國立交通大學

資訊科學與工程研究所

碩 士 論 文

根 據 使 用 者 興 趣 來 增 進 非 結 構 性 同 儕 網 路 的 搜 尋 效 能

Improving Search Efficiency in Unstructured P2P Networks Based on Users' Interest

研 究 生：劉宜政

指導教授：林寶樹　教授

中 華 民 國　九 十 八　年 二 月

根據使用者興趣來

增進非結構性同儕網路的搜尋效能

Improving Search Efficiency in Unstructured P2P Networks Based on Users' Interest

研　究　生：劉宜政　　　　Student：Yi-Chen Liu

指導教授：林寶樹　　　　Advisor：Bao-Shuh Lin

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

Feb 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年二月

根據使用者興趣來增進非結構性 P2P 網路的搜尋效能

學生:劉宜政　　　　　　　　指導教授: 林寶樹

國立交通大學資訊科學與工程研究所

摘要

P2P 檔案分享系統是現在消耗最多網路資源的應用之一，如何有效率的搜尋是一個挑戰。在非結構性的 P2P 系統中， 以 Gnutella 爲例子，搜尋是透過洪流(Flooding)的機制來找尋資源。雖然這是個簡單且穩固機制, 但是這會產生很大的網路流量，當系統擴規模大的時候，會非常的沒有效率。 在本篇論文中，我們提出了一個有效率找尋資源的方法。 系統的成員會在 Gnutella 的網路上，逐漸的自我組織成結構。 本篇提出的方法核心爲，我們假設，發出相同搜尋的成員們，他們可能會有類似的興趣或喜好。我們將這些成員們組織在一起，搜尋的訊息會先送給這些成員，因爲他們有可能在過去搜尋的時候，已經找到所需要的資源。本篇論文，將一般系統視爲是負擔的搜尋訊息，

當成資源來運用。我們也思考搜尋的本質。一般搜尋，都是先有需求，然後尋找資源。然而，是否有可能反過來，讓擁有資源的成員，找出誰有需求呢？本篇提供了一種解答。我們也透過模擬，加以證實我們提出的方法，可以增進搜尋效率並且降低網路流量。

Improving Search Efficiency in Unstructured P2P Networks Based on Users' Interest

Student : Yi-Chen Liu   Advisor：Bao-Shuh Lin

Abstract

P2P file-sharing systems consume significant bandwidth nowadays. A fundamental challenge is how to locate resources efficiently. In unstructured P2P systems, take Gnutella for example, search is performed by flooding to find resources. Although simple and robust, flooding generates tremendous traffic, and is also not scalable. In this paper, we propose an efficient method to tackle this problem. Peers will gradually self-organize into an interest-based overlay on top of Gnutella. The basic concept is that peers request the same files are likely to have similar interests. Query messages are first sent to those peers that have issued the same query. Chances are the requested resources may have been retrieved through previous searches. We turn query messages, while other researches regard them as merely overheads, into useful resources. Furthermore, we also take an eye on the nature of search. In common practices, we want resources, and we locate them via search and pull it back. However, is it possible that we have the resources, and push them to those in need? We provide an answer as well. Our algorithm, called Déjà vu, can improve search efficiency and alleviate network overheads, as confirmed by simulation.

致謝

首先我要感謝林寶樹教授，引領我踏入了知識的殿堂。在他的指導下，我才恍然大悟，原來一門科學和研究，是如何嚴謹並且小心的去求證和分析。宛如父親的嚴肅慈祥，他在人生上也帶給了我很多啓發和轉折。同時我也感謝三位口試委員，林一平教授、曾煜棋教授以及蔣村杰教授，在百忙之中抽空給予我建議，讓我一睹大師們的風采。

我要感謝工研院資通所的先進們給予我的指導，以及才華姐的關心和照顧。我也要感謝俊偉，身爲研究所同學，無論在生活還是課業上，我們一直都是共患難，給予彼此支持。再來，我要感謝 Vincent、Calvin、Eddy、小育、米格魯和會長等人，在我論文和人生中遇到瓶頸時，也不斷給予我建議和鼓勵。我同時也要感謝清交泳隊的夥伴，爲我研究生活添加了不少歡笑和回憶。

接著，我也要感謝爲我加油集氣的易總、姿含、獅子頭、餅乾、白、土蛋、柏齊、嘉懋、邱寶、歐弟、偉、阿推、阿宏、阿部、阿路、阿達、阿賢、阿樂、阿狗、小許、小江、小樂、小由、小維、Ann、Alan、Albert、Ben、Clark、Daniel、Draz、Edery、Gabriel、George、 SATA、Martin、Ian、IN、Issac、Joan、Kai、Kaku、Kelvin、Leon、Sirece、Winifred、Zodiac 等人。

最後，我要感謝我的父母，從小到大給予我的照顧和關心。要感謝的人太多，就謝天吧！僅以本論文獻給所有關懷我的人們，我不勝感激。

Content

List of Figures

List of Tables

Chapter 1 Introduction

Peer-to-Peer network has been a great success in recent years. Application-level overlays are established on top of the internet to facilitate file-sharing. Gnutella[1] is among one of the famous applications. One fundamental challenge of Peer-to-Peer network is how to discover resources efficiently.

Gnutella uses flooding to locate content as depicted in Figure 1-1. Each query is tagged with a maximum Time-To-Live (TTL) to limit the number of hops it can travel. When a peer receives a query message, it first checks if it has the requested resource. If it has, it responds to the query initiator. Otherwise, it passes the query message to its neighbors. Gnutella faces serious problems when the network is very large [2].

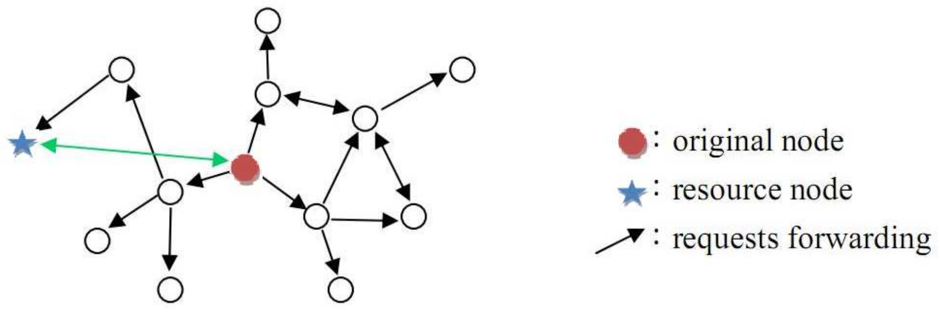


Figure 1-1 Gnutella

Our design is to overcome the major weakness of Gnutella, which is not scalable, while retaining its simplicity and robustness. We propose interest-based Déjà vu. The meaning of Déjà vu is "an individual feels as though an event has already happened or has happened in the near past." We borrow the meaning and redefine it in P2P search as "peers have Déjà vu on its current search that other peers have searched it."

The design is under the presumption that peers who issue the same query are likely to have similar interest. When a peer starts a search, it has Déjà vu that other peers have searched it. Query messages are first sent to peers have Déjà vu as in Figure 1-2. Chances are that the requested resources may have been retrieved through previous search. Déjà vu is learned simultaneously when flooding is performed. Consequently, no extra overhead is imposed.

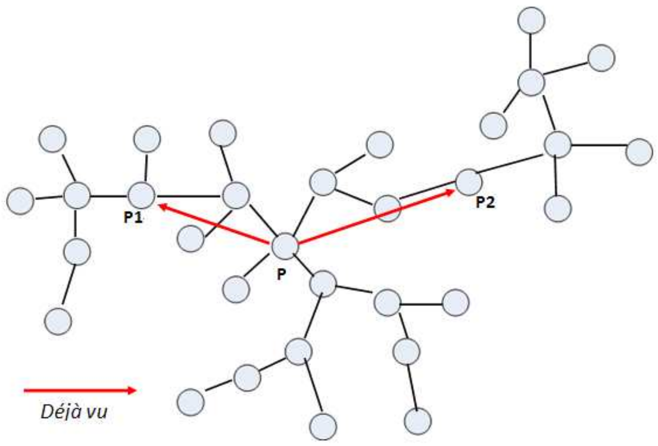


Figure 1-2 Peer searches via Déjà vu first

Based on Déjà vu, a mutually beneficial strategy for peers called synergy is proposed. Resources are actively pushed to those in need as shown in Figure 1-3, instead of random replication [3], path replication [4], or one hope replication[5] as in some researches. Those treatments without discrimination may result in low efficiency for local resources. Synergy is regarded as an incentive for peers to adopt Déjà vu approach as well.

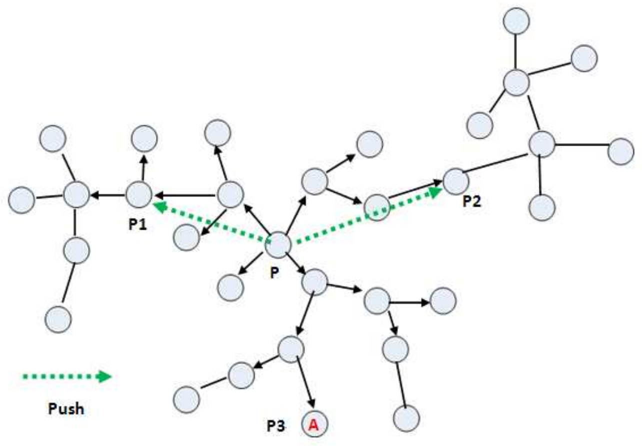


Figure 1-3 Actively pushes resources to those in need

We compare the performance with and without Déjà vu on Gnutella and on Shortcut. Simulation reveals that our Déjà vu approach improves the performance by boosting hit ratio, alleviating network loads and accelerating the speed to locate resources.

The subsequent parts of this paper are organized as follows. In chapter two, we discuss the background knowledge and related works that improves performance of the P2P network. In chapter three, the

Déjà vu architecture is presented. In chapter four, we show the simulation results. Finally in chapter five, we give conclusions.

Chapter 2 Related Work

In the following section, the P2P networks and some approaches to improve search performance are introduced.

2.1 Peer-to-Peer Networks

The essence of P2P networks is resource sharing. Everyone in the network contributes its resources and benefits from others at the same time. P2P networks can be categorized into centralized p2p networks and decentralized P2P networks.

For centralized P2P networks, it requires a central server to manage and take control of the network, such as Napster[6] in Figure 2-1. The main advantage is its high search efficiency. Resources and related peers can rapidly be found by querying the server. As a double-blade sword, the disadvantage lies in server as well. If the server goes wrong, the whole network would not function. Moreover, as the scale of the network grows, the server's capability will become the bottleneck.

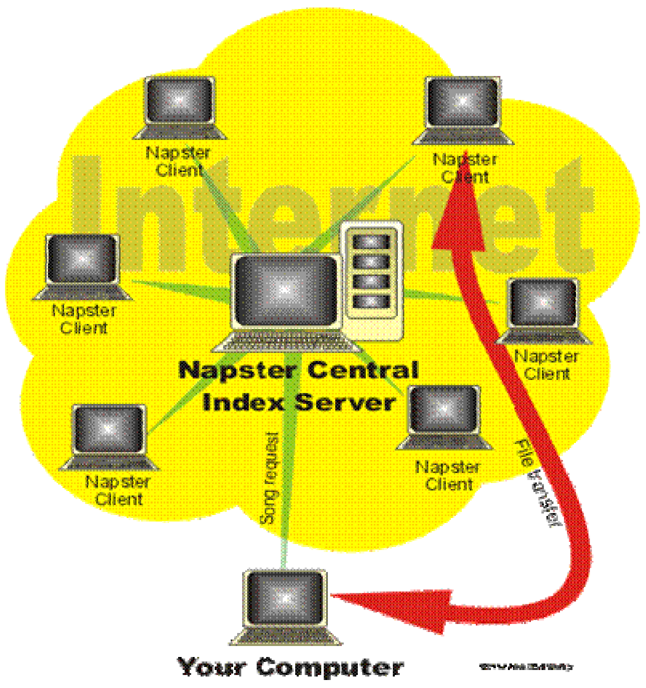


Figure 2-1 Napster

For decentralized p2p networks, there is no central server. Search is achieved by the cooperation of the peers. The search efficient is relatively lower, but the system is robust. In decentralized p2p networks, it can be further divided into structured and unstructured p2p networks.

There are also hybrid P2P networks between centralized and decentralized p2p networks such as KaZaA[7] and Skpye[8] . Hybrid P2P networks are built on the basis of decentralized p2p network, it utilizes some peers (Super-nodes) that have stronger powers to act like servers as shown in Figure 2-2. Search efficiency in decentralized p2p network is thus improved. Nevertheless, super-nodes still remain a bottleneck.

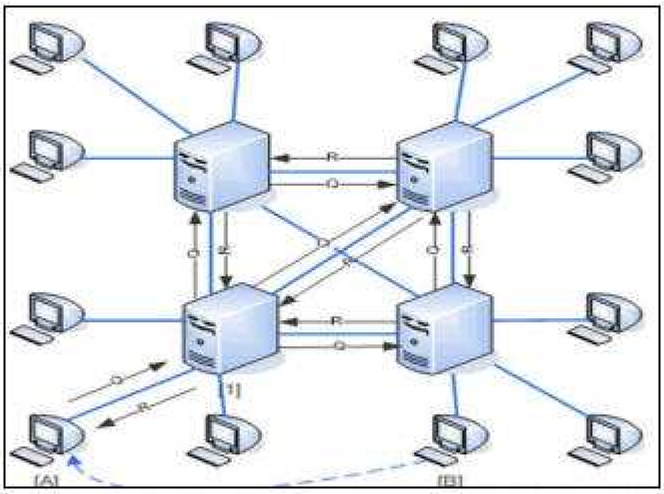


Figure 2-2 A Hybrid P2P network

Decentralized p2p networks are the trend of emerging p2p researches. Next, we will introduce some decentralized p2p networks.

2.1.1 The Structured P2P Networks

Structured p2p networks utilize a globally protocol (DHT – Distributed Hash Table) to ensure that any node can route a search to the peers that have the resources. For example, Chord[9], CAN[10], Pastry[11]. Although DHTs are elegant and scalable, their performance under dynamic environment, which is common in p2p networks, remain unknown[12] . Figure2-3 is what a Chord ring looks like.

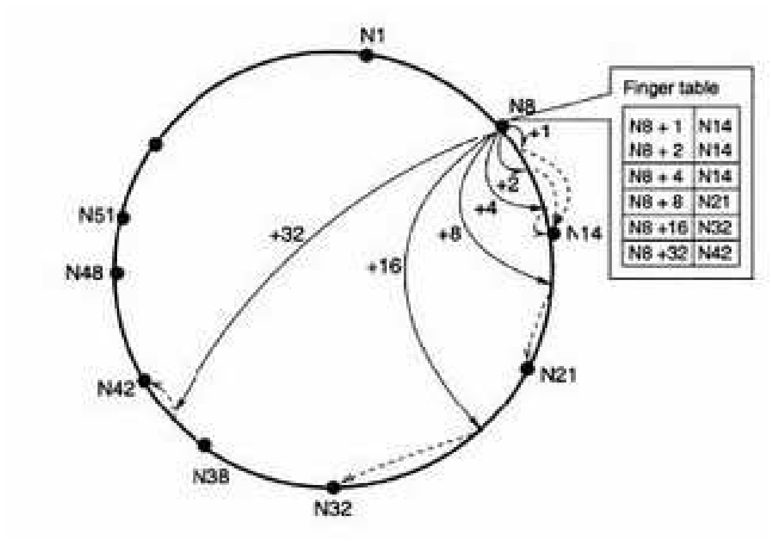


Figure 2-3 Chord ring

2.1.2 The Unstructured P2P Networks

For unstructured p2p networks, each peer maintains a partial view and performs local actions. The peers will gradually organize themselves into an overlay. The characteristics of unstructured p2p network are simplicity and robustness, and attract more practical use in the internet accordingly, for instance, Gnutella.

2.2 Interest-based P2P Networks

Clustering peers with similar interests and forwarding query messages to peers that are more likely to have the requested resources, is an instinct way to improve search efficiency. However, how to capture and represent interest is a challenging task. Next, we will introduce some method to capture peers' interest.

2.2.1 Semantic Overlay

The inherent interest of a peer is formalized within a semantic framework in semantic overlay, for example [13], [14], [15]. Files are characterized by the topics. The distance between the topics or peers is calculated by some semantic similarity functions of various forms. The distance is usually numerical values, which is used to facilitate the search process. In [13], the similarity of two peers is calculated by the fraction of documents that they have for every topic. In [15], each file is represented by a semantic vector, and the similarity is measured by Euclidean distance.

2.2.2 Shortcut

Another way to exploit interest more implicitly is on the principle that, if p’ has particular piece of content that p is interested, then it is likely that p’ will have other pieces of content that p is also interested in. In [16], it introduce interest-based shortcuts. In Figure 2-4, peers that share similar interest create shortcut to one another. Peers then use shortcut to locate content. When shortcut fails, peers resort to flooding.

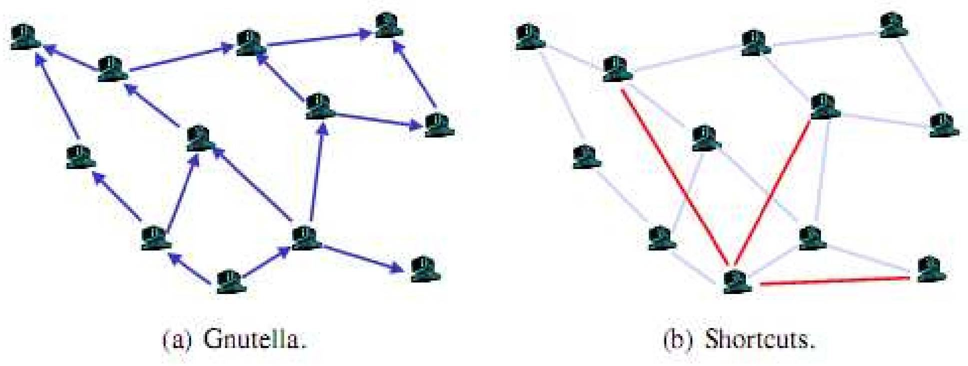


Figure 2-4 Content location path

2.2.3 Acquaintance

In [17], it introduces Acquaintance based on the same principle. However, it does not create additional links, but dynamically modifies the overlay network to reflect the shared interests of peers. Figure 2-5 demonstrates the topology adaptation.

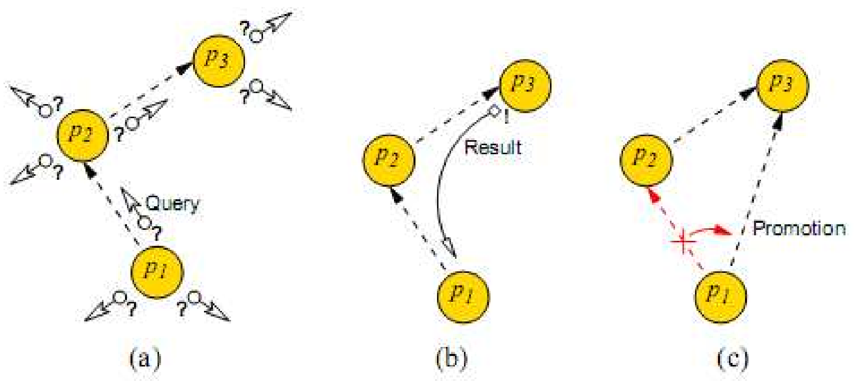


Figure 2-5 Basic principle of topology adaptation

Figure 2-6 is the topology at the beginning of simulation. After some generation of simulations, the peers will cluster into groups as in Figure 2-7.

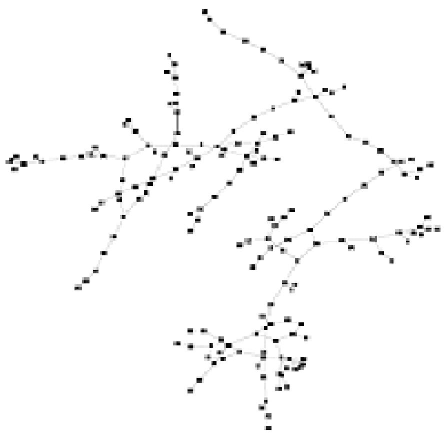


Figure 2-6 At the beginning of simulation

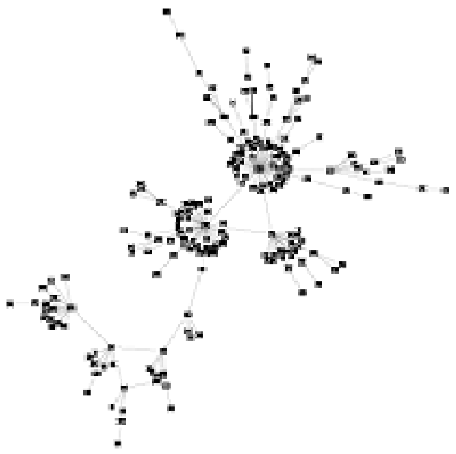


Figure 2-7 At the end of simulation

2.2.4 Popular Peer

In [18], it proposes Popular peers based on the principle of shortcut. A peer that shares many files may be connected by several shortcuts from others peers. If the number exceeds the threshold, it is promoted to a popular peer. In figure2-8, many peers have shortcuts on peer G, and G becomes a popular peer.

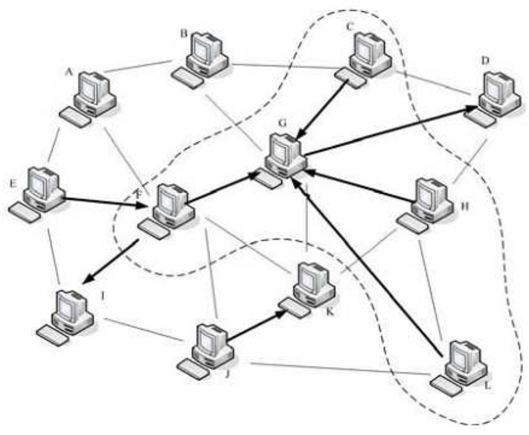


Figure 2-8 Popular peers

The search is first performed among popular peers, and then on shortcuts, and finally by flooding. In Figure2-9, Q is the query initiator. The query messages were first sent to popular peer P1. P1 forwards the query message to P2. If P1 and P2 don't have the file Q requests, Q will send the query to all the shortcuts except the popular peer. Finally, if the query fails in the popular peers and shortcuts, it can still be flooded on the unstructured overlay.

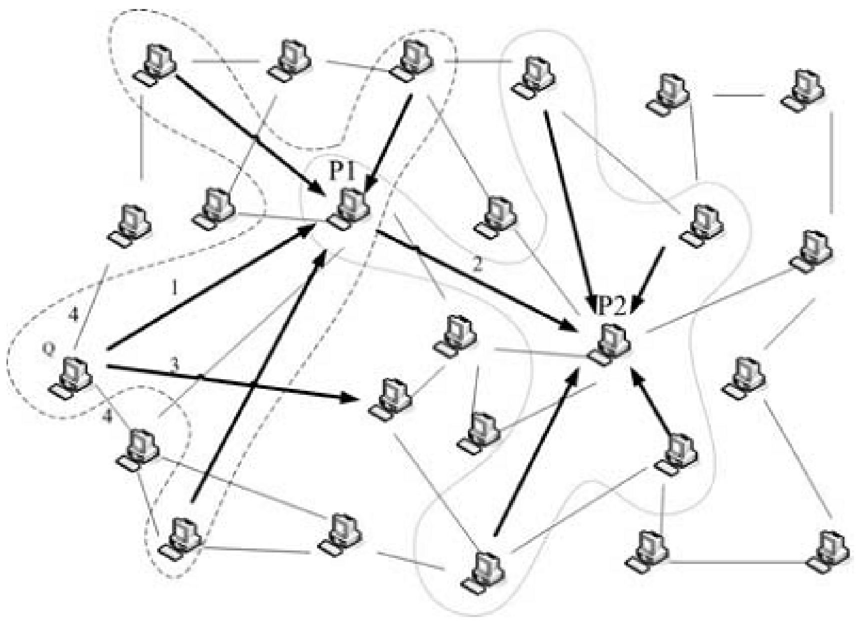


Figure 2-9 An Example of Search

Chapter 3 System Design

In this chapter, firstly we introduce our design philosophy. How and why we come out with this design. Then, we describe our approach – Déjà vu. Finally, some theoretical analyses and predictions are made.

3.1 Philosophy of Design

Our system design is based on two philosophies. First, “can we make use of what others take as overheads, and turn garbage into gold?” For most peer-to-peer systems, message overheads are considered as loads which consume tremendous network bandwidth. However, messages do carry information that can assist us to capture users’ interests. We will show how Déjà vu utilize messages to facilitate search.

Second, “can we inverse the process of search?” For common practices, we want resources, and we locate them via search and pull it back. However, is it possible that we have the resources, and actively push them to those in need? Synergy is proposed to provide this functionality.

We have a presumption that peers that issue the same query are likely to have similar interest. Our approach, Déjà vu, creates additional link on top of the p2p overlay. Déjà vu is implemented as a performance enhancement over the existing content location mechanism, such as flooding in Gnutella. Peers use Déjà vu to locate resources. Query messages are first sent to peers have Déjà vu. Chances are that the requested resources may have been retrieved through previous search. If the resource cannot be located through Déjà vu, it can always be located via underlying overlay. Consequently, Déjà vu does not affect the correctness and scalability of the underlying network.

3.2 Déjà vu Approach

The meaning of Déjà vu used in P2P search is "peers have Déjà vu on its current search that other peers have searched it." There are two types of Déjà vu – Déjà vu file, and Déjà vu peers. Déjà vu is a way to turn overhead into resources. Let's see how it works.

3.2.1 Déjà vu Discovery

When a peer joins the system, it doesn't have any information about others peers' interests. As long as connected to other peers, it learns others' interest when queries pass through. In Figure 3-1, when P1 queries file A through flooding, the peers on the searching path all have Déjà vu on P1. That is, the peers on the path know about the fact that P1 have searched for A. Déjà vu discovery can be done simultaneously when receiving query messages. Table 3-1 is the Déjà vu of all peers on the flooding path includes P.

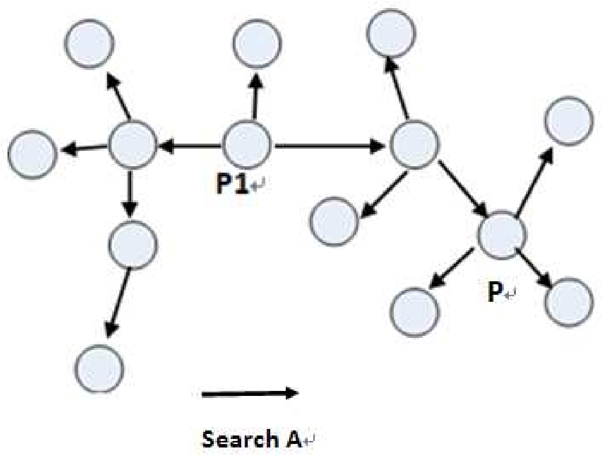


Figure 3-1 Déjà vu Discovery

Déjà vu file	Déjà vu peer
A	P1

Table 3-1 Déjà vu of P1

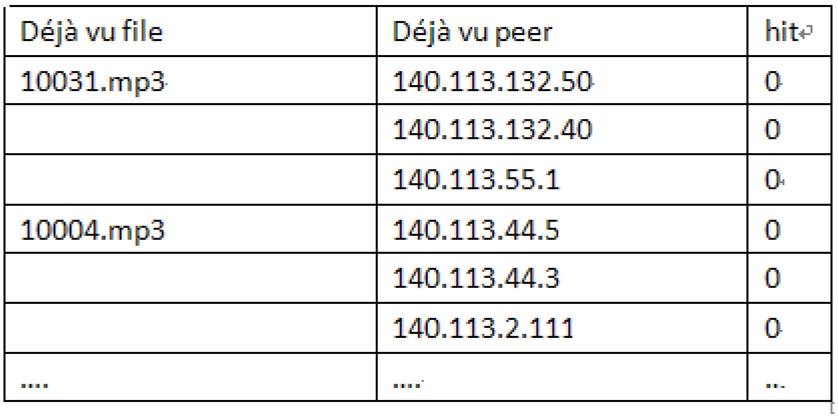
There are two kinds of Déjà vu –Déjà vu file and Déjà vu peers. A D Déjà vu file can be related to several Déjà vu peers and vice versa. Since we want to search files, we use Déjà vu file as an entry to find Déjà vu peers list. The algorithm of adding Déjà vu is illustrated in Figure 3-2.

```
Add  dejavu()
{
        For (i=1 to max déjà vu )
        {
                If (déjà vu file exists)
                Append the current déjà vu peer to the déjà vu peer list.
                Return 0;
        }

        If ( déjà vu file doesn't exit)
                Add déjà vu file and déjà vu peer.
                Return 1;

       }
```

Figure 3-2 Algorithm of adding a Déjà vu

The existence of Déjà vu file is first determined when adding a Déjà vu. If the Déjà vu file already exists, we only have to append the Déjà vu peer on the list. If the Déjà vu file doesn't exist, then we have to make a new entry for Déjà vu file and store its peer list.

Table 3-2 is an example of what a Déjà vu list looks like. If a Déjà vu file is 10031.mp3 or 10004.mp3, then the Déjà vu peer is appended to the current list. If a Déjà vu file is 10099.mp3 and it doesn’t exist in Déjà vu file list, then we have to make a new entry for storing its Déjà vu peer. The column “HIT” is for another function which will be explained later.



Déjà vu file	Déjà vu peer	hit
10031.mp3	140.113.132.50	0
	140.113.132.40	0
	140.113.55.1	0
10004.mp3	140.113.44.5	0
	140.113.44.3	0
	140.113.2.111	0
….	….	…

Table 3-2Déjà vu Database

As far as scalability is concerned, each peer only keeps certain amount of Déjà vu. Déjà vu are added and removed from the list based on its utility. In Chapter 3.3.3, we will introduce the ranking algorithm for selecting Déjà vu.

3.2.2 Instant Search

With Déjà vu, we have a function that with a single query and feedback, it stands for that a flooding task has been done. We call it instant search. After Déjà vu discovery, if P also wants to find A, it recalls that A has been searched by P1. Instead of performing flooding to locate A, P will first query on P1, as in Figure 3-3. If P1 have gotten the resources, then P can retrieve A from P1 accordingly. With a simple query on P1, it represents the search performed via flooding by P1. If there are many Déjà vu peers as in Figure 3-4, this function can be very powerful.

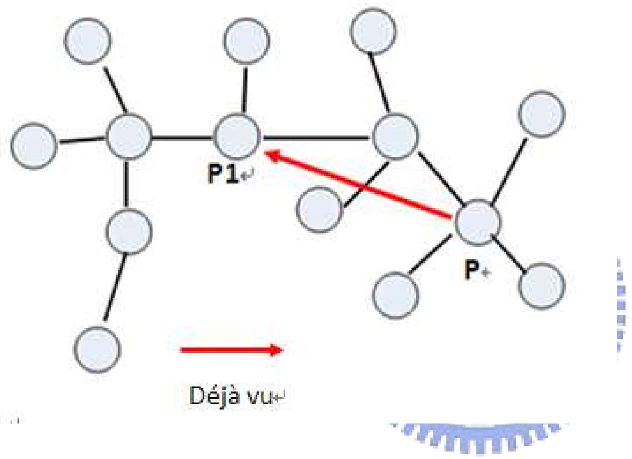


Figure 3-3 P Has Déjà vu on P1

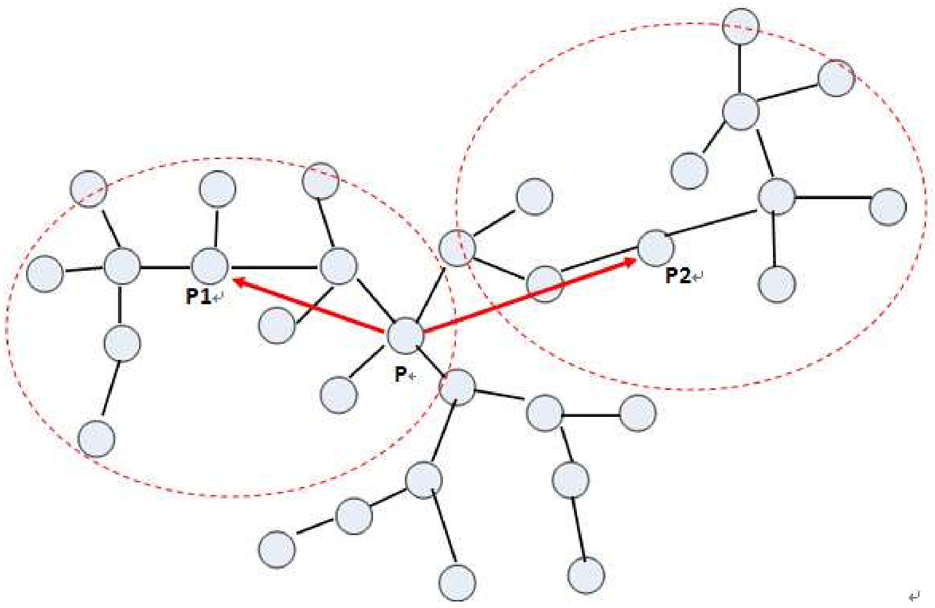


Figure 3-4 Instant search

In figure3-6, if we want to locate 10004.mp3, we will query peer-140.113.44.5, peer-140.113.44.3, and peer-140.113.2.111 first. If we find 10004.mp3 at peer-140.113.2.111, then the hit column will modified to 1.

Déjà vu file	Déjà vu peer	hit
10031.mp3	140.113.132.50	0
	140.113.132.40	0
	140.113.55.1	0
10004.mp3	140.113.44.5	0
	140.113.44.3	0
	140.113.2.111	1
....	....	...

Table 3-3 Find Resource by Déjà vu

3.2.3 Synergy

From the scenario mentioned, P benefits from P1 and P2 via Déjà vu to perform instant search. We present another function that P can contribute back to P1 and P2. We call this mutually beneficial strategy as synergy.

In figure 3-5, If P finds A in P1 but cannot find A in P2, after P retrieves A from P1, it will actively push A to P2.

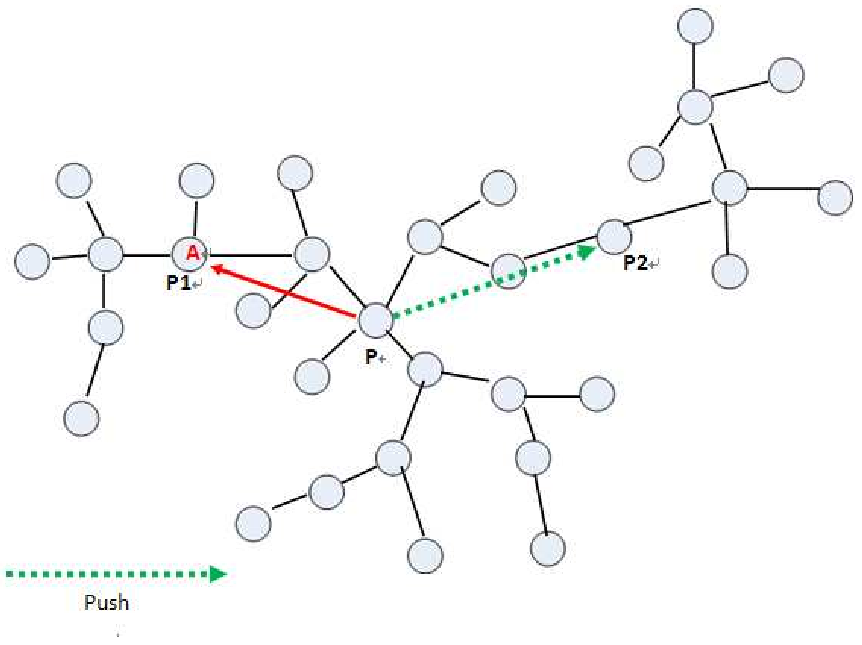


Figure 3-5 P Pushes A to P2

However, if P1 and P2 did not find A, search via Déjà vu fails. Then P would issue a query through flooding. In figure3-6, P sends the query message to its neighbors.

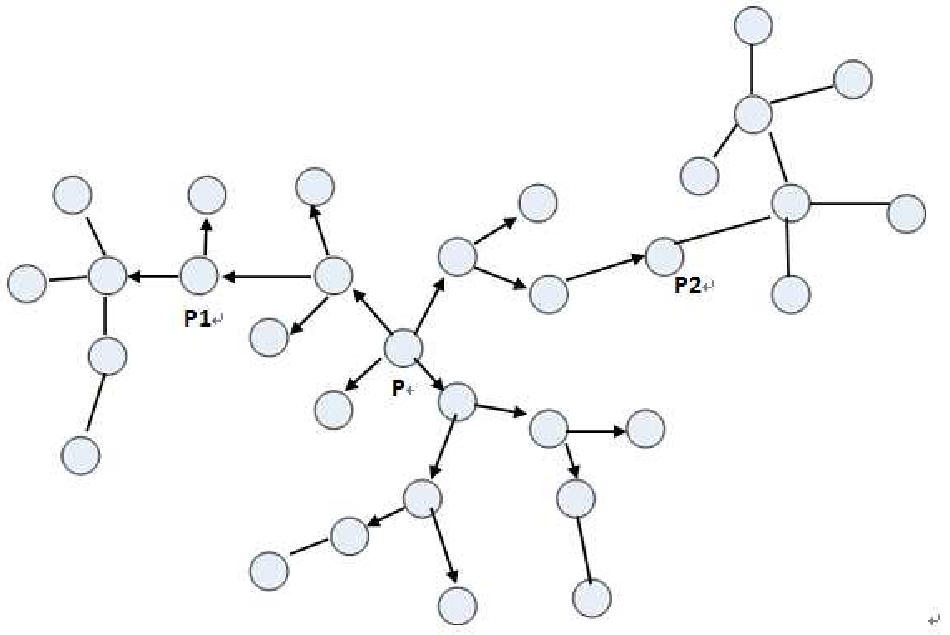


Figure 3-6 P makes a Flooding to Find A

Chances are that A cannot be found at P1 and P2, but found via flooding in P3. In figure 3-7, after P retrieves A, it will push A to P1 and P2, who also wanted the resource yet couldn’t find it. In this way, the resources can be distributed on the network through needs instead of random replication, path replication, or one hop replications.

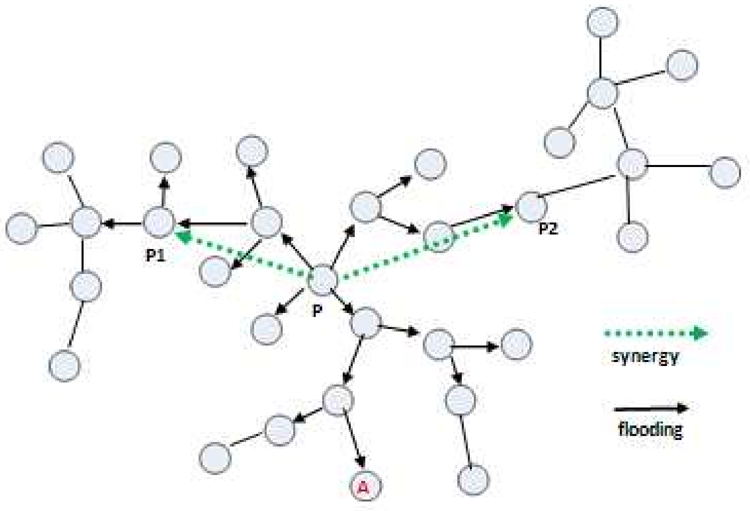


Figure 3-7 P Pushes A to P1 and P2

In Table 3-4 , if we find 10004.mp3 at peer-140.113.2.111, but don't find 10004.mp3 at peer-140.113.44.5 and peer-140.113.44.3, then 10004.mp3 is pushed to peer-140.113.44.5 and peer-140.113.44.3.

Déjà vu file	Déjà vu peer	hit
10031.mp3	140.113.132.50	0
	140.113.132.40	0
	140.113.55.1	0
10004.mp3	140.113.44.5	0
	140.113.44.3	0
	140.113.2.111	1
….	….	…

Table 3-4 An Example of Push

With Déjà vu, we can have the information about who is in need, and push resources to them.

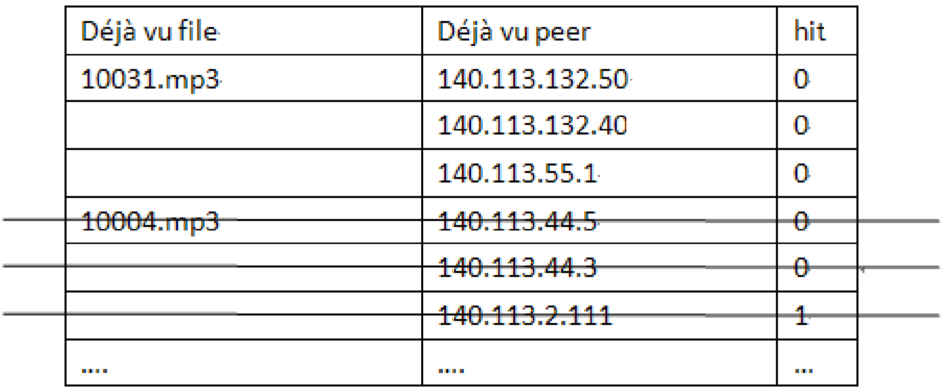
3.3 Déjà vu Selection

As far as scalability is concerned, each peer only keeps certain amount of Déjà vu. Déjà vu are added and removed from the list based on its utility. We now introduce the ranking algorithm to determine which Déjà vu is kept and which is faded away when the list is full.

The Least Recently Used (LRU) policy is the simplest. After receiving a query message, a peer first determines if the Déjà vu of file is new or not. If it is new, it adds in front of the list. If Déjà vu of file is not new, it is appended to the peer's list, and placed it in front of the list. The scheme guarantees that popular files and new files can stay longer, while Déjà vu of the rarely requested files are discarded.

Besides The Least Recently Used (LRU) policy, we also periodically removed the Déjà vu that has facilitated peer acquire the resources. In Table 3-5, If P wants to locate 10004.mp3, and it uses Déjà vu to find 10004.mp3 at peer-140.113.2.111, and push 10004.mp3 to peer-140.113.44.5 and peer-140.113.44.3.It means that it doesn't need

Déjà vu of 10004.mp3 anymore. The 10004.mp3 is at local.



Déjà vu file	Déjà vu peer	hit
10031.mp3	140.113.132.50	0
	140.113.132.40	0
	140.113.55.1	0
~~10004.mp3~~	~~140.113.44.5~~	~~0~~
	~~140.113.44.3~~	~~0~~
	~~140.113.2.111~~	~~1~~
....	....	...

Table 3-5 Remove a Déjà vu

3.4 Theoretical Improvement

Our design theoretically improves the hit ratio, and reduces message overhead and latency. We give some demonstrations to support our presumption.

Message overhead and Latency

In Figure 3-8, if P uses Déjà vu to locate resource either P1 or P2, then P doesn't have to perform flooding which brings about enormous traffic. Consequently, the message overhead is expected to be reduced. In addition, it only takes 1 hop to locate resource, while flooding may need several hops.

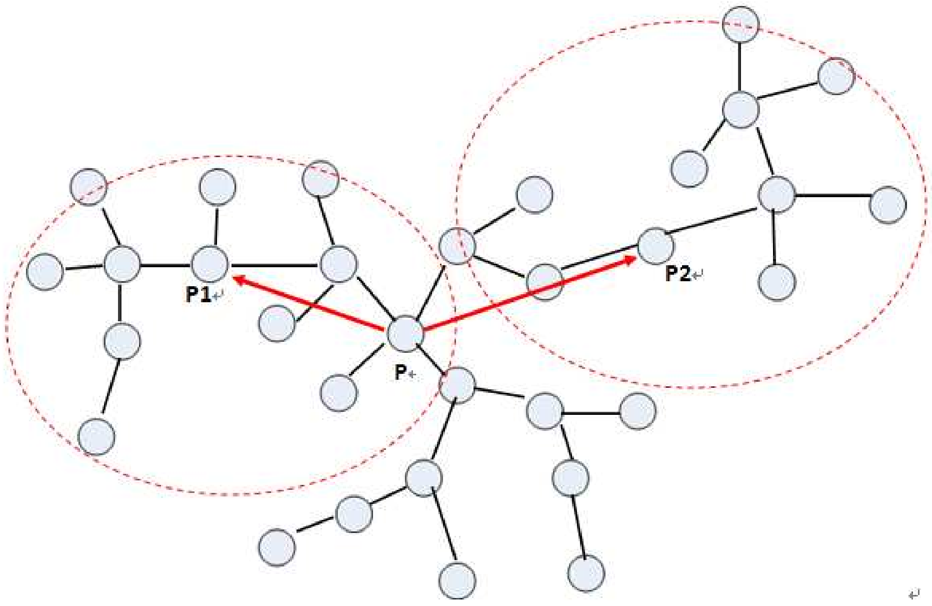


Figure 3-8 P uses Déjà vu

Hit ratio

Synergy pushes resource to those in need, accelerates the distribution of resource, and increases hit ratio. We give an example of how synergy can improve hit ratio.

We assume that there exist such a search sequence, with TTL = 2:

1. In Figure 3-9 , P2 and P3 search resource A. P2 finds A at P4 while P3 fails to find. Since P1 is on the search path, P1 has Déjà vu on P2 and P3 about A.

2. In Figure 3-10, later when P1 wants to locate A, it has Déjà vu and asks P2 and P3.

3. In Figure 3-11, P1 locates A via Déjà vu from P2, and pushes A to

P3, since P3 wants A yet could not find A.

4.In Figure 3-12 , P5 wants to locate A and find it at P3. Without synergy in Figure 3-13, A isn't pushed to P3 thus P5's search may fail.

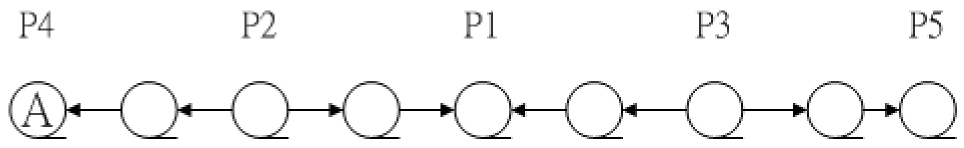


Figure 3-9 P2, P3 search for A

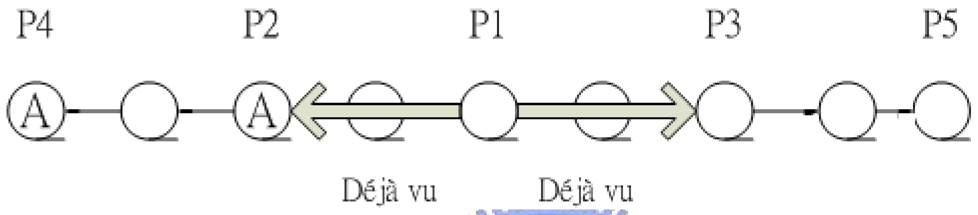


Figure 3-10 P1 has Déjà vu on P2, P3

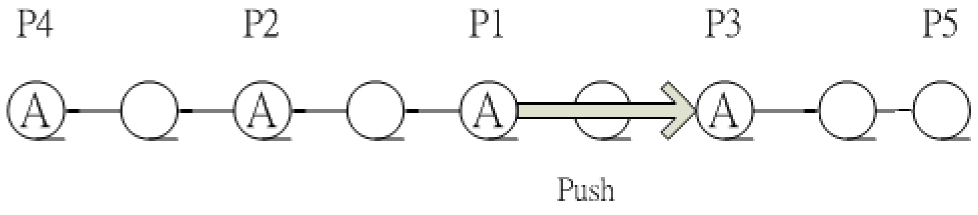


Figure 3-11 P1 pushes A to P3

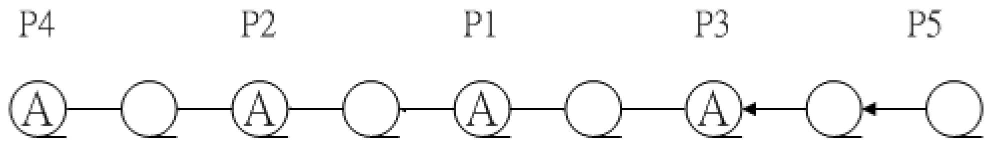


Figure 3-12 P5 locates A at P3

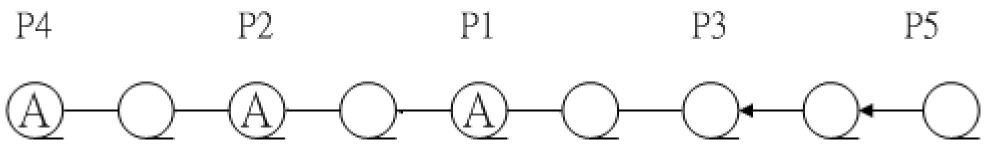


Figure 3-13　Without synergy, P5 fails to locate A.

This is only a simple demonstration. There are different combination and chances that synergy can improve hit ratio.

It is worth noticing that in our example, if P3 searches for A between sequence 3 and sequence 4, then P5's search on A is also success. We find that sequence is an important factor to hit ratio. For flooding, resource MAY be propagated according to probability, while for synergy, resource is SURE to be replicated meaningfully. Synergy accelerates the speed of resource dissemination.

Chapter 4

Simulation and Performance Evaluation

In this chapter, we will describe how the P2P network is simulated and its performance evaluation versus Gnutella and Shortcut.

4.1 System Settings

We use a PC to run the simulation. The CPU is Core Duo 1.6GHz, the memory is 2G, and the operating system is Windows XP professional. We use JAVA running threads to represent different peers in the system, and open sockets to transfer messages. The simulation starts from 2 peers and gradually increments to 140 peers.

4.2 Methodology

Parameters

The followings are some important parameters for the simulation:

1. Resource:

   Resource range: How many kinds of resource (from 10000.mp3 to 10200.mp3)

   Resource Holding Number per peer: The storage capacity of a peer (max=40)

2. Query:

   Query times: How many queries a peer issues (max = 30)

   Query frequency: How often a peer queries (10 seconds)

   Query range: The scope of the query (from 10000.mp3 t0 10200.mp3)

3. Neighbor:

   Neighbor number: How many neighbor a peer has (max=15)

4. Déjà vu:

   Déjà vu file: The number of Déjà vu files (max=200)

   Déjà vu peer: The number of Déjà vu peers (max=10)

Search

Each round each peer in the system will make a search, there are 4 steps in the search.

Figure 4-1 is the flowchart of search.

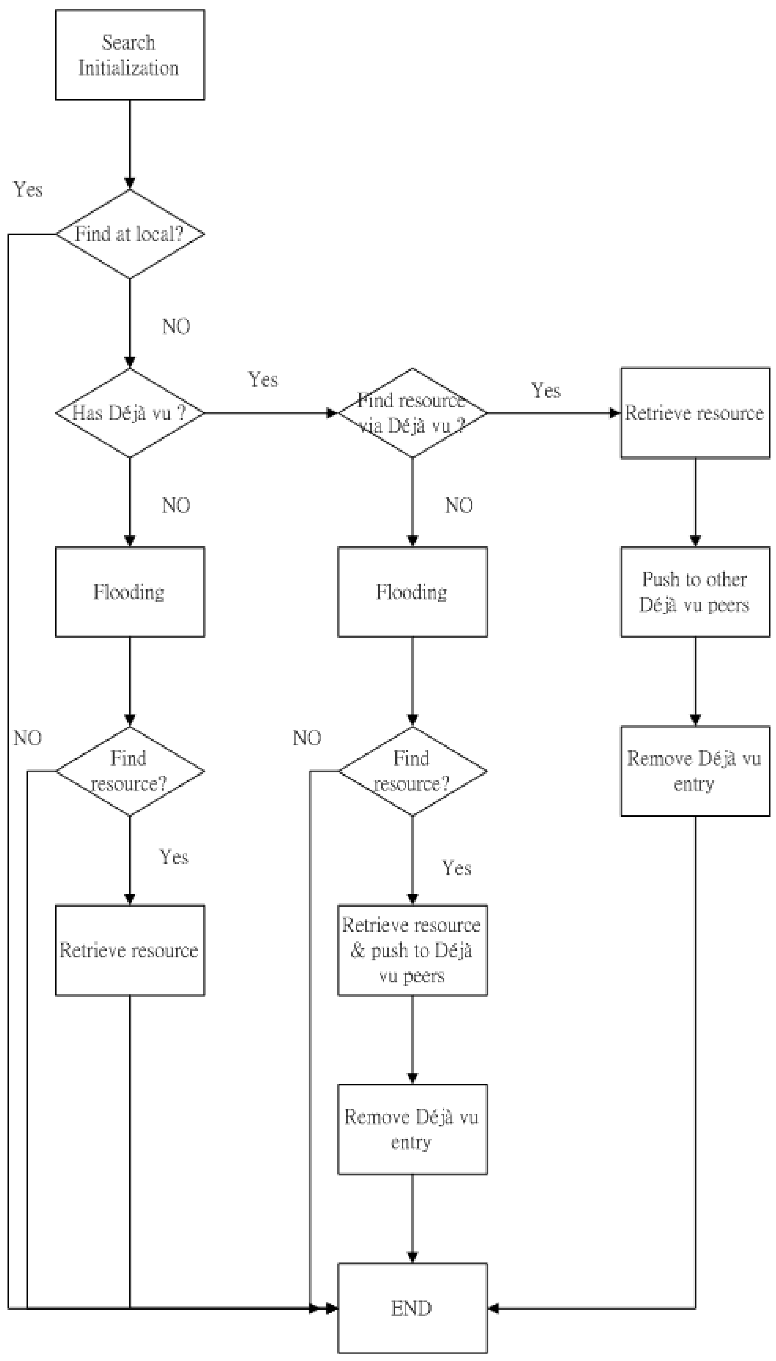


Figure 4-1 Search flowchart

Step1. It checks if local has the resource. If yes, it finds the resource successfully.

Step2. If there is no resource at local, it looks up in the Déjà vu index, to see if other peers have searched for the resource. If it has Déjà vu about the search, it sends out query message to the Déjà vu peers.

DEJAVU; Session ID; Query file; Query Initiator

The major different between the DEJAVU queries and queries of flooding is that, DEJAVU queries only responds “success” or “fail”. For queries of flooding, if the search is fail, it will be forwarded to all its neighbors, if the Time-to-Live (TTL) value is greater than 0.

Step3. If some Déjà vu peers responds “success” and some “fail”, after the peer retrieves the resource, it will push the resource to the Déjà vu peers responding with “fail”. Then, it removes the Déjà vu from its Déjà vu list.

Step4. If the peer cannot locate resource via Déjà vu, it resorts to flooding. The flooding message will be send to all its neighbors.

QUERY; Session ID; Query file; Query initiator; From; TTL; Latency

Step5. If the peer cannot locate resource via flooding, the search is failed. If the peer successfully find the resource, after retrieving the resource, it will push the resource to the Déjà vu peers, who wanted the resource but couldn’t find it. Finally, it removes the Déjà vu entry.

Receive

Figure 4-2 is the flowchart of receiving message.

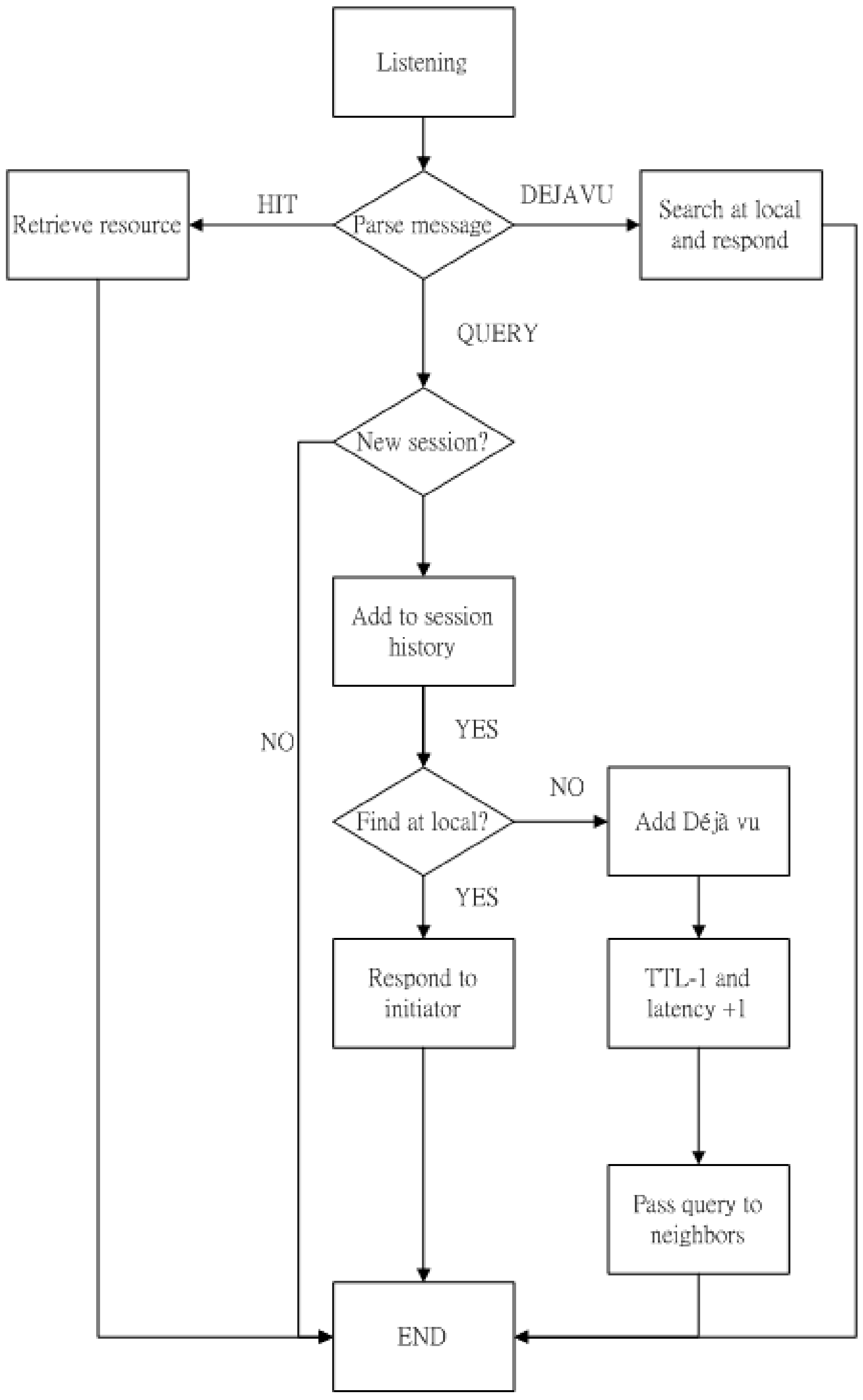


Figure 4-2 Receive message flowchart

Step1. Upon receiving message, the peer first parses the message and categorizes it and performs different task. There are three types of messages : HIT, QUERY, and DEJAVU. “HIT” indicates a previous search or Déjà vu request has been responded with success. “DEJAVU” is under the circumstance that other peers have Déjà vu on it. “QUERY” is the flooding message.

Step 2. Check if this session already exists in the session history. If yes, it means that this query message have visited here before, by multipath from different peers. Then the peer will perform nothing to this message.

Step3. If it is a new session, the peer will append the session to session history, and search the requested resource at local.

Step4. If success, it sends a HIT message to the query initiator with current latency. Type is used to determine whether HIT is for Déjà vu search or flooding search.

HIT; Session ID; Query responser; Latency; Type

Step5. If there is no requested resource at local, the peer adds it to Déjà vu list. Then it decrease TTL by 1 and increase latency by 1, and floods the query messages to all its neighbors except the query initiator and where the query previous from.

QUERY;Session ID;Query file;Query initiator; From; TTL-1;Latency+1

4.3 Performance Evaluation

There are some metrics to evaluate the performance : Message overhead, Average latency, and Hit ratio. The following is the definition and the formula to evaluate the performance.

1. Message overhead = (Flooding message +Déjà vu message)

   Message overhead: Total overhead of the system.

   Flooding message: The message generated by flooding.

   Déjà vu message: The message generated by Déjà vu.

2. Average latency = Total latency ÷ Total hit

   Average latency: Average delay time of all the successful hits.

   Total latency: Sum of all the delay of successful hits.

   Total hit: Sum of all the successfully hit.

3. Hit ratio = Total hit ÷ Total query

   Total hit = Flooding hit +Déjà vu hit

   Hit ratio: The chance a resource can be located.

   Total hit: Sum of all the successful hit.

Total query: Sum of all the queries.

Flooding hit: Find resource by flooding

Déjà vu hit: Find resource by Déjà vu

Message Overhead

We implement the Gnutella, shortcut with their algorithms, and use the same parameters as in Déjà vu .

The lower the message overhead, the better the performance is. In figure 4-3, we compare the message overhead with Gnutella and Shortcut. We can find that the message overhead is greatly reduced with Déjà vu approach.

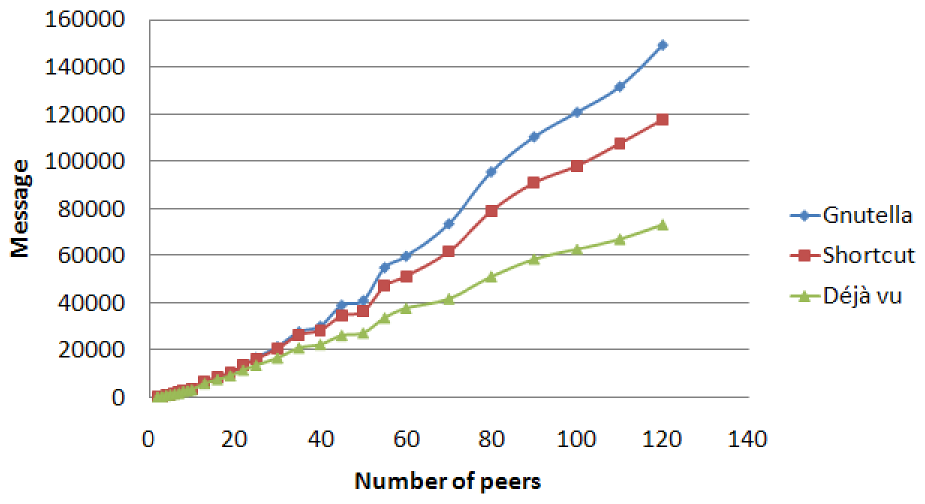


Figure 4-3 Message overhead

In figure 4-4, it is the message percentage compared with Gnutella and Shortcut. When the number of peer is low, the message overhead of shortcut and Déjà vu is greater than that of Gnutella. It is expectable. The reason is that when the peer number is low, the overhead of flooding of Gnutella is also small. However, if there is an enhancement as shortcut and Déjà vu, the first query of shortcut and Déjà vu also incurs overhead. When the peer number grows, the flooding overhead gradually becomes tremendous. In this case, if the first enhancement search is successful, a peer doesn't have to perform flooding, thus the overhead is reduced. When the number of peer grows over 100, Déjà vu approach can save 40% of message overhead.

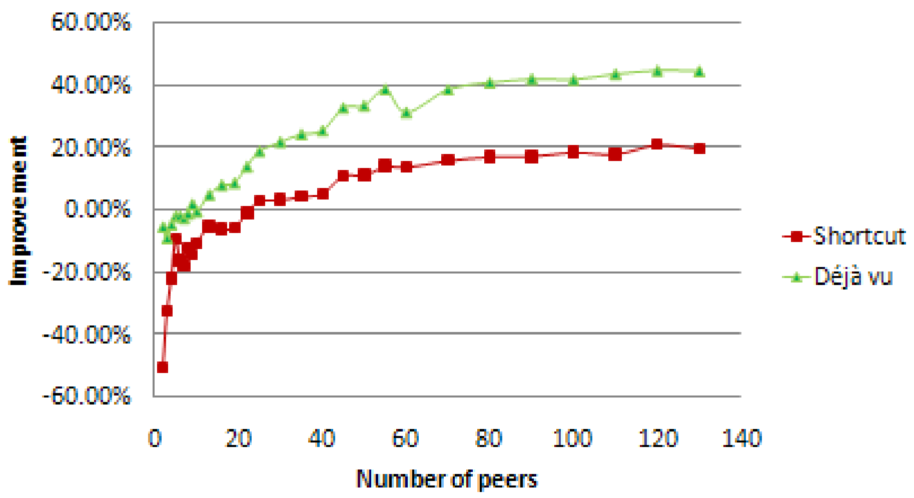


Figure 4-4 Message overhead improvement

Latency

Latency represents how long does it takes for a successful hit to reply. The first responded successful hit is taken as latency. Here the latency is the hop number. Lower latency indicates the resource is located faster. Figure 4-5 is the comparison of Gnutella, Shortcut and Déjà vu. Figure 4-6 is the latency percentage compared with Gnutella, Shortcut, and Déjà vu. Déjà vu approach can be 20% faster to locate resource than Gnutella.

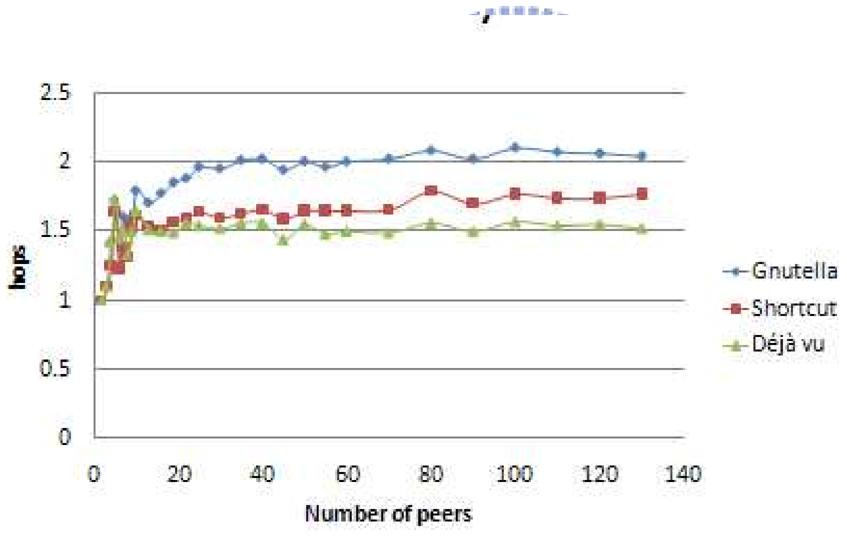


Figure 4-5 Latency

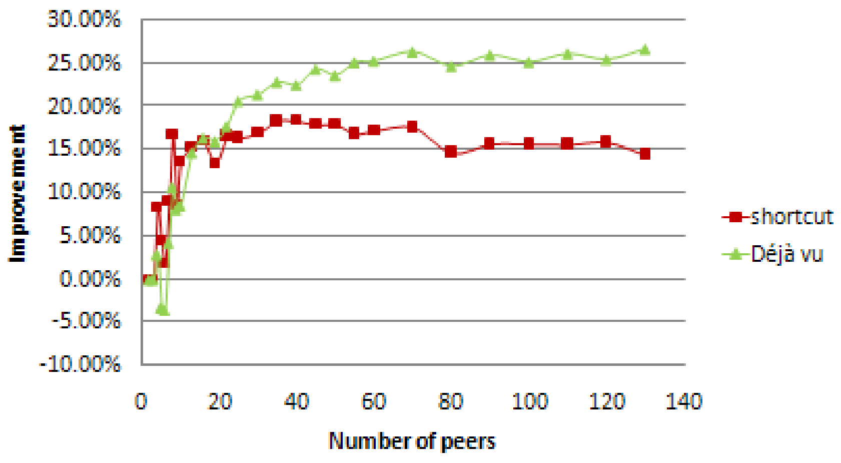


Figure 4-6 Latency improvement

Hit Ratio

The hit ratio refers to the chance that a resource is expected to be found. Figure 4-7 is the hit ratio comparison between Gnutella, Shortcut, and Déjà vu. Figure 4-8 is the improvement of Déjà vu approach.

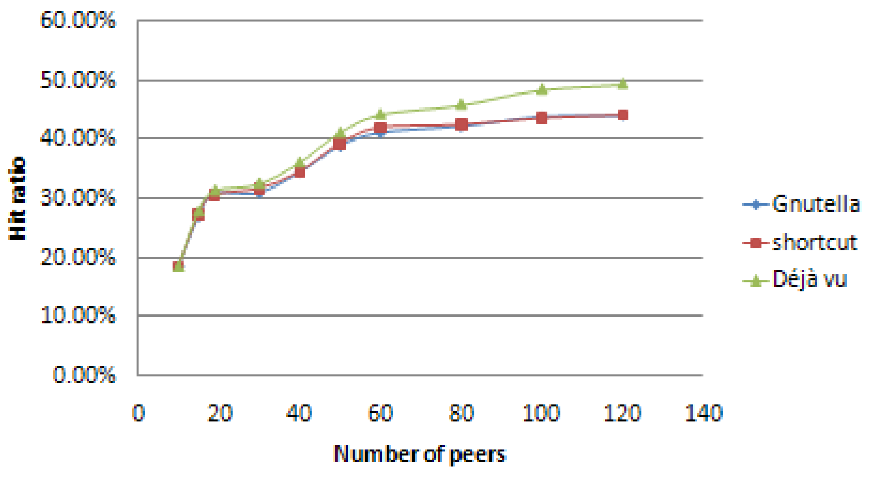


Figure 4-7 Hit Ratio

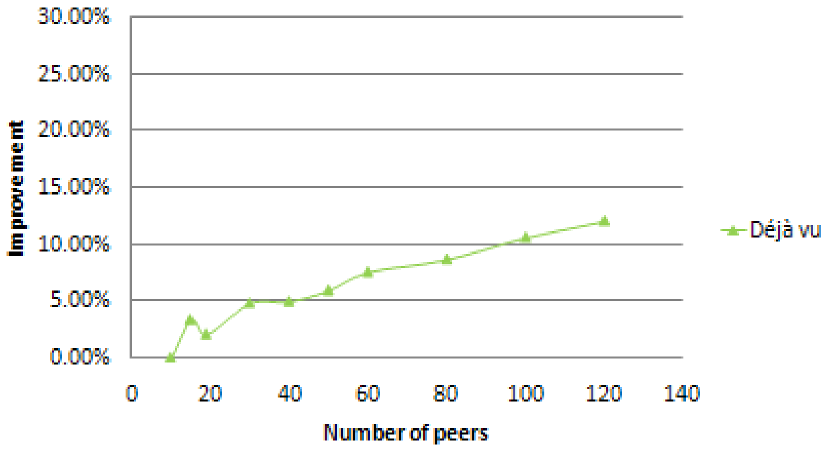


Figure 4-8 Hit Ratio Improvement

Simulation shows that the Gnutella and shortcut have the same hit ratio. The reason is obvious since shortcut is obtained from flooding, while Gnutella's search scope is also flooding. Déjà vu approach has higher hit ratio compared with Gnutella and Shortcut. The benefit comes from synergy.

Chapter 5 Conclusion

In this thesis, we present Déjà vu approach to improve search efficiency. Through our design philosophies, we use message as a tool to capture interests, turning stone into gold. What's more, we try to inverse the process of search, to actively push the resources to those in need. We use synergy to achieve the mutually beneficial strategy, and the speed of resource dissemination is accelerated accordingly.

The simulation results confirm that Déjà vu approach can reduce message overheads, shorten resource retrieval time, and increase hit ratio.

Reference

[1]Gnutella, http://en.wikipedia.org/wiki/Gnutella

[2]Ajay chander, Steven Dawson, etc., “NEVRLATE: Scalable Resource Discovery”, CCGRID, 2002

[3]Q.LV, P.Cao, E.Coehn, K.Li, S.Shenker, “Search and Replication in Unstructured Peer-to-Peer Networks”, International Conference on Supercomputing Proceedings of the 16th international conference on Supercomputing,2002

[4]I. Clarke, O.Sandberg, etc., “Freenet: A distributed anonymous information storage and retrieval system in designing privacy enhancing technologies”. LNCS, 2001

[5]Y.Chawathe, s. Ratnasamy, L. Breslau,etc. ,”Making Gnutella-like P2P systems scalable”. SIGCOMM, 2003

[6]Napster, http://www.napster.com

[7]KaZaA, http://www.kazaa.com

[8]Skype, http://www.skype.com

[9] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan,“Chord: A Scalable Peer-to-Peer Lookup Service for Internet Application.”

SIGCOMM 2001.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker ," A Scalable Content-Addressable Network." SIGCOMM 2001

[11] A. Rowstron, P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," IFIP/ACM, 2001

[12] S. Ratnasamy, S. Shenker, and I. Stoica, .Routing algorithms for DHTs: Some open questions,. in Proc. of International Peer-To-Peer Workshop,2002

[13] H. Jin, X. Ning, and H. Chen, "Efficient Search for Peer-to-Peer Information Retrieval Using Semantic Small World," Proc. Int'l Conf. World Wide Web ,2006.

[14] A. Lo¨ ser, S. Staab, and C. Tempich, "Semantic Social Overlay Networks," IEEE J. Selected Areas in Comm.,2007.

[15] M. Li, W. Lee, and A. Sivasubramaniam, "Semantic Small World:An Overlay Network for Peer-to-Peer Search," ICNP, 2004.

[16] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in Proceedings of INFOCOM, 2003.

[17] V. Cholvi, P.A. Felber, and E.W. Biersack, “Efficient Search in nstructured Peer-to-Peer Networks,” European Trans. Telecomm., 2004.

[18] W.T.Chen, C.H.Chao, and J.L.Chiang, “An Interested-based Architecture for Peer-to-Peer Network Systems”, Proceedings of the 20th International Conference on Advanced Information Networking and Applications, 2006