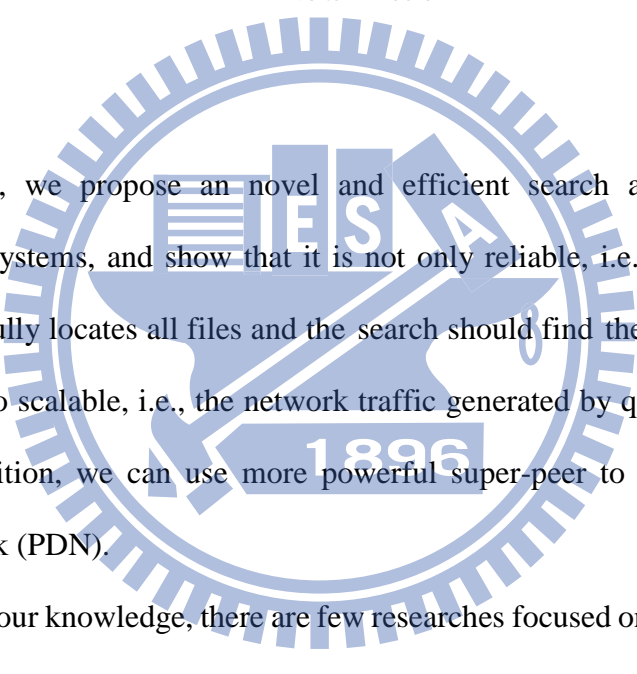
An Efficient Query Mechanism Based on Perfect Difference Graph/Set in P2P Networks

Student : Chen-Wei Wang

Advisor : Dr. Yaw-Chung Chen

Institute of Computer Science and Engineering
National Chiao Tung University

Abstract



In this thesis, we propose an novel and efficient search approach for multi-layer unstructured P2P systems, and show that it is not only reliable, i.e., if any content is in the system, it successfully locates all files and the search should find the content with reasonable guarantees, but also scalable, i.e., the network traffic generated by queries will be limited by super-peer. In addition, we can use more powerful super-peer to connect to other Perfect Difference Network (PDN).

To the best of our knowledge, there are few researches focused on the reliable and scalable search mechanism for unstructured P2P systems. The broadcasting performance of the P2P system is enhanced through the use of a Multi-hop Index Replication with Perfect Difference Graph (PDG) forwarding algorithm, which make certain that each super-peer receives just one copy of the broadcast message. Furthermore, by using the Multi-hop Index, super-peer has extra information to know what the files it queries is available or unavailable files, and demonstrate their effectiveness and efficiency through the simulation.

The experimental results show that our proposed scheme improves existing unstructured P2P systems in terms of a higher query success ratio, a smaller number of query flooding

messages and a lower average delay. In addition, we also point out that our proposed mechanism is efficient not only in scalability but also in reduction of communication overhead as well as the quality of query responses.



Acknowledgement

這篇論文的完成，首先要感謝我的指導老師陳耀宗教授，老師在研究上的指導讓我學習到做研究的本質與意義，除了學術上的指導以及研究上的訓練之外，老師在為人處事方面，也是我的學習典範。同時感謝留忠賢教授，詹家泰教授與李程輝教授在口試時給予意見，使我的論文更加地完整。

接著要感謝實驗室的阿華學長、同學。此外還有智程、家銘兩位學長在我研究上遇到瓶頸，以及論文撰寫的技巧方面，皆不吝地給予我許多的指點，使我的論文得以順利地完成。同時要感謝寢室的所有同學、學長，在生活以及研究上，互相的勉勵、加油與打氣，使我的研究所生活過得精采許多。也要感謝其他一起成長的夥伴們，陪我度過這飛快的兩年。

此外我要感謝士強學長以及維哲，還有我的朋友們昱志、明鑫、耀鋒、阿直、嘉佑、大頭、新翔、勇旗等，讓我的生活增添了許許多多珍貴的回憶。最後我還要感謝我的家人，謝謝他們一路上的支持，他們是我最大的動力及支柱。

Contents

摘要	i
Abstract.....	ii
Contents	v
Table List	vii
Figure List	viii
Chapter 1 Introduction.....	1
1.1 Overview	1
1.2 Motivation and Purpose.....	2
1.3 Thesis Organization	5
Chapter 2 Background	6
2.1 P2P Network.....	6
2.1.1 Introduction	6
2.1.2 P2P File Sharing Applications	7
2.2 The P2P Lookup Problems	12
Chapter 3 Proposed Approaches.....	14
3.1 Super-peer Overlay Networks and Forwarding Protocols.....	14
3.1.1 Perfect Difference Graphs	15
3.1.2 Broadcasting Over a Super-peer Overlay Network	18
3.1.3 Multi-hop Index Replication	19
3.2 System Construction and Architecture	23
3.2.1 System Construction.....	23
3.2.2 New Super-peer Join	26
3.2.3 Super-peer Leave	27
3.3 Numerical Analysis	29
Chapter 4 Simulation and Numerical Results	30
4.1 Simulation Environment and Simulation Setup	30
4.1.1 Simulation Environment.....	31
4.1.2 Simulation Setup	32
4.2 Numerical Results	33
4.2.1 Performance of the Average Traffic.....	33

4.2.2 Performance of the Network Traffic	36
4.2.3 Performance of the Response Time.....	37
Chapter 5 Conclusion and Future Works.....	39
Reference	41



Table List

Table 3.1 Comparison of vertex degree and graph diameter	15
Table 3.2 Relation between Super-peer Numbers N , Order δ and PDSs.....	16
Table 3.3 An example of super-peer table	24
Table 4.1 Experimental environment.....	31
Table 4.2 Peer Bandwidth Distribution	32
Table 4.3 Average value comparison	36



Figure List

Figure 1.1 P2P Architecture overview	1
Figure 1.2 Bootstrapping a new peer	4
Figure 2.1 Example 2-d coordinate overlay with 5 nodes	7
Figure 2.2 Partition of the CAN space as 5 nodes join succession	8
Figure 2.3 Identifier circle (ring) consisting of ten nodes storing five keys	9
Figure 2.4 A P2P system of nodes without central infrastructure.....	10
Figure 2.5 The peer obtains list of peers from tracker server	11
Figure 3.1 PDG with 7 vertices based on PDS {1, 3}	17
Figure 3.2 An example of PDG-based forwarding algorithm	18
Figure 3.3 An example of AVL Tree-Based Index	20
Figure 3.4 Multi-layer architecture for storage system	21
Figure 3.5 Network configuration of a new peer joining the super-peer	25
Figure 4.1 Search messages incurred in mesh-based and PDG overlay networks	33
Figure 4.2 Query successful ratio in mesh-based networks	34
Figure 4.3 Query successful ratio in pure PDG networks	34
Figure 4.4 Query successful ratio in AVL PDG networks	35
Figure 4.5 Comparison of Query successful ratio in mesh-based & PDG networks	35
Figure 4.6 Network Traffic in mesh-based and PDG overlay networks..	37
Figure 4.7 Average response time incurred in mesh-based and PDG overlay networks.....	38

Chapter 1 Introduction

1.1 Overview

The subject of this thesis is regarding the performance of content search in unstructured peer-to-peer (P2P) systems. Such systems have been used for a variety of applications, including content distribution, file-sharing and video streaming. These applications have been very popular; today's Internet traffic is mostly contributed by these services with number of users typically in millions.

A P2P network is a logical overlay network on top of a physical network. Each peer corresponds to a node in the P2P network and resides in a node in the physical network. All peers are of equal roles. Figure 1.1 shows the overview of the P2P architecture. The physical path is determined by a routing protocol and composed of one or more physical links. Logical links can be added to the P2P network arbitrarily as long as a corresponding physical path can be found, that is, the physical network is connected.

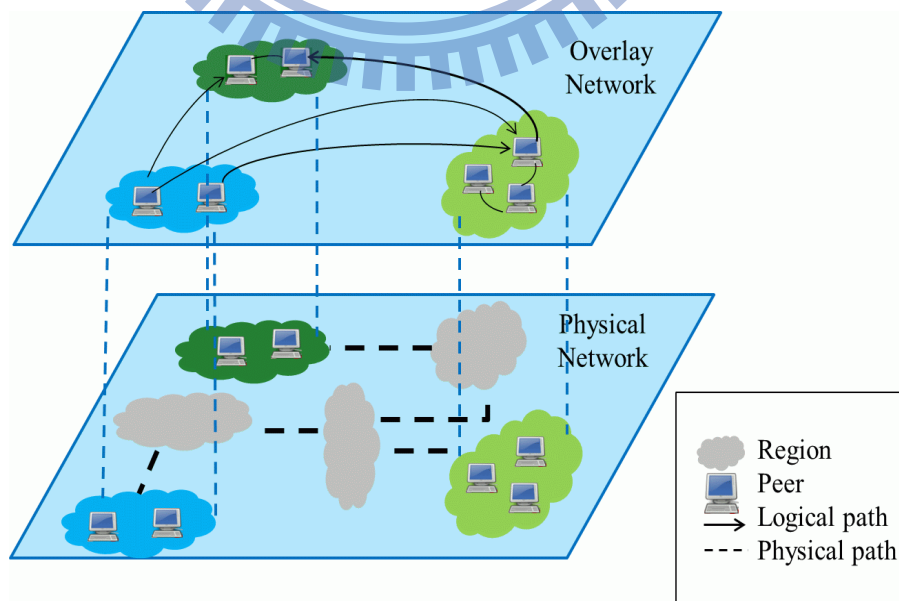


Figure 1.1 A P2P architecture overview.

1.2 Motivation and Purpose

Structured P2P overlay networks can provide efficient and accurate query service but need a lot of effort to maintain the DHT, it leads to frequent peer joining and leaving, also known as churn. Churn is a common phenomenon in P2P overlay networks. Measurement studies of deployed P2P overlay networks show a high rate of churn [21], [22].

Unstructured P2P overlay networks organize peers into an arbitrary network topology, and use flooding or random walks to look up data items. Each peer receiving the flooding packets or random walk packets checks its own database for the data item queried. This approach does not impose any constraint on the network topology. It can perform complex data lookup and support peer heterogeneity. Unstructured P2P networks are resilient to churn. However, queries in unstructured systems can generate a lot of traffic load, thus making such systems unscalable.

Although random walk certainly reduces the amount of traffic, it incurred by queries for files that are available in the network, i.e., for files that are stored by at least one peer [6]. Random walk search protocols still result in a large amount of traffic load when the requested resource does not exist in the overlay.

Unfortunately, searches for files that are not in the system are very common in practice [7]. W. Acosta and S. Chandra observed that roughly half of the queries (between 44% and 55.6%) cannot be matched to any file in the system. One solution used in practice to reduce the amount of query traffic generated by queries for unavailable files is to set the time-to-live (TTL) of query packets to a small value.

However, searching with a small TTL value will only search a small part of the peers in the system and queries are likely to be unsuccessful, even if the requested file is actually available in the network. Measurement studies on actual unstructured P2P networks observed that the query success rate is very small, typically close to 10% [24].

Therefore, neither structured P2P overlay networks nor unstructured P2P overlay networks can provide efficient, flexible, and robust service at the same time. The motivation of this thesis is to combine the two types of P2P networks and provide a hybrid approach which can support scalability and reliability at the same time. To achieve this goal, the approach should inherit the advantages of both types in such a way that their disadvantages are minimized.

In this thesis, we propose Multi-hop Index Replication that can improve search quality for rare objects while minimizing the overhead incurred by participating peers. Not only does index replication incur much lower overhead compared with data replication, previous work has shown it to be effective at improving the scalability of unstructured networks [6], [12], [22]. Our Multi-hop Index Replication with PDG forwarding algorithm eliminates the impacts of redundant query flooding messages and reduces the amount of network traffic generated by searches for unavailable files between the super-peer and ordinary peer layer of the P2P system.

In the proposed hybrid P2P system, bootstrap peer has to maintain a super-peer table. Any peer joining the P2P network and wishing to become a super-peer must first issue a request to the bootstrap peer (BSP). After examining each requesting peer's bandwidth conditions, the BSP may select the peer as a super-peer, and send the peer the corresponding connection information or register the peer as a redundant super-peer, and provide the peer with a list of super-peers which can be used to connect to the system or just only provide a list of super-peers. The joining procedure involved in overlay network is shown in Figure 1.2.

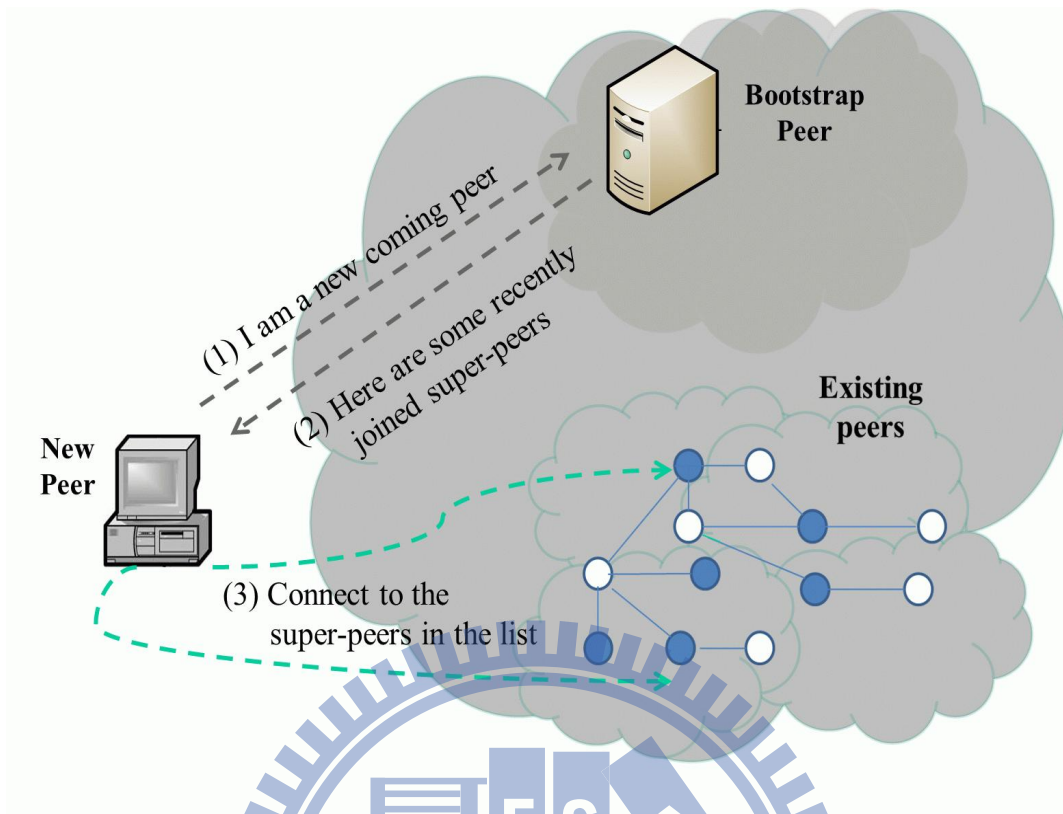


Figure 1.2 Bootstrapping a new peer.

When the overlay topology has been established, a pure PDG [8-9] forwarding algorithm is used to transport the query messages from the originating super-peer to the other super-peers in the overlay in such a way that each super-peer receives just one message. In addition, each super-peer has to maintain an AVL tree-based index. AVL tree-based index is constructed with a randomly generated key which is the resource name published by the ordinary peers through the SHA1-liked algorithm. Besides, its average case complexity of search, insert, and delete operations is $O(\log n)$, where n is the number of sharing files in the overlay network.

In addition, our system also can support to be a storage system. By multi-layer and multi-hop architecture, we can allocate more powerful super-peer (MSP) with large storage space, computational capability and higher bandwidth to manage a whole PDN cluster, and MSP still form a Perfect Difference Network (PDN) in order to further enhance reliability and scalability.

1.3 Thesis Organization

The main contributions of this thesis can be summarized as follows:

- We propose a novel and efficient search mechanism for multi-layer unstructured P2P systems, and show that it is not only reliable, i.e., if any content is in the system, it successfully locates all files and the search should success with reasonable guarantees, but also scalable, i.e., the network traffic generated by queries will be limited by super-peer. In addition, we can use more powerful super-peer to connect to other Perfect Difference Network (PDN).
- Multi-hop Index with PDG flooding algorithm eliminates the impacts of redundant query flooding messages and reduces the amount of network traffic generated by searches for unavailable files between the super-peer and ordinary peer layer of the P2P system.
- Evaluate the proposed scheme through extensive simulations. Performance evaluation demonstrates that the proposed flooding algorithm outperforms existing unstructured P2P overlay network in terms of a higher query success ratio, a lower number of lookup query flooding messages and a lower average delay.

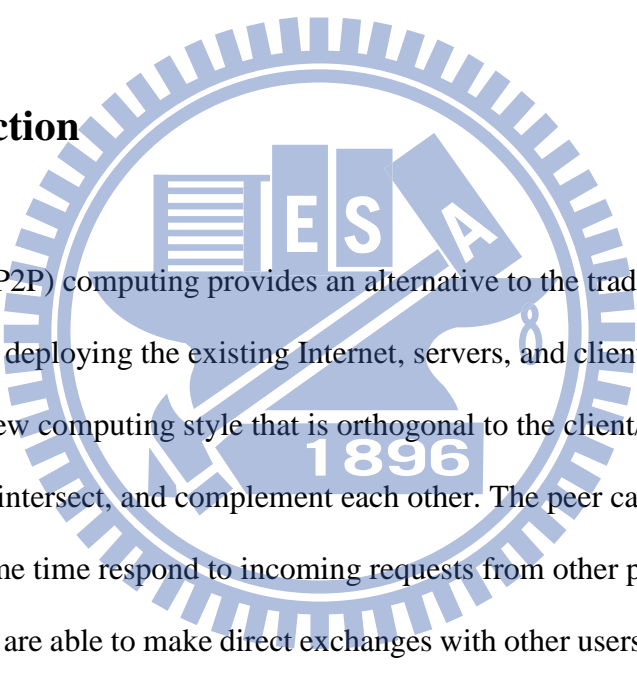
We organized the remaining thesis as follows: In Chapter 2 we present the background of the literature relating to structured P2P and unstructured P2P systems. In Chapter 3, we discuss the proposed hybrid P2P system and describe our methods in representing a numeric range and the analytic models. In Chapter 4, we evaluate the effectiveness of our methods and discuss the performance of the system. Finally we give the concluding remark and future work in Chapter 5.

Chapter 2 Background

In this chapter, we briefly describe peer-to-peer network features focusing on those content sharing applications that are relevant to this thesis. Next we shortly describe and review some literatures regarding the different kinds of lookup problems in P2P overlay networks.

2.1 P2P Network

2.1.1 Introduction



Peer-to-peer (P2P) computing provides an alternative to the traditional client/server architecture. While deploying the existing Internet, servers, and clients infrastructure, P2P supports a whole new computing style that is orthogonal to the client/server model. The two characters coexist, intersect, and complement each other. The peer can send requests to other peers and at the same time respond to incoming requests from other peers on the overlay network. The peers are able to make direct exchanges with other users liberates P2P users from the traditional dependence on the central servers. Users have a higher degree of autonomy and control over the services they utilize.

One of the greatest benefits of P2P computing is community. P2P makes it possible for users to organize themselves into ad hoc groups that can efficiently and securely fulfill requests, share resources, collaborate, and communicate. As P2P computing evolves, we can anticipate the emergence of a wide variety of these online communities [18].

2.1.2 P2P File Sharing Applications

Many P2P networks have been proposed for different applications in the literatures, for examples, [1], [2], [3], [4], [5], [6], [11], [12], [13], [14], [15].

In this thesis, we focus on P2P networks for efficient distributed data (file) sharing among peers.

The Content Addressable Network (CAN) [16] was proposed to provide a scalable, fault tolerant, and self-organizing indexing mechanism for file sharing over a large network. The entire space is partitioned to distinct zones such that each peer is in charge of one zone. Every peer maintains a routing table which holds the IP address of its neighbors in the coordinate space.

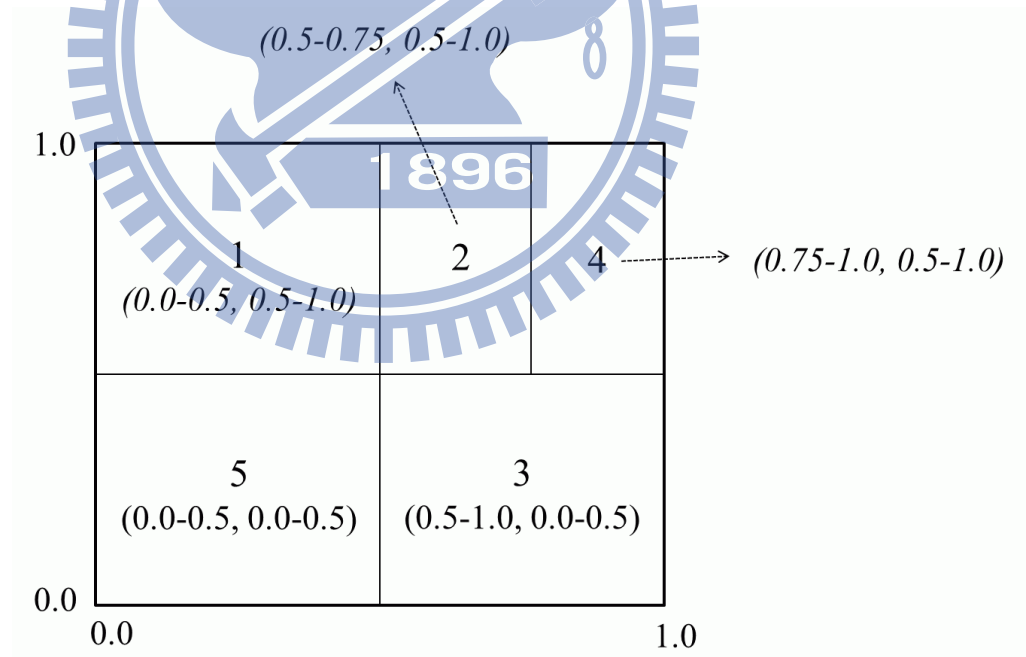


Figure 2.1 Example 2-d coordinate overlay with 5 nodes.

In Figure 2.1 shows a 2-dimensional $[0, 1] \times [0, 1]$ coordinate space with 5 nodes. Nodes in the CAN self-organize into an overlay network that represents this virtual coordinate space.

The data is stored in and retrieved from the peer that owns the zone covering the data. CAN takes advantage of the ordering of the Cartesian coordinate space in the routing protocol. When a new peer joins the system, currently existing zone will be split into two zones, one of which is assigned to the new coming one and all the related peers need to update their neighbor lists. Figure 2.2 shows that the evolution of a 2-d CAN space as 5 nodes join in succession. The first node to join owns the entire CAN space, i.e., its zone is the complete virtual space. When the second node joins, the space is split in two and each node gets one half. The third node to arrive picks one zone and splits it in half, and this process repeats as new nodes arrive. When a peer leaves the system, a neighboring peer will take over the zone by running a take-over algorithm, and all the related peers need to update their neighbor lists again.

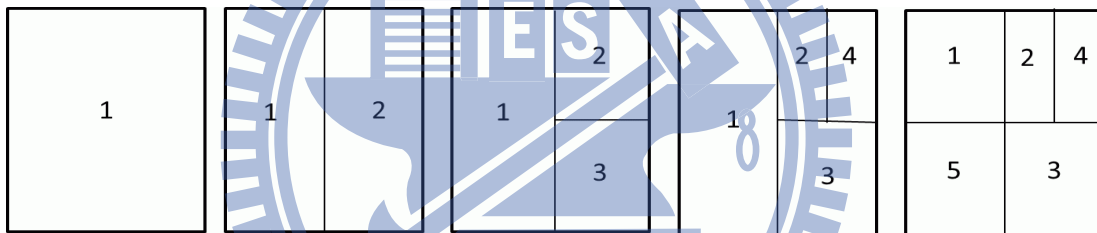


Figure 2.2 Partition of the CAN space as 5 nodes join in succession.

Chord [11] organizes the node keys into a circle which is called a chord ring, where each peer is assigned an ID. Peers are inserted into the ring in the order of their IDs. Each peer has two neighbors: successor and predecessor. When a peer joins the system, it first finds the position to insert the new peer. Then, the successor pointers of both the new peer and an existing peer must be changed. The correctness of Chord relies on the fact that each peer is aware of its successor.

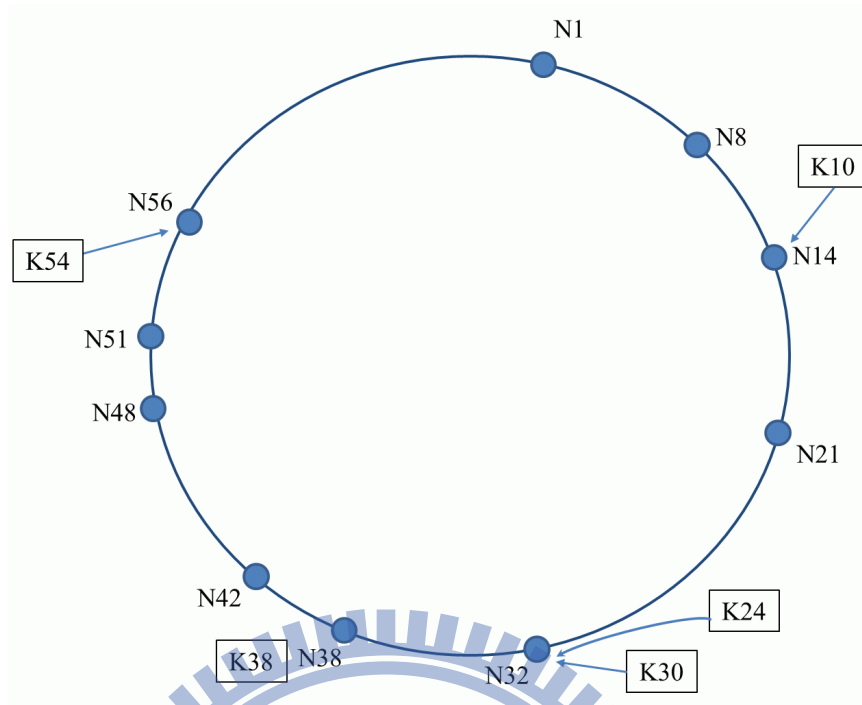


Figure 2.3 Identifier circle (ring) consisting of ten nodes in which five keys are stored.

To guarantee this, each peer maintains a successor list of size r which contains the peer's first r successors. Each data item also has an ID and is stored in a peer such that the ID of the data item is between the ID of the peer and its predecessor. In Figure 2.3 shows that packets are forwarded along the circle. In order to accelerate the search, each peer maintains a finger table, where each finger points to a peer with a certain distance from the current peer. Chord uses a "stabilization" protocol running in the background to update the successor pointers and finger tables. Compared to CAN, Chord is simpler as the key is hashed into a 1D space.

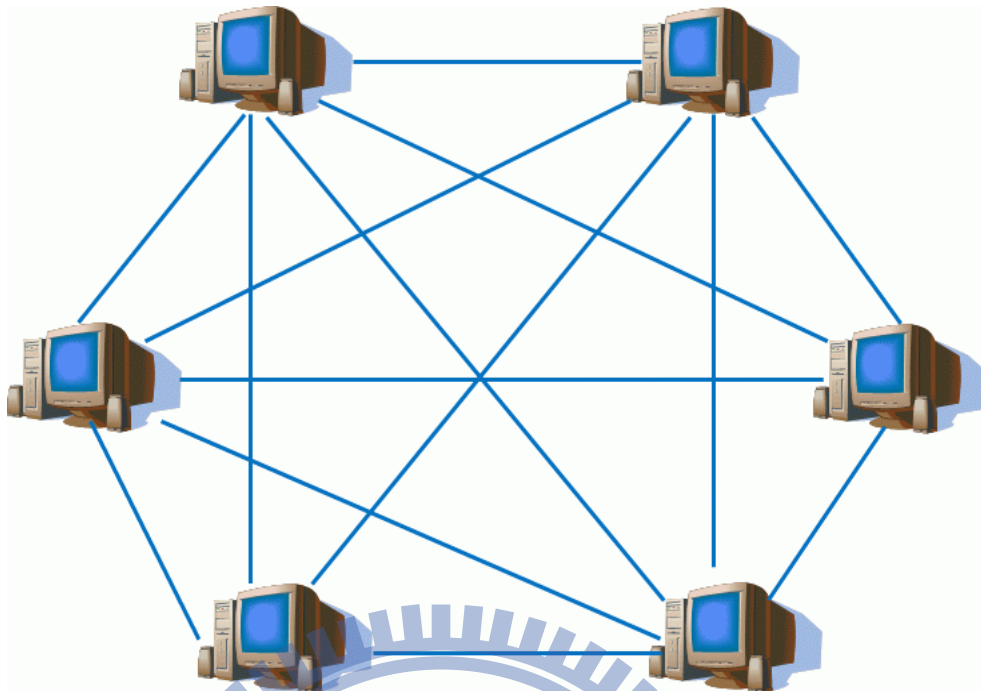


Figure 2.4 A P2P system infrastructure without a central server.

Gnutella [13] is a decentralized unstructured peer-to-peer network. The network is formed by peers joining the network following some loose rules, Figure 2.4 shows an example. All peers have same capability and responsibility. There is no constraint on the network topology. To look up a data item, a peer sends a flooding search request to all neighbors within some radius. Although Gnutella system has no requirement on the network topology and data placement, it is considerably resilient to peer joining and leaving the system frequently. However, flooding method is not scalable and consumes a lot of network bandwidth. Also, it is hard to find a rare data item as it has to flood the search request to all of the peers. There has been some research [19] on improving the efficiency while looking up a rare data item.

BitTorrent [14] is a centralized unstructured peer-to-peer network for file sharing. It uses a central server named tracker which keeps track of all peers who have the file. Figure 2.5 depicts the tracker that maintains information about all BitTorrent clients utilizing each torrent. Specifically, the tracker records the network position of each client either uploading or downloading the P2P file associated with a torrent.

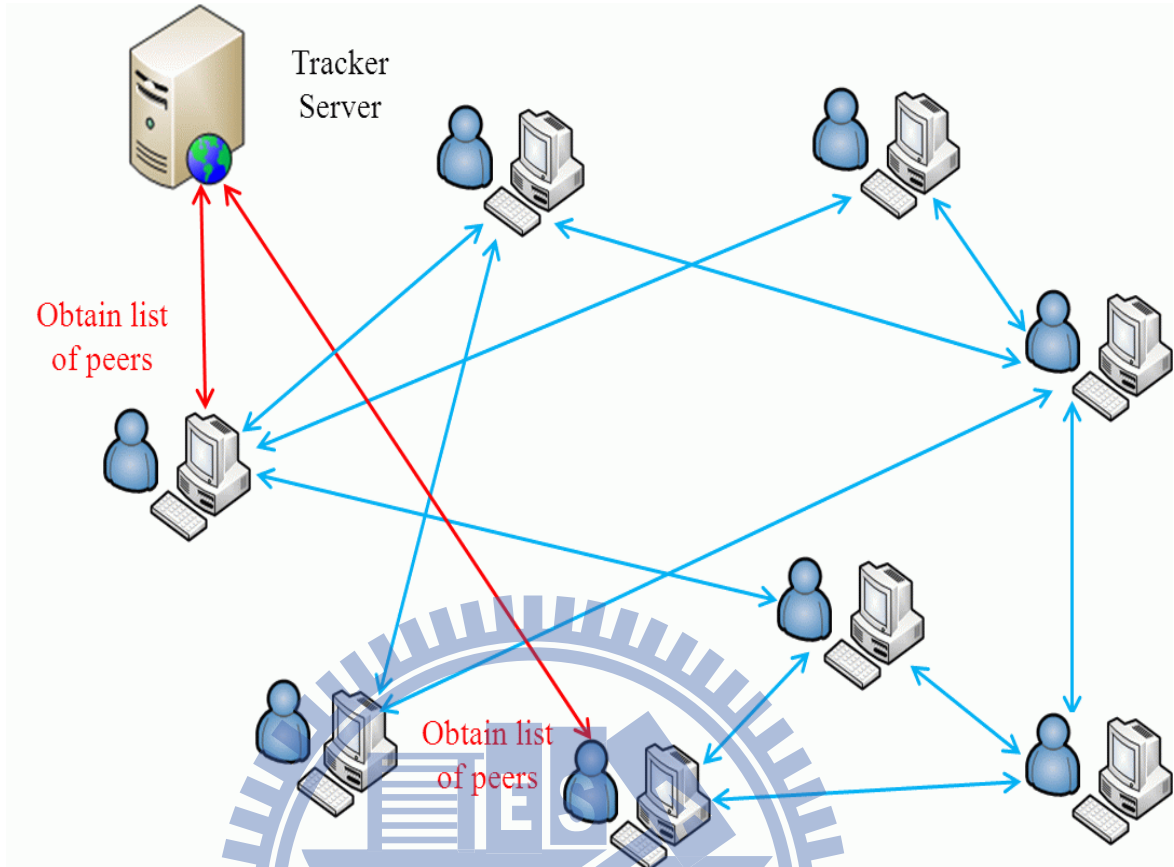


Figure 2.5 The peer obtains a list of peers from the tracker server.

Each file has a corresponding torrent file stored in the tracker which contains the information about the file, such as its length, name, and hashing information. When receiving a download request, the tracker sends back a random list of peers which are downloading the same file. Clients that have already begun downloading also communicate with the tracker periodically to negotiate with newer peers and provide statistics; however, after the initial reception of peer data, peer communication can continue without a tracker.

When a peer has received the complete file, it should stay in the system for other peers to download at least one copy of the file from it. Since BitTorrent uses a central server to store all the information about the file and the peers downloading the file, it suffers the problem called single point of failure which means that if the central server fails, the entire system is brought to a halt.

YAPPERS [15] combines both structured P2P networks and unstructured P2P networks to provide a scalable search service over an arbitrary topology. However, YAPPERS is designed for efficient partial search which only returns partial values of data. For a complete search, YAPPERS still needs to flood the query to all peers which are in the same color as the data. Compared to YAPPERS, the multi-layer unstructured P2P system proposed in this thesis can further increase the accuracy of the lookups in a more efficient way.

In [17], the main difference is that the structured overlay was used to support for unsuccessful flooding data search, however in this thesis, the structured overlay is responsible for connecting all the unstructured overlays and transmitting query requests between them.

2.2 The P2P Lookup Problem

P2P systems lack a centralized administrative entity that serves and controls the peer resources. Its decentralized storage with decentralized downloads in P2P systems makes the file transfer process inherently scalable; this makes it difficult to ensure high class of performance and availability. Users are free to reboot their machine or turn off the application, so a high degree of redundancy is required. This makes P2P systems have to ensure reliability of the data, such as web hosting, or other centralized entities.

Centralized approaches are typically cited as being vulnerable to cause a single point of failure and being hard to scale for supporting millions of users. An additional drawback of centralized approach is that, being easy to shut down, they are vulnerable to malicious and legal attacks as was evinced by the termination of Napster. These shortcoming and legality issues led to the adoption of decentralized solutions.

However, W. Acosta and S. Chandra observed more than 2.5 million queries generated by peers in Gnutella over a long period in April 2007. Queries for files that are not in the system

are very common in practice. W. Acosta and S. Chandra took several analysis of the Gnutella P2P system and observed queries issued by peers, as well as files available in the network.

They observed that roughly half of the query results (about 44% in Oct. 2006 and about 55.6% in Apr. 2007) could not be matched with any file in the system. In general, unstructured P2P systems exhibit a much lower query success rate, close to 10%. This low success rate can be attributed to many reasons. For example, time-to-live fields are usually set to a low value such that a large portion of searches for available files did not succeed.

Unfortunately, currently deployed systems have significant scaling problems; for example, in Gnutella, searches are flooded with a certain scope; flooding on every query is not scalable and, because the flooding has to be shortened, may fail to find the desired resource in P2P systems. FreeNet, another decentralized system, uses a random-walk-based search algorithm that may fail to find files even when they are actually in the system.

The motivation of this thesis is to combine the two types of P2P networks and provide a hybrid approach which can support scalability and reliability at the same time. To achieve this goal, the solution should inherit the advantages of both types in such a way that their disadvantages are minimized.

Chapter 3 Proposed Approaches

As mentioned in previous chapters, neither structured P2P networks nor unstructured P2P networks can fulfill the requirements of efficiency, scalability, and reliability of P2P service alone. The motivation of this thesis is to combine the two types of P2P networks and provide a hybrid approach which can offer better efficiency and scalability. In this chapter, we present a network model and describe our scheme in detail.

3.1 Super-Peer Overlay Networks and Forwarding Protocols

Since super-peers usually have a high bandwidth connection, they can adapt to a higher traffic demand. The super-peers overlay topology can be constructed as a graph, in which vertices represent individual super-peers while undirected edges stand for connections between super-peers. Table 3.1 gives a summary of the vertex connection degree and graph diameter of various well-known graph methods.

As shown in Table 3.1, each peer in perfect difference graph (PDG) has a degree $O(\sqrt{n})$, and thus the topology is much more flexible than the complete graph (i.e., $O(n)$). Furthermore, even though the connection degree of peers in the PDG is much lower than those in the complete graph, the search range of a PDG-based topology is similar to the complete graph-based topology.

Table 3.1 Comparison of vertex degree and graph diameter

Order of vertex degree	Graph diameter	Example network
$O(n)$	1	Complete Graph
$O(\sqrt{n})$	2	Perfect Difference Graph
$O\left(\frac{\log n}{\log \log n}\right)$	$\theta\left(\frac{\log n}{\log \log n}\right)$	Star, Pancake
$O(\log n)$	$\log n$	Binary Tree Hypercube
$O(1)$	$n/2$	Ring

In addition, Table 3.1 shows that the other graph topologies have both a lower peer connection degree and a greater diameter than the PDG approach. (The diameter represents the value of maximum number of hops in the path between the source and destination in the graph.) Thus, the PDG-based overlay topology is a better choice for the hybrid-structured P2P system presented in this thesis.

3.1.1 Perfect Difference Graphs

PDGs [8], according to the math definition of perfect difference sets (PDSs), provides the mathematical knowledge for achieving this optimum number of peers to construct the framework of perfect difference networks or PDNs. Now, we give the definition of perfect difference network.

Definition 1: Perfect Difference Network (PDN) — there are $n = \delta^2 + \delta + 1$ nodes, numbered 0 to $n-1$. Node i is connected to node $i \pm 1$ and $i \pm s_j \pmod{n}$, for $2 \leq j \leq \delta$, where s_j is an element of the PDS $\{s_0, s_1, \dots, s_\delta\}$ of order δ .

Table 3.2 Relation between Super-peer Numbers N, Order δ and PDSs

N	δ	PDS $\{s_1, s_2, \dots, s_\delta\}$
7	2	{1,3}
13	3	{1,3,9}
21	4	{1,4,14,16}
31	5	{1,3,8,12,18}
57	7	{1,3,13,32,36,54,63}
73	8	{1,3,7,15,31,36,54,63}
91	9	{1,3,9,27,49,56,61,77,81}
133	11	{1,3,12,20,34,38,81,88,94,104,109}
183	13	{1,3,16,23,28,42,76,82,86,119,137,154,175}
273	16	{1,3,7,15,63,90,116,127,136,181,194,204,233,238,255}

Table 3.2 illustrates the number of peers, the number of elements and the order in the first ten PDSs. Figure 3.1 presents a PDG overlay based on the PDS $\{1, 3\}$. Since there are two elements in the PDS, the graph has $(2^2 + 2 + 1 = 7)$ seven peers. For example, peer 0 has edges connecting to peers $(0 \pm 1) \pmod{7}$ and $(0 \pm 3) \pmod{7}$. In other words, peer 0 has edges connecting to peer 1, 3, 4 and 6.

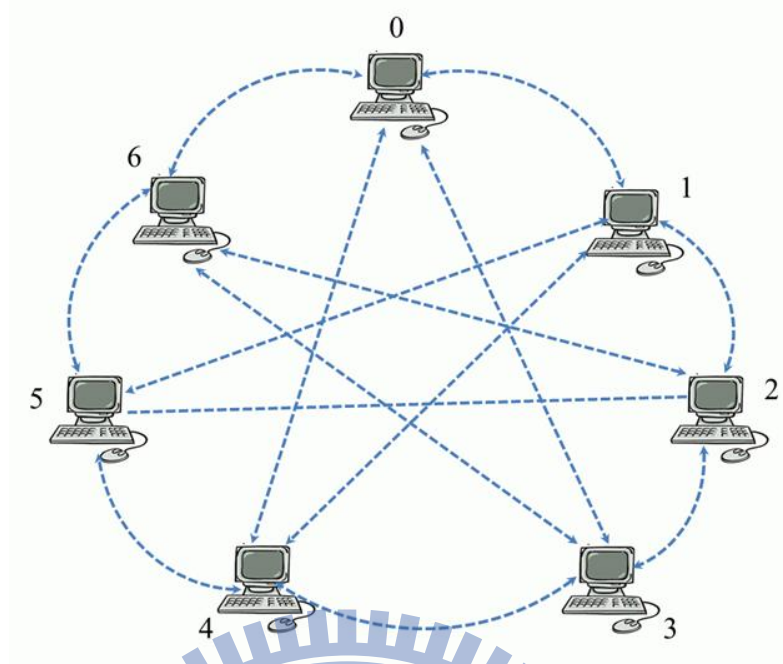


Figure 3.1 PDG with 7 vertices based on PDS $\{1, 3\}$.

For convenience to discuss the PDN topology, the following terms are adopted in the remainder of this thesis:

- Ring edge: the edge connecting consecutive peer i and $i \pm s_1(\text{mod } n)$, where $s_1 = 1$.
- Chord edge: the edge connecting non-consecutive peer i and $i \pm s_j(\text{mod } n)$, $2 \leq j \leq \delta$.
- Forward edges: for peer i , the forward edges include the ring edge connecting peer i and $i + s_1(\text{mod } n)$ and the chord edge connecting peer i and $i + s_j(\text{mod } n)$.
- Backward edges: for peer i , the backward edges include the ring edge connecting peer i and $i - s_1(\text{mod } n)$ and the chord edge connecting peer i and $i - s_j(\text{mod } n)$.

For example, in Figure 3.1, we present here a brief description of the forward edges of peer 0, which are the edges connecting peer 0 to peer 1 and 3, respectively, and the backward edges are the edges connecting peer 0 to peer 4 and 6, respectively.

3.1.2 Broadcasting over a Super-peer Overlay Network

This research deploys a PDG-based forwarding algorithm [9] in which the query requests are delivered to all the super-peers in the overlay network via the forward and backward edges of the perfect difference network. Each super-peer will send the search requests by using the forwarding algorithm of PDG and make certain that each super-peer receives only one copy of the search requests.

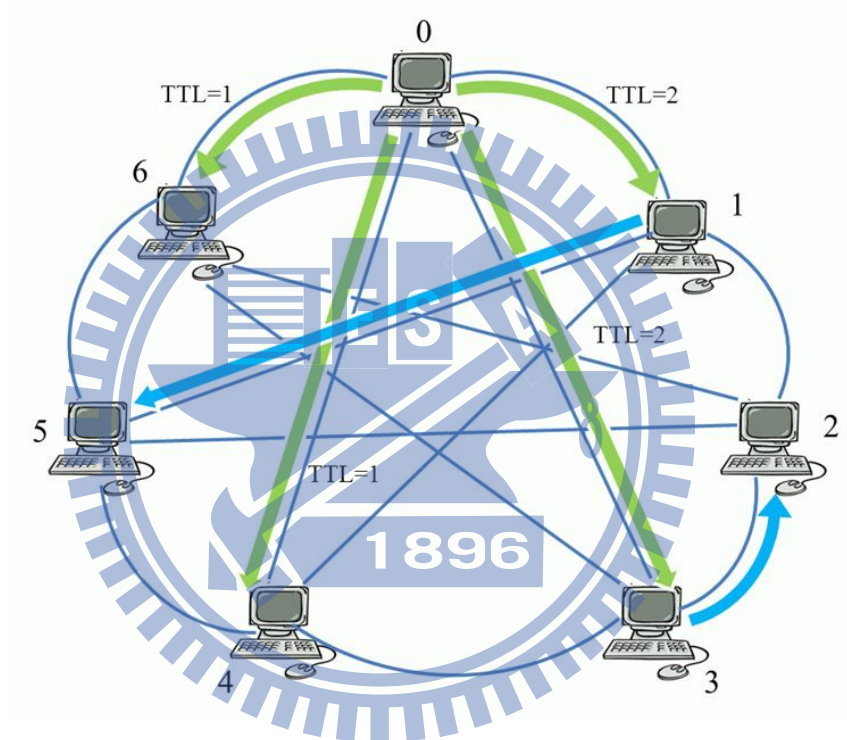


Figure 3.2 An example of PDG-based forwarding algorithm.

Assume that super-peer i wants to flood a query to all the other super-peer in the overlay network. The PDG forwarding algorithm follows two steps:

Step 1: Super-peer i sends a request message with $TTL=2$ to all of its forward partners and sends a request message with $TTL=1$ to all of its backward partners.

Step 2: If an intermediate super-peer receives the request message, it duplicates the message to all of its backward partners other than the partner from which it received the original message.

Figure 3.2 presents an illustration of the PDG-based forwarding algorithm for a super-peer overlay network forming a PDG with an order of $\delta = 2$. In this example, it is assumed that super-peer 0 wants to send a query to all other super-peers.

According to the two steps described above, super-peer 0 sends a query request with TTL=1 and 2 by its forward and backward edges to partners {1, 3} and {4, 6}, respectively. In the case of TTL=2, the TTL value is reduced to 1, and partners {1, 3} forward a copy of the query request to all their backward peers other than the edge on which they received the original message. In the case of TTL=1, since the TTL value is reduced to zero, partners 4 and 6 take no further action.

3.1.3 Multi-hop Index Replication

We explore the use of multi-hop index replication, which can significantly improve the cover region or effective search space of these overlays, while incurring low overhead compared to alternatives such as data replication. We explore the effectiveness of two-hop index replication, and that the super-peers have to manage the metadata which is sent by the ordinary peer. So super-peers have the ability to decide whether metadata queried by ordinary peers is available or unavailable to super-peers or the global index. We use an AVL tree to be the global index by converting the index into hash value through the SHA1-liked algorithm. Within the multi-hop index, we can efficiently stores very large names of files and maintain them as the database.

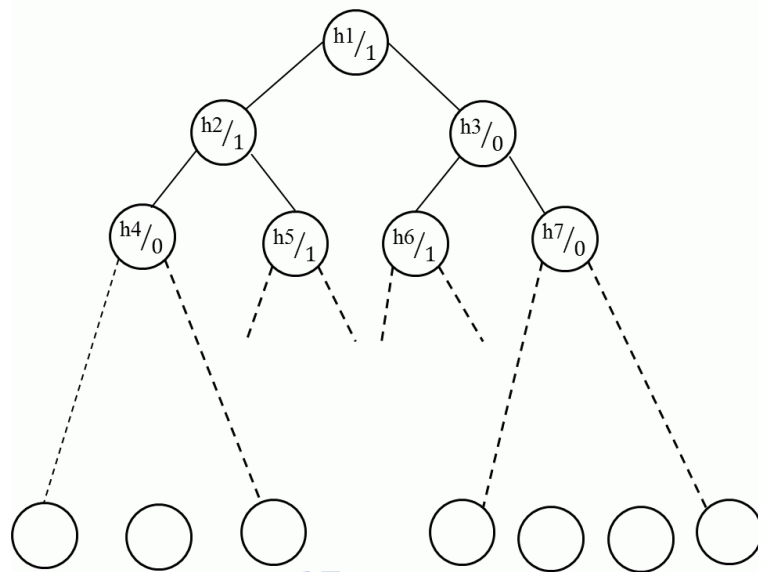


Figure 3.3 An example of AVL Tree-based Index.

We want to further reduce the lookup messages between super-peers. Each super-peer has to maintain the index replication which is constructed by AVL tree data structure. An AVL tree is a self-balancing binary search tree. In an AVL tree, the heights of the two child sub-trees of any node differ by at most one; therefore, it is also said to be height-balanced.

Its lookup, insertion, and deletion all take $O(\log n)$ time in both the average and worst cases, where n is the number of sharing files in the overlay network. Insertions and deletions may destroy the ideally height-balanced property requiring the tree to be reorganized by one or more tree rotations. Due to the characteristic that the nodes position in AVL tree can be changed dynamically in real-time, the AVL tree can support dynamic mechanism in data management.

We explore two-hop index replication strategy. In this strategy, each ordinary peer sends the name of sharing files to all of its one-hop super-peer. Each super-peer maintains the index replication using AVL tree data structure. The index is constructed with a randomly generated key which is the name of sharing files published by the ordinary peers through the SHA1-liked algorithm. These hash keys would be inserted to the AVL tree-based index according to the size of hash values. Furthermore, the other hop, each super-peer broadcast the available

resource names by using PDG algorithm. In order to further reduce the redundant broadcast messages, each super-peer use 1-bit to record the information of the sharing files where it comes from. If the bit were set, it means that the sharing files would come from ordinary peers it controlled. Otherwise, the sharing files are shared by other super-peers.

By this way, only when the bit is set, does a super-peer sends the query messages to its ordinary peers. Otherwise, the super-peer will forward the lookup messages to other super-peers directly by PDG-based algorithm without sending to its ordinary peers. For example, in Figure 3.3, we present here a brief description of the example of AVL tree-based index. According to the criteria of the query messages, each super-peer can search the AVL tree to decide whether to broadcast messages to its ordinary peers or not. If we can't find any records in the AVL tree-based index, it means that there is no resource published by peers. By the Multi-hop index replication, we can guarantee the reliability of the system and make it more efficient.

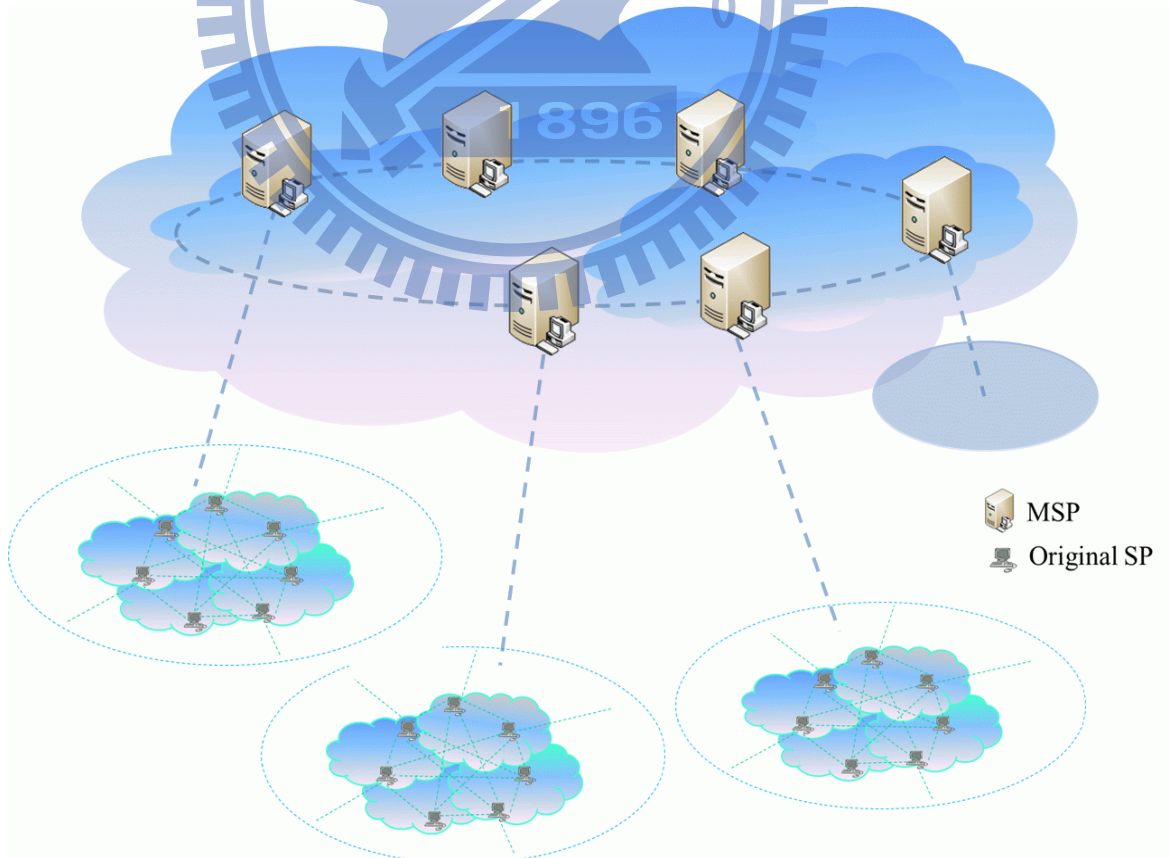


Figure 3.4 (a) Multi-layer architecture for storage system.

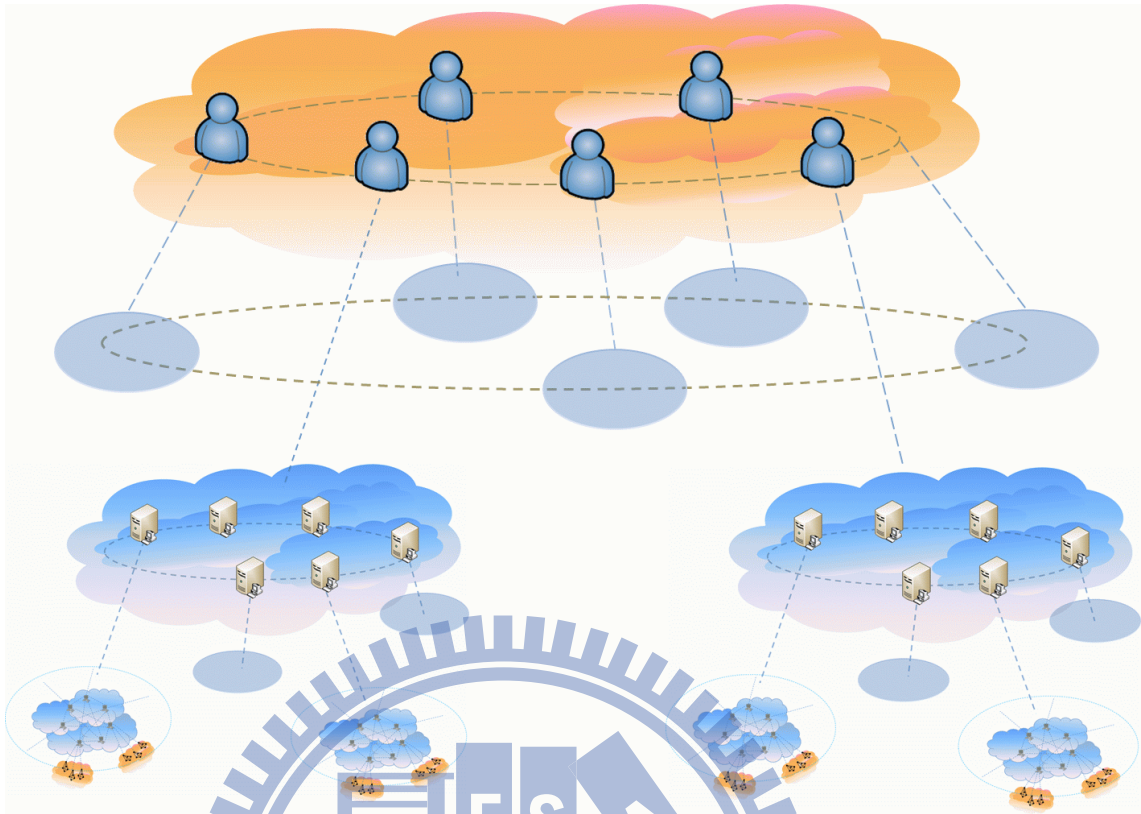


Figure 3.4 (b) Multi-layer architecture for storage system.

Our system also can support to be a storage system. By multi-layer and multi-hop architecture, we can allocate the more powerful super-peer (MSP) with large storage space, computational capability and higher bandwidth to manage a whole Perfect Difference Network (PDN) cluster, and MSP still form a PDN and maintain an AVL-tree index, as shown in Figure 3.4 (a) and (b). By using the MSP, when each peer goes down or want to stay offline it pushes all its latest data to the MSP. MSP support a large space to save the files which each peer wants to share with others. So that when the original super-peer can't find any resources or files in the PDN cluster, then the super-peer of query initiator sends the search message to the MSP. Furthermore, there are many MSPs which serve different PDN clusters still form the PDG architecture. By this approach, we can use PDG forwarding algorithm to broadcast to the other networks to achieve the goal of enhancing the reliability and scalability.

3.2 System Construction and Architecture

In this section, we present a multi-layer unstructured P2P system for data sharing as an example. The bootstrap peer (BSP) uses a super-peer table to maintain the super-peer overlay structure. For convenience, we discuss only one bootstrap peer attached to the overlay network. Actually, it makes no difference to our scheme when there is more than one bootstrap peers.

3.2.1 System Construction

In the proposed multi-layer unstructured P2P systems, there is at least one entry point for bootstrapping. It is able to use bootstrapping to connect to other peers and provides new peers entering the system with a list of addresses of super-peers seen recently in the overlay. The new ordinary peers then tried to establish overlay connections to the super-peers in this list. In our system, the ordinary peer can connect to two super-peers at the same time. By this way, when one of the super-peers leaves or crashes, the other one still has its records.

In the situation when a new peer wants to enter the overlay as an ordinary peer, it will send a request to the bootstrap peer. On request by a newly joining node, BSP sends the node a list containing the IP addresses of randomly selected super-peers. When the peer receives this list, it chooses a super-peer based on the less loaded to make a connection. If the super-peer still has connection degree, it will reply it. Once the new peer connects to the super-peer, it becomes one of ordinary peers of that super-peer and super-peer sends it a peer list which contains the part of the ordinary peers in the same network. When the ordinary peer wants to leave the system, it simply sends a message to inform its super-peer, which then updates the corresponding AVL tree-based index to show that the sharing files no longer forms part of the AVL tree-based index.

In accordance with the bandwidth of the peers, the BSP chooses an appropriate peer to become a super-peer. The peer should have a high bandwidth such as download speed of a 2 Mbps. Moreover, nodes behind NATs or firewalls typically cannot be accepted to become super-peer. If it achieves the bandwidth requirements, BSP will select the peer as a super-peer. Then, the BSP sends the forward and backward information to the new peer, or marks it as a redundant super-peer.

Table 3.3 An example of super-peer table

Super-peer ID	Super-peer IP	Forward connections	Backward connections	Redundant
0	IP_0	IP_1, IP_3	IP_4, IP_6	0
1	IP_1	IP_2, IP_4	IP_5, IP_0	0
2	IP_2	IP_3, IP_5	IP_6, IP_1	0
3	IP_3	IP_4, IP_6	IP_0, IP_2	0
4	IP_4	IP_4, IP_0	IP_1, IP_3	0
5	IP_5	IP_6, IP_1	IP_2, IP_4	0
6	IP_6	IP_0, IP_2	IP_3, IP_5	0

The BSP uses a table to manage the relationship of the super-peer. Table 3.3 includes the super-peer ID number, the super-peer IP address, the forward and backward information of each super-peer, and the redundant type of the super-peer. Here, the ID number is just the sequence that a peer enters the overlay network, and is mapped to the IP address with respect to each super-peer.

Meanwhile, the forward and backward connection fields represent the IP addresses of the corresponding partners of each super-peer, respectively. Finally, the redundant type field contains a value of 0 if the super-peer is not a redundant super-peer, otherwise it has a value of 1 if the peer has been delegated as a redundant super-peer by the BSP.

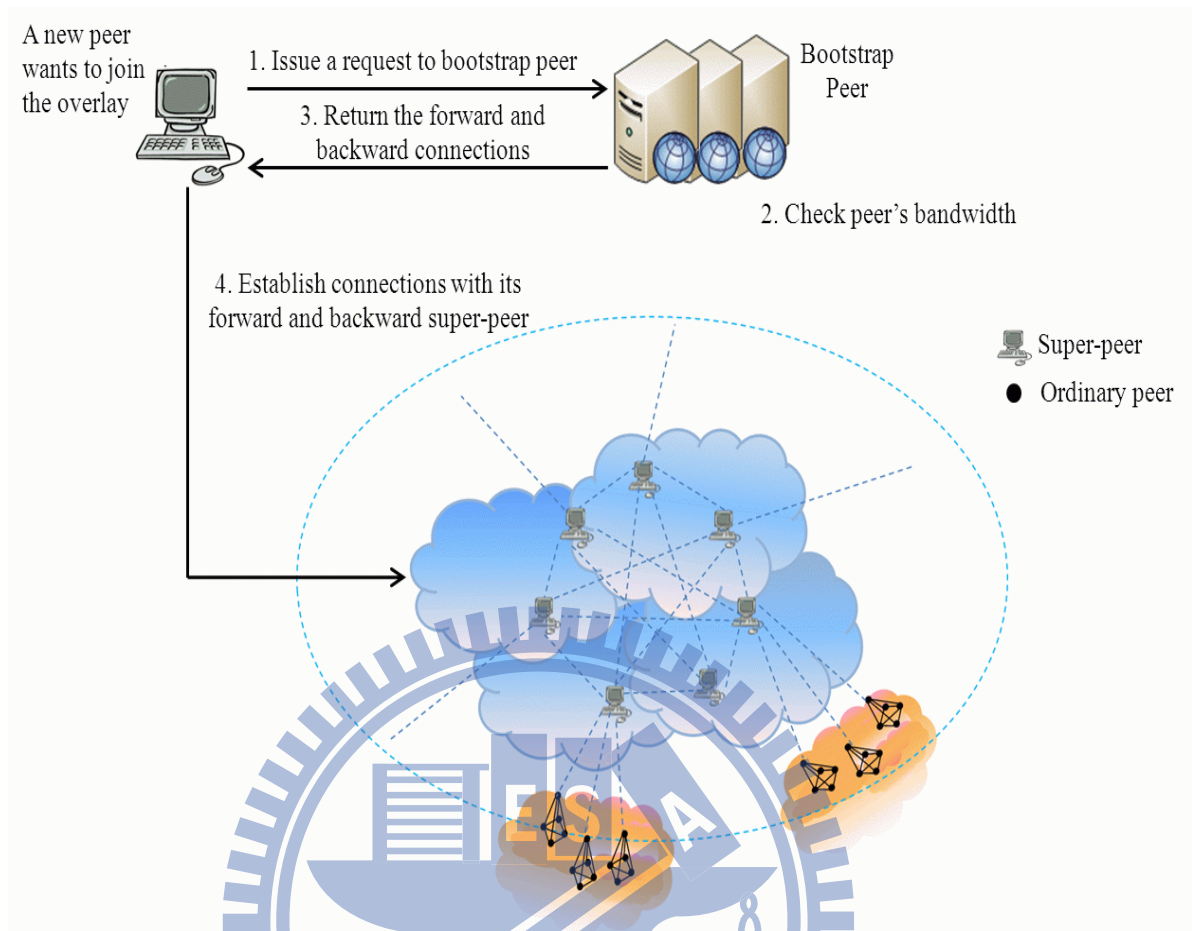


Figure 3.5 Network configuration of a new peer joining the super-peer.

Figure 3.5 shows the network configuration in which a new peer wants to join the P2P overlay network. Note that in this configuration, a network overlay topology has already been established by the BSP and the perfect difference overlay is composed of super-peers with an order of $\delta = 2$. In addition, the new peer (with an address IP_6) has a bandwidth higher than required for a super-peer. By verifying the download bandwidth of the peer, BSP selects it to become a super-peer. The BSP adds IP_6 in the super-peer table, and sends IP_6 information, such as the corresponding forward partners of IP_0 and IP_2, backward partners of IP_3 and IP_5, and is a redundant super-peer or not.

3.2.2 New Super-peer Join

Any peer with high bandwidth entering the P2P overlay network sends a joining request with its bandwidth information and IP to the BSP. After checking the bandwidth quality, the BSP accepts the peer as a super-peer, and appoints the new super-peer the appropriate forward and backward partners.

When the number of super-peers is larger than the value $(\delta^2 + \delta + 1)$, it means that PDG overlay is filled with active super-peers. If a new coming peer can satisfy the bandwidth requirement to become a super-peer, the BSP will mark the peer as a redundant super-peer, and is allowed to connect to the overlay. When the total number of super-peers increase to a threshold value, $1/2 [(\delta^2 + \delta) + (l^2 + l)]$, in order to make full use of the bandwidth capability of the redundant super-peers and increase system scalability, current perfect difference sets (PDS) is extended to the successor PDS order and the super-peer overlay is extended accordingly.

Thus, the BSP first assigns the new super-peer a new peer ID into the super-peer table. It then updates the status of the new super-peer and all of the redundant super-peers to 0. Next, the BSP calculates and updates new forward and backward partners based on the new order δ in the super-peer table for these active super-peers. Finally, the BSP informs the new super-peer about the forward partners, the backward partners, and the status. We give an example to describe the overlay topology extension.

In the initial set-up phase (i.e. no super-peers in the overlay network), the BSP utilizes a min-order PDS (i.e. an order of 2) to construct a basic super-peer overlay network for a maximum of 7 super-peers. Assume that there are 9 new peers fulfill the bandwidth requirements to be a super-peer, since the number of new peers exceeds the number of available spaces in the overlay network, the former 7 peers are assigned as super-peers, and the remaining peers are appointed as redundant peers.

Later, when a new incoming peer wishing to become a super-peer enters the system, it will result in the total number of super-peers, including active, new incoming, and redundant super-peer, exceeding a threshold $9(= (6+12)/2)$. The BSP extends the super-peer overlay topology using a PDS with an order of 3, thus allowing for super-peers up to 13. Therefore, the redundant super-peers and the newly joining one are appointed as new super-peers and are informed about the IP addresses of their forward and backward partners by BSP. At this moment, 10 active super peers exist in the newly extended configuration.

3.2.3 Super-peer Leave

A super-peer sends a leave message to both the BSP and all of its ordinary children peers, when it is going to leave the P2P system. The BSP selects one of the proper redundant super-peers as a super-peer to take place the leaved one. The BSP assigns the peer ID to it, the forward and backward partner information of the leaving super-peer, and the active status. The BSP then replaces the leaving super-peer IP with the new super-peer IP. Finally, the BSP sets the active state of the redundant peer and updates other active peers to update their connection partner records correspondingly.

Having received a leave message from a super-peer wishing to disconnect from the P2P overlay, each ordinary peers re-enter to the overlay by choosing one of the super-peers with the quickest response time in its super-peer list. When the number of super-peers is below the threshold $(\delta^2 + \delta + 1)$, there will be no enough super-peers to take over the leaving peers in the overlay network. Since the current number of super-peers in network is not sufficient, some of the super-peers lose their forward or backward partners. As a result, some of the super-peers may fail to receive the messages delivered by the other super-peers in the overlay network.

To overcome the effect, when the number of super-peers decreases to the threshold, $1/2 [(\delta^2 + \delta) + (l^2 + l)]$, the order of the current PDS is reduced to the predecessor of the PDS, and the super-peer overlay topology is reduced consequently. Then BSP computes and updates new forward and backward partners based on the new order δ in its super-peer table for those active super-peers and sets the status of those redundant super-peers to 1. Finally, the BSP notifies active super-peers of the forward and backward partners and redundant super-peers about the status and the addresses of some randomly selected super-peers.

We illustrate an example to describe the overlay topology reduction, consider 10 active super-peers existing in a super-peer overlay network using a PDS with an order of 3. If one active super-peer sends a leaving message, the BSP reduces the topology since the number of super-peers equals the threshold $9(= (6+12)/2)$. The BSP appoints a min-order PDS (an order of 2) to reduce the current super-peer overlay network, thus allowing for super-peers up to 7. It assigns new peer ID to the remaining super-peers. The super-peers with peer ID less than 7 are appointed as active super-peers to participate in the reduced topology. Others are appointed as redundant super-peers. At this moment, 7 active super-peers and 2 redundant super-peers exist in the system.

3.3 Numerical Analysis

In this section, we present a numerical study to illustrate the validity of the theorems presented in the previous sections. We evaluated the existing mechanisms in terms of the query success ratio, number of lookup query flooding messages and average delay. All the results reported in this thesis are the average of 10 simulation runs.

Before we analyze the performance, we define a parameter p_a , the probability of the resource recorded in the AVL tree and connection degree τ . For a two-layer unstructured P2P system, super-peer layer is with mesh-based structure, the location of a data item is arbitrary, and it uses flooding to perform a best-effort search.

$$1 + \tau + \tau^2 + \dots + \tau^{ttl} = \frac{\tau^{ttl+1} - 1}{\tau - 1} \quad (1)$$

For super-peer layer with pure-PDG structure, whether the resources exist or not, it always sends the query messages, which is

$$p_a \times \tau + (1 - p_a) \times (1 + \tau + \tau^2) \quad (2)$$

For super-peer layer with AVL-PDG structure, we want to further reduce the lookup messages between super-peers. The range of the flooding which is

$$\begin{cases} 1 & \text{, if the resource can't be matched} \\ p_a \times \tau & \text{, if the resource is in the same local area} \\ (1 - p_a) \times (1 + \tau + \tau^2) & \text{, if the resource is in the other network} \end{cases} \quad (3)$$

Chapter 4 Simulation and Numerical Results

In this chapter, we present the simulation and show the results to see the difference in performance between the random mesh-based, hierarchical unstructured P2P system and our proposed scheme, as well as to see how much it has been improved. From various aspects of performance concern, we show that our proposed approach is practical and works well.

4.1 Simulation Environment and Simulation Setup

We perform simulation with NS-2 (version 2.27) simulation tool [15] with GnutellaSim to evaluate our proposed method. GnutellaSim is designed and developed by the Networking and Telecom Group College of Computing Georgia Institute of Technology Atlanta, GA. GnutellaSim is a scalable packet-level Gnutella simulator that enables the complete evaluation of the Gnutella system with a detailed network model.

GnutellaSim is based on a framework we designed for packet-level peer-to-peer system simulation, which features functional isolation and a protocol-centric structure, among other characteristics. The framework is designed to be extensible to incorporate different implementation alternatives for a specific peer-to-peer system and is portable to different network simulators.

We adopt Gnutella protocol as our basic architecture in the simulation and add our proposed approach to the basic scheme. Also, we validate the performance and improvement through the simulation result.

4.1.1 Simulation Environment

The topology considered for simulation as the network model is presented in Figure 3.4. It consists of a bootstrap peer (BSP), super-peers and ordinary peers. When a new peer wants to enter the overlay, it will send a request to the BSP. Afterward BSP sends the peer a list containing the addresses of randomly selected super-peers. Then the newly joining peer starts to query the resources it needs. Regardless of different simulation scenarios, the general parameters are the same and are presented in Table 4.1.

Table 4.1 Experimental environment

Parameter	Value
Peer Number	5,000
Simulation Time	1,000(sec)
Number of files	100,000
Query frequency	5 per sec
Super-peer Download Bandwidth	> 2 Mbps
Super-peer Upload Bandwidth	> 1.5 Mbps
Peer Max degree	2
Connection link delay	1~10(ms)

Table 4.2 Peer Bandwidth Distribution

Peer Bandwidth Distribution		
Upload Bandwidth	Download Bandwidth	Ratio of Peers
>1500 Kbps	> 2000 Kbps	10%
1000 ~ 1500 Kbps	1500 ~ 2000 Kbps	60%
< 1000 Kbps	< 1500 Kbps	30%

4.1.2 Simulation Setup

Table 4.1 shows the system environment setting in simulation. Each network topology is composed of 1,000 nodes, and each node is assigned to be either a super-peer or an ordinary peer randomly. The ratio between the number of super-peers and the total number of peers is set to 10%. We set super-peer's download bandwidth to at least 2 Mbps and upload bandwidth at least 1.5 Mbps. We set ordinary peer's download bandwidth to no more than 1.5 Mbps and connection degree is 2. That means the ordinary peer can only connect to 2 super-peers at the same time.

We set the peers with heterogeneous link capacities such that 20% of the peers have the highest link capacities, 20% of them have the lowest link capacities, and 60% of them have the medium link capacities, as shown in Table 4.2. Each connection's link delay is randomly set between 1 and 10 ms. We run the simulation 10 times to get its average result and the duration is 1000 seconds each time.

4.2 Numerical Results

We present our experimental results on NS2, an event-based overlay network simulator. For our simulations, we modified an implementation of Gnutella [13].

4.2.1 Performance of the Average Traffic

Figure 4.1 illustrates the variation of the broadcast messages with different simulation times in a mesh-based overlay, a pure PDG overlay and an AVL-list with PDG overlay network, respectively. Thus, Figure 4.1 show that the AVL-list with PDG overlay network has the lightest broadcast overhead. When a traditional mesh-based overlay is used, queries for unavailable files can generate an unbounded traffic load. In contrast, the traffic load is bounded when AVL-list is used.

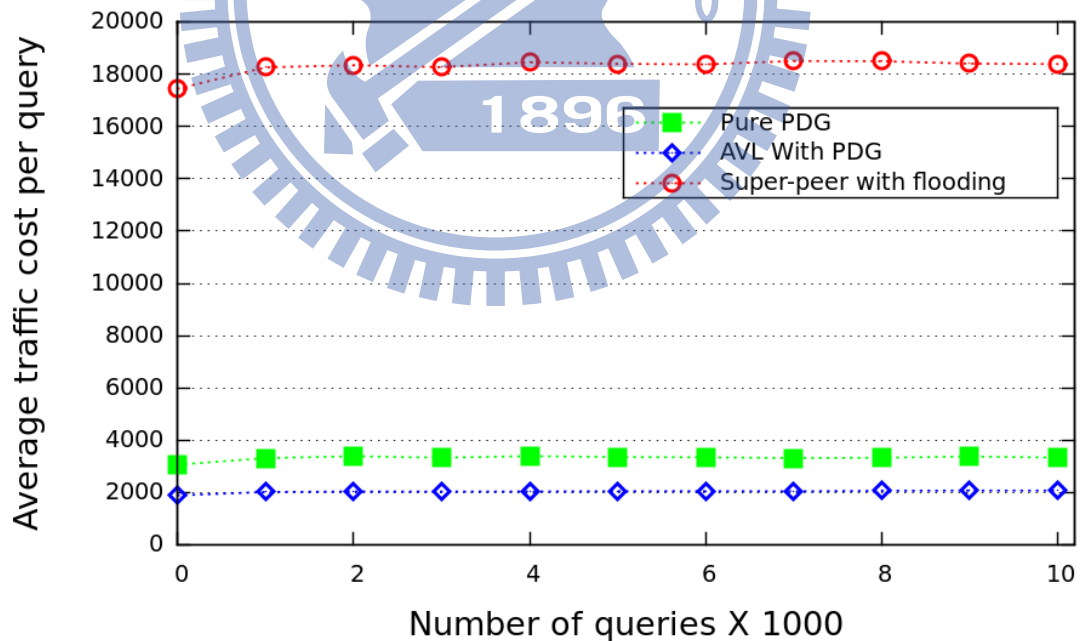


Figure 4.1 Comparison of number of broadcast search messages incurred in mesh-based and PDG overlay networks

In Figure 4.2 to Figure 4.4, the cumulative distribution function (CDF) of different algorithm is used to demonstrate performance difference, and the corresponding average values are also summarized in Figure 4.5 and Table 4.3. As compared to the other two schemes, our AVL-list with PDG algorithm shows a better performance in the sense that large portion of peers in the network experience a higher successful ratio and less amount of messages.

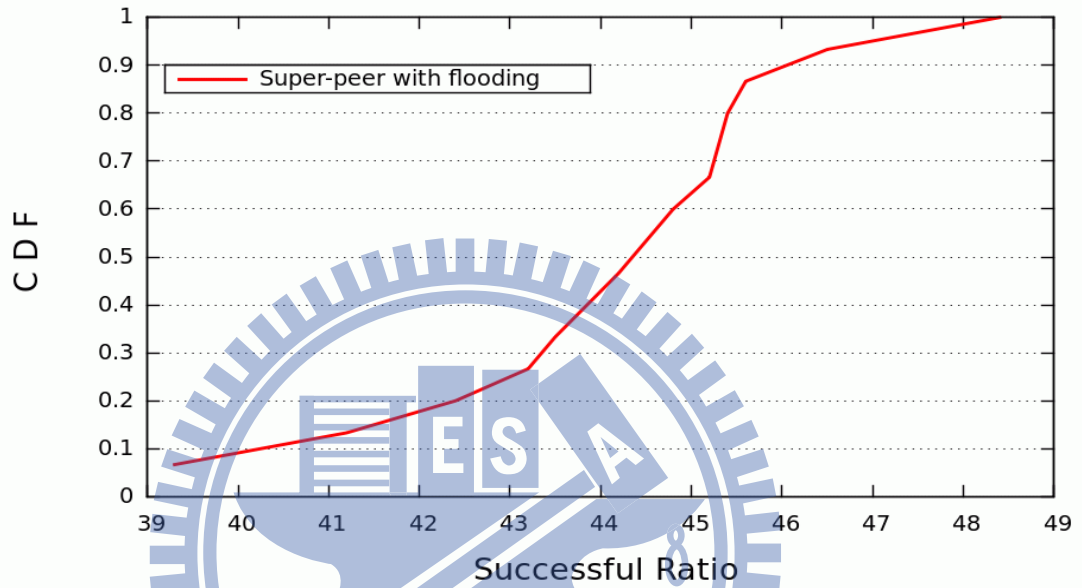


Figure 4.2 Query successful ratios in mesh-based networks.

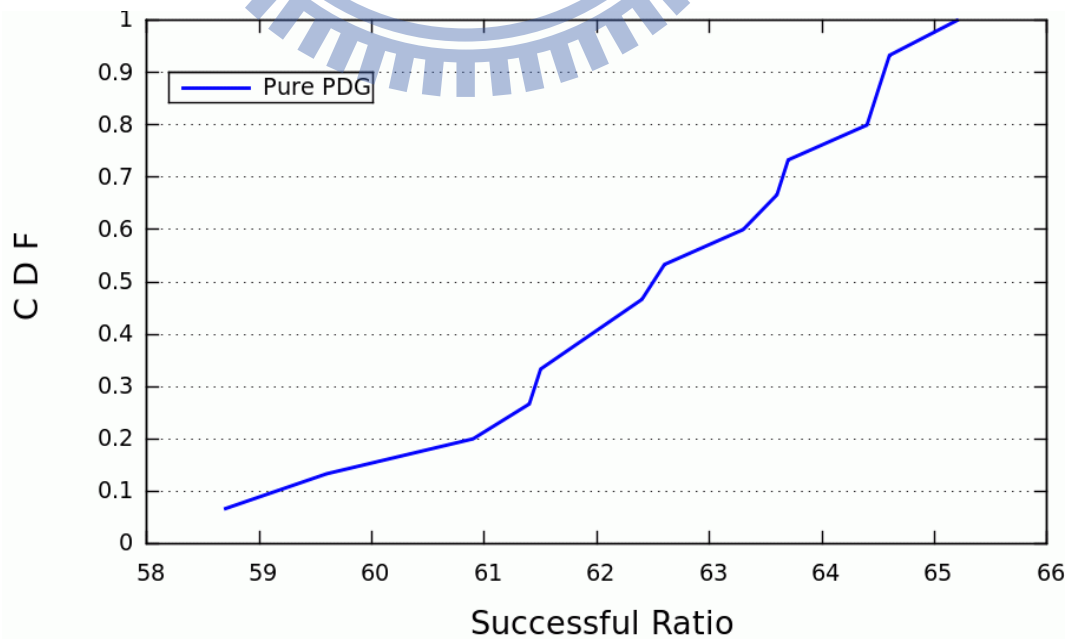


Figure 4.3 Query successful ratios in pure PDG networks.

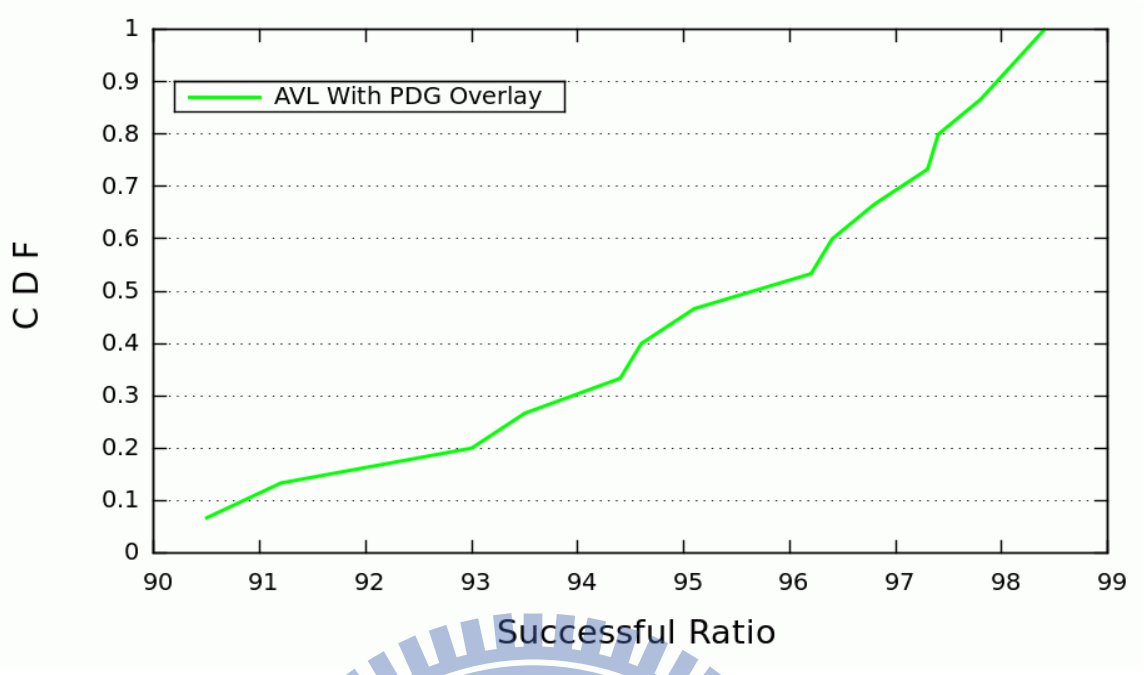


Figure 4.4 Query successful ratios in AVL PDG networks.

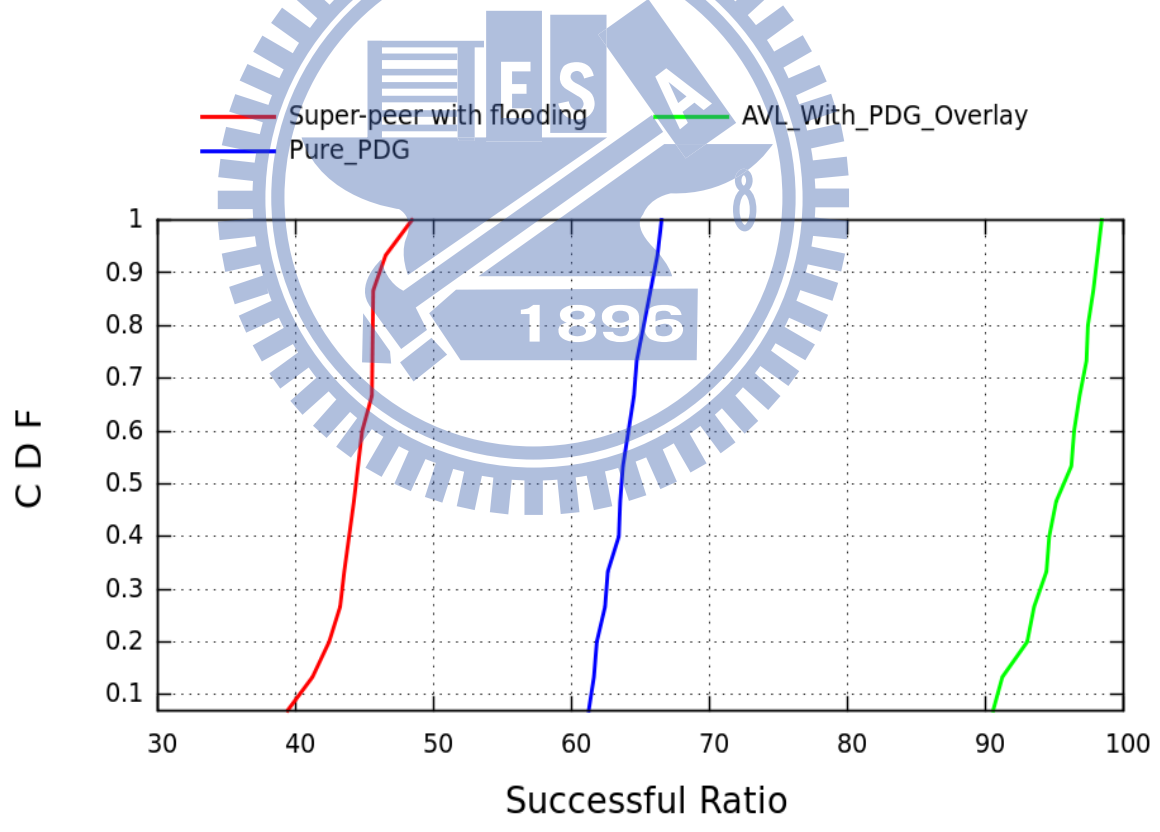


Figure 4.5 Comparison of query successful ratio in mesh-based and PDG overlay networks.

In Figure 4.1 and Figure 4.5, we show the query success rate of the same experiments. In Figure 4.5, we find that the results clearly demonstrate the success rate of the AVL-list with PDG overlay is significantly higher than mesh-based overlay and pure PDG overlay. Since the AVL-list limits the queries which can't find in the list. It improves about 40% and 50% of query success ratio compared to the pure-PDG and the mesh-based overlay, respectively.

Table 4.3 Average value comparison

Average Value Comparison		
	Volume of Traffic	Success Ratio
Mesh-Based	100%	45.6%
Pure-PDG	18.15%	62.5%
AVL-PDG	11.09%	96.6%

4.2.2 Performance of the Network Traffic

To illustrate the effect of AVL-list with PDG flooding algorithm on network traffic, we generate queries at a speed of five queries per second. The total number of messages in experiment, consisting of: queries, replies to queries, and other control messages used to discover nodes, are used to represent network traffic.

Figure 4.3 shows the comparison of network traffic between mesh-based overlay and PDG-based overlay. When a traditional mesh-based overlay is used, queries for unavailable files can generate an unbounded traffic load. In contrast, the traffic load when AVL-list used is bounded.

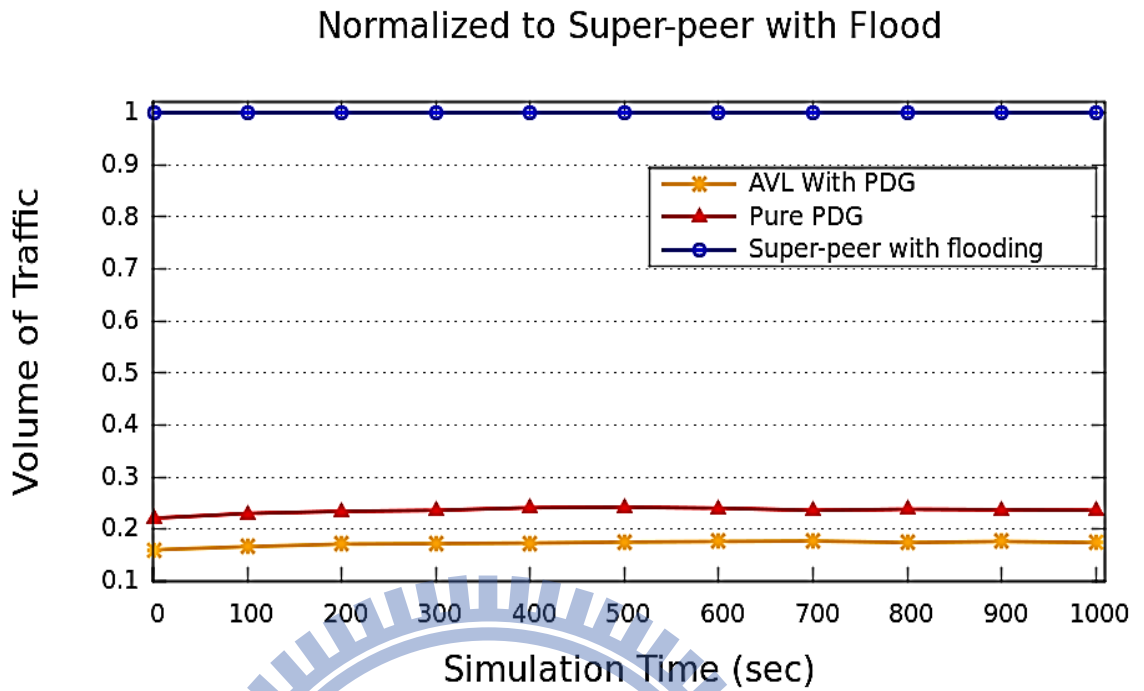


Figure 4.6 Network Traffic in mesh-based and PDG overlay networks.

4.2.3 Performance of the Response Time

We define response time as the minimum number of hops for the query reply to be forwarded back to the querying peer. Figure 4.4 shows that, with the help of the PDG forwarding such that most of the queries can be solved in the super-peer overlay, which has smaller diameter than the entire P2P network. The average response time keeps between 1.5 and 2 hops with 5000 peers in the system. The simulation result shows that the average response time decreases from 4.3 hops to 1.7 hops. The overall user perceived response time can be reduced by 60 percent.

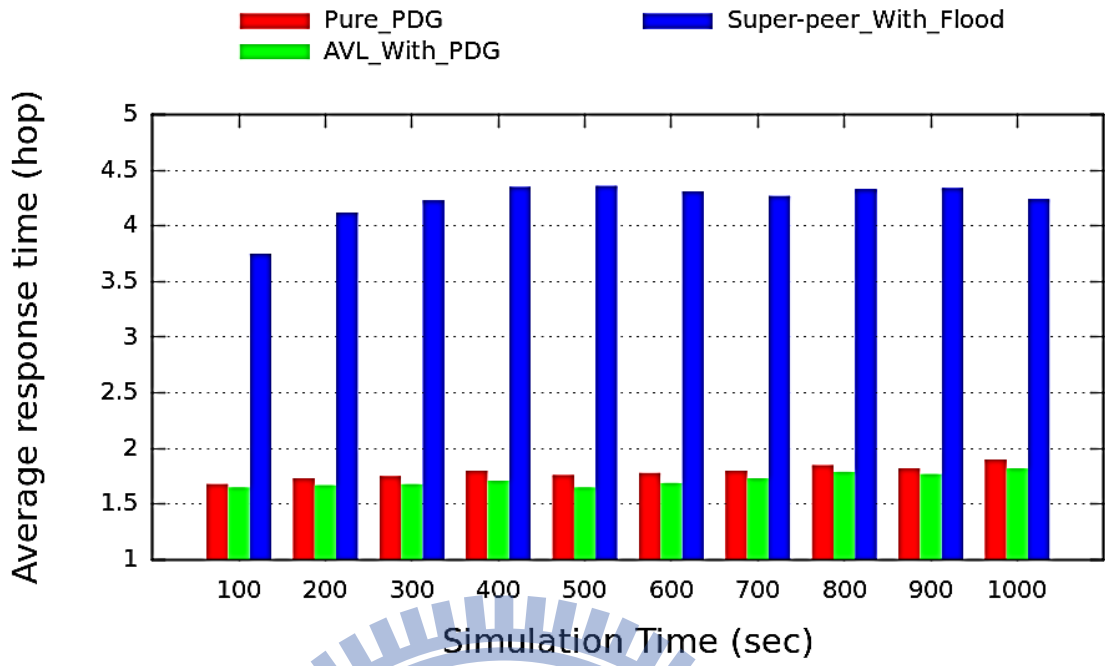
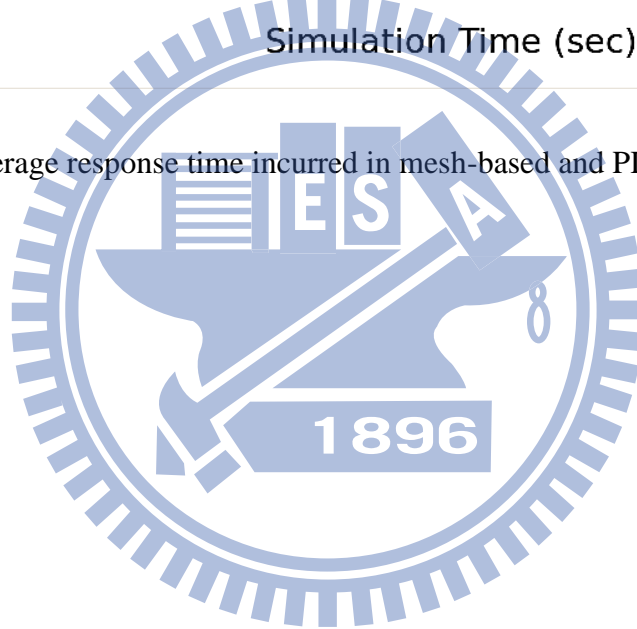


Figure 4.7 Average response time incurred in mesh-based and PDG overlay networks.



Chapter 5 Conclusion and Future Works

Our work suggests that unstructured peer-to-peer systems have excellent scalability and reliability properties. In the hybrid P2P system setting, our work shows that an unstructured system can be deployed to reduce the traffic load at a server without overwhelming its clients. In the context of pure systems, our work has identified a query forwarding mechanism that is both reliable and scalable.

In this thesis, we propose a novel and efficient search mechanism for multi-layer P2P systems using a Multi-hop Index Replication with PDG forwarding algorithm, a multi-hop index is proposed for enhancing the efficiency of communication overhead. And show that it is not only reliable, *i.e.*, if certain content is in the system, it successfully locates all files and the search should be completed with reasonable guarantees, but also scalable, *i.e.*, peers have limited bandwidth, the traffic load generated by queries will be limited by super-peer.

The performance of the proposed super-peer overlay topology based on binary index search tree with a perfect difference graph has been benchmarked against a super-peer overlay topology based on a mesh graph using the flooding with TTL value 7 forwarding algorithm. The theoretical results have shown that the Multi-hop Index Replication with PDG -based construction scheme yield a higher query success ratio, a reduced number of search flooding messages, and a lower average hop-count delay. Through experimental results in our experiment, the proposed Multi-hop Index Replication with PDG-based multi-layer unstructured P2P overlay is an efficient P2P approach in the dynamic network environment.

In addition, peers participating in a P2P network are often heterogeneous in terms of their network bandwidth, storage space, and/or computational capability. It would be interesting for our future work to investigate how the heterogeneity affects our proposal. It would be

challenging to design an overlay formation algorithm aware of both the similarity of participating peers and the physical network topology.



Reference

- [1] S. Banerjee, C. Kommareddy, K. Kar, S. Bhattacharjee, and S. Khuller, “*Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications*,” Proc. IEEE INFOCOM ’03, pp. 1521-1531, Mar. 2003.
- [2] S. Sen and J. Wang, “*Analyzing Peer-to-Peer Traffic Across Large Networks*,” Proc. Internet Measurement Workshop, Nov. 2002.
- [3] Y. Chu, S. Rao, and H. Zhang, “*A Case for End System Multicast*,” Proc. ACM SIGMETRICS ’00, pp. 1-12, June 2000.
- [4] E. Brosh and Y. Shavitt, “*Approximation and Heuristic Algorithms for Minimum Delay Application-Layer Multicast Trees*,” Proc. IEEE INFOCOM ’04, Mar. 2004.
- [5] M. Freedman and R. Morris, “*Tarzan: A Peer-to-Peer Anonymizing Network Layer*,” Proc. ACM Conference on Computer and Communications Security, Nov. 2002.
- [6] S. Tewari and L. Kleinrock, “*Proportional replication in peer-to-peer networks*,” Proc. IEEE INFOCOM ’06, pp. 1-11, Apr. 2006.
- [7] W. Acosta and S. Chandra, “*Understanding the practical limits of the Gnutella p2p system: An analysis of query terms and object name distributions*,” in MMCN, 2008.
- [8] B. Parhami and M. Rakov, “*Perfect Difference Networks and Related Interconnection Structures for Parallel and Distributed Systems*,” IEEE Transactions on Parallel and Distributed Systems, VOL. 16, NO. 8, pp. 714-724, Aug. 2005.
- [9] B. Parhami and M. Rakov, “*Performance, Algorithmic, and Robustness Attributes of Perfect Difference Networks*,” IEEE Trans. Parallel and Distributed Systems, vol. 16, no. 8, pp. 725-736, Aug. 2005.
- [10] J.-S. Li and C.-H. Chao, 2010, “*An Efficient Super-Peer Overlay-Construction and Broadcasting Scheme Based on Perfect Difference Graph*,” IEEE Transactions on Parallel and Distributed Systems, VOL. 21, NO. 5, pp. 594-606, May. 2010.
- [11] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, “*Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications*,” IEEE/ACM Transactions on Networking (TON), VOL. 11, NO. 1, pp. 17-32, Feb. 2003.
- [12] Gkantisdis, C., Mihail, M., and Saberi, A., “*Random walks in peer-to-peer networks*,” Proc. IEEE INFOCOM ’04, pp. 7-11, Mar. 2004.
- [13] Gnutella Development Forum, the Gnutella v0.6 Protocol, http://groups.yahoo.com/group/the_gdf/files/, 2009.
- [14] BitTorrent, <http://www.bittorrent.com/>, 2009.
- [15] The Network Simulator - ns-2; <http://www.isi.edu/nsnam/ns/>
- [16] P. Ganesan, Q. Sun, and H. Garcia-Molina, “*YAPPERS: A Peer-to-Peer Lookup Service over Arbitrary Topology*,” Proc. IEEE INFOCOM ’03, pp. 1250-1260, Apr. 2003.

- [17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “*A Scalable Content Addressable Network*,” Proc. ACM SIGCOMM ’01, pp. 161-172, Oct. 2001.
- [18] B.T. Loo, R. Huebsch, I. Stoica, and J.M. Hellerstein, “*The Case for a Hybrid p2p Search Infrastructure*,” Proc. Workshop Peer-to-Peer Systems (IPTPS ’04), pp. 141-150, Feb. 2004.
- [19] Iwayan.info/Research/PeerToPeer/Papers_Research/P2P_Architecture/it02012.pdf
- [20] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, “*Making Gnutella Like p2p Systems Scalable*,” Proc. ACM SIGCOMM ’03, pp. 407-418, Aug. 2003.
- [21] F. Wang, Y. Xiongand, and J. Liu, “*mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast*.” In The 27th IEEE International Conference on Distributed Computing Systems (ICDCS’07), Toronto, Canada, June 2007..
- [22] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, Scott Shenker, “*Making Gnutella-like P2P systems scalable*,” Proc. ACM SIGCOMM ’03, pp. 407-418, Jan. 2003.
- [23] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan, “*Measurement, Modeling and Analysis of a Peer-to-Peer File-Sharing Workload*,” Proc. ACM Symp. Operating System Principles (SOSP), Dec. 2003.
- [24] Stratis Ioannidis, Peter Marbach, “*Absence of Evidence as Evidence of Absence: A Simple Mechanism for Scalable P2P Search*,” Proc. IEEE INFOCOM ’09, pp. 576-584, Apr. 2009.

