

國立交通大學

資訊科學與工程研究所

碩士論文

支援語義搜尋之自組式同儕網路



A Self-Organizing P2P Network for Semantic Search

研究生：鄭仁傑

指導教授：陳俊穎 教授

中華民國九十八年八月

支援語義搜尋之自組式同儕網路

A Self-Organizing P2P Network for Semantic Search

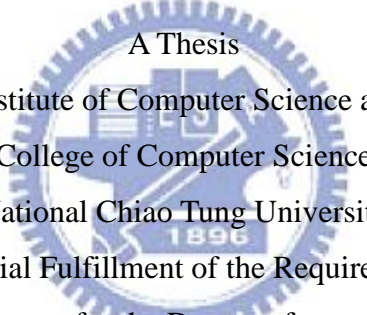
研究生：鄭仁傑

Student：Jen-Chieh Cheng

指導教授：陳俊穎

Advisor：Jing-Ying Chen

國立交通大學
資訊科學與工程研究所
碩士論文

The logo of National Chiao Tung University is a circular emblem with a gear-like border. Inside the circle, there is a stylized building and the year '1896'. The text 'A Thesis' is positioned above the logo, and 'Submitted to Institute of Computer Science and Engineering' is written across it.

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

August 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年八月


支援語義搜尋之自組式同儕網路

研究生：鄭仁傑

指導教授：陳俊穎 博士

國立交通大學
資訊科學與工程研究所
碩士論文

摘要



在最近幾年裡，同儕網路已成為網路上檔案管理與分享平台的重要技術之一。而隨著網路資訊的急遽累積，如何在網路上快速有效的搜尋相關文件以符合使用者的要求變的更加重要。然而，在同儕網路中，資訊是由大量的電腦主機共同管理，如何以關鍵字搜尋文件且避免浪費過多電腦資源已是一個很大的挑戰，要進一步搜尋語義相關的文件將更為困難。針對此問題，我們設計一種自組式同儕網路，讓網路中每個節點觀察所有經過的信息，不斷學習與替換所管理的檔案以及其他同儕的資訊，藉此讓自己所管理的檔案能逐漸偏向某一類相關的資料。除此之外，整個同儕網路也能逐漸演進並具有 small world 的特性，使搜尋更加快速準確。有別於其他以 DHT 為基礎的同儕網路像是 pSearch 及 SSW，我們的方法是採用 Freenet 的概念讓網路能自我組織。我們進行了大量的實驗，並且與 pSearch 做比較。實驗結果顯示我們的系統能有效的自我的組織，讓搜尋能以更快的速度得到比 pSearch 更精確的資料。

關鍵字：同儕網路、語義搜尋、自組式

A Self-Organizing P2P Network for Semantic Search

Student: Jen-Chieh Cheng

Advisor: Dr. Jing-Ying Chen

Institute of Computer Science and Engineering

National Chiao Tung University

1001 Ta Hsueh Road, Hsinchu, Taiwan, ROC

Abstract

P2P network has become a popular platform for distributed file sharing and management in recent years. As the information accumulated in the network growing exponentially, targeted content-based search is becoming a necessary feature for P2P networks to be viable. However, because information in a P2P system is usually dispersed among a large number of peers, providing efficient keyword-based search is already challenging, not to mention supporting semantic search which allows users to query information using high-level concepts. To address this challenge, we propose a self-organizing P2P network, by letting each peer continuously observe query and response information passed by and update information about related documents and other peers gradually. Because each peer also keeps several references to other peers that hold completely unrelated documents, the resulting network can gradually evolve into a small-world network where a query can be routed to the set of related peers quickly following very simple greedy strategy. Our method adopts some of the key principles from Freenet, and is architecturally different from other DHT-based P2P networks such as pSearch and SSW. We conduct extensive simulation and compare our approach with pSearch. The simulation results show that our network keeps improving itself, and can achieve better search quality even with a smaller number of search steps when compared to pSearch.

Keywords: peer-to-peer, semantic search, self-organization, small world

誌謝

對於碩士論文的完成，首先要感謝我的指導教授陳俊穎老師，在這兩年研究生涯中，不斷地給予我詳盡的指導與適時的鼓勵，讓我在解決問題與研究方法上得到許多豐富的經驗和技巧。同時也感謝口試委員江清泉教授、陳健教授以及易志偉教授，提供許多寶貴的意見，使得我的論文內容能夠更加完備和充實。

其次，感謝實驗室共同奮鬥的學長以及同學，在研究生涯裡給我許多的支持與鼓勵，並且當我遇到困難時，也提供了需多意見，讓我能順利的完成碩士論文。

最後，由衷地感謝我的家人，由於他們全力的支持，提供一個無後顧之憂的學習環境，使我能順利的完成碩士學位，我願將這份榮耀獻給我的家人



鄭仁傑 謹誌 2009 年 8 月

於交通大學協同合作實驗室

Table of Contents

摘要	i
Abstract	ii
誌謝	iii
Table of Contents	iv
List of Tables	vi
List of Figures	vii
List of Algorithm	viii
Chapter 1. Introduction	1
Chapter 2. Background and Related Work	5
2.1. Key-based Search.....	5
2.1.1. Napster and Gnutella.....	5
2.1.2. Freenet.....	6
2.1.3. Small-World Networks.....	7
2.1.4. Content addressable Network (CAN).....	8
2.2. Semantic-based Search.....	9
2.2.1. Semantic Vector.....	9
2.2.2. pSearch.....	10
2.2.3. Semantic Small World.....	11
Chapter 3. Self-Organizing P2P Network Architecture.....	14
3.1. Overview.....	14
3.2. Overlay Construction.....	15
3.3. Semantic Search.....	16
3.4. Document Cache Swapping.....	22
Chapter 4. Peer Movement.....	24

Chapter 5. Experiments 27

 5.1. Performance Result for 2-Dimensional Synthetic Data..... 28

 5.2. Performance Evaluation..... 31

 5.2.1. The Effect of Data Distribution..... 31

 5.2.2. The Effect of Flocking..... 32

 5.2.3. Search Performance..... 33

Chapter 6. Discussion and Future Work 35

Chapter 7. Conclusion 36

Reference 37



List of Tables

Table 5.1: The detailed parameters used in simulation: 27



List of Figures

Fig 2-1: Locating files in Napster and Gnutella.....	5
Fig 2-2: An illustrative example of Freenet's routing table.....	6
Fig 2-3: An illustrative example of small world.....	7
Fig 2-4: A 2-dimension CAN.....	8
Fig 2-4: A rolling index of pSearch.....	10
Fig 2-6: An illustrative example for SSW.....	11
Fig 2-7: An illustrative example of SSW-1D.....	12
Fig 3-1: A key idea.....	14
Fig 3-2: Peer positioning using mediums.....	16
Fig 3-3: The flow chart of request procedure.....	17
Fig 3-4: The flow chart of response procedure.....	21
Fig 4-1: Three basic flocking rules.....	25
Fig 5-1: The distribution of peers and documents.....	28
Fig 5-2: Distribution of documents peer managed.....	29
Fig 5-3: The change of search path length.....	30
Fig 5-4: The effect of flocking.....	30
Fig 5-5: The effect of data distribution.....	31
Fig 5-6: The effect of flocking.....	32
Fig 5-7: Document overlap.....	32
Fig 5-8: Result quality.....	33
Fig 5-9: Recall per hop.....	34

List of Algorithm

Algorithm 3-1: Algorithm for RDP	18
Algorithm 3-2: Algorithm for RRP.....	19
Algorithm 3-3: Algorithm for search procedure.....	20
Algorithm 3-4: Algorithm for response procedure.....	21



Chapter 1. Introduction

With the development of network technologies progressing vigorously and the use of Internet applications becoming more popular during the past few years, information is accumulating in an alarming speed. Consequently, how to sieve through this vast amount of information residing in the network and make the best sense out of it efficiently becomes important and valuable for both individuals and companies. However, the technical challenges are becoming more difficult to deal with not only because the volume of information is extremely large, but also the semantic relations embedded within them cannot be analyzed and queried timely.

Recently, peer-to-peer (P2P) architecture has shown its strength in handling large volume of data in a fully decentralized, scalable, and robust way. A typical P2P system consists of a dynamic network of machines each of which can serve as a client and a server at the same time. When a user connects to and interacts with a P2P system, his/her machine often becomes part of the network in which member machines communicate with each other in a prescribed protocol. Typical P2P systems in active use today enable users to contribute and share some kind of information, such as multimedia files.

Although P2P networks can effectively manage large volume of data, how to search information efficiently and meaningfully in such networks is still an active research topic pursued by many researchers. To facilitate document search, earlier P2P applications either use centralized servers to maintain document indexes, or rely on query flooding to probe the participating machines for documents requested by the user. The former is not fully decentralized and exhibits a single point of failure, while the latter is not scalable because of the amount of messages generated for each query, especially when the network grows larger.

To improve the search performance while preserving the desirable characteristics of P2P architecture, many modern P2P networks employ the so-called distributed hash tables (DHTs). Notable examples include CAN [Ratnasamy, et al], Chord [Stoica, et al], Pastry [Rowstron, et al], and Tapestry [Zhao, et al]. In these approaches, documents are first mapped to certain hashed keys, and some network overlays are established on top of which keys can be searched efficiently.

DHT-based P2P networks are promising alternatives when what one wants is a simple document storage system, where the user can recall a document if he/she knows the document name or even the hashed key. These systems fall short when content-based search is concerned, that is, when the user wants to search for documents that contain content satisfying some criteria. This is because the key-hashing step has removed the content information. To add content-based search capability, additional mechanisms are needed on top of the DHT-based overlay.

Many approaches have been proposed to support content-based or semantic-based search in P2P architecture. For example, [Shen, et al] proposes hybrid architecture in which peers are divided into super nodes and ordinary ones, and the super nodes maintain index information. In contrast, other researchers adapt existing DHTs approach, but replace the key hashing with *semantics-preserving* mapping. Notable examples include pSearch [Tang, et al] and SSW [Li, et al], which make use of Vector Space Model [Berry, et al] in conjunction with Latent Semantic Indexing [Deerwester, et al] that can effectively map a set of documents into points in a semantic space. Because the closeness between two documents in the semantic space also suggests the semantic similarity between them, semantic-based search is supported simply by allowing the user to query documents that fall in a certain range in the semantic space. This approach is conceptually straightforward, and can be implemented easily when the dimension of the key space is small. When the dimension is high, the incurred

maintenance overhead become large, and additional engineering effort is needed to cope with the high dimensionality, therefore making the network more complicated to implement. Furthermore, these DHT-based approaches often proceed by successively dividing the semantic space in a top-down manner, driven by the sequence of joining or leaving events of peers. This partition strategy may not respect the natural clustering of documents, even when the document distribution is considered when partitioning the semantic space.

In this paper, we investigate the use of the same semantic-preserving document-key mapping in *unstructured* P2P architecture to support efficient content-based search. Specifically, our P2P architecture resembles Freenet [Clarke, et al.] with many design principles drawn from it. In Freenet, documents are also hashed not only to permit anonymity but also to preserve P2P characteristics. A greedy search heuristic is used to guide search; that is, when receiving a query either locally from the user or relayed from another peer, each machine either responds to the query if it has the corresponding document, or redirects the request to peers that to its knowledge has best chance to host the document, based on the closeness of the query (key) and the keys maintained in the local cache. Furthermore, each machine also updates its routing table to reflect new knowledge it learned. Such a distributed learning process permits Freenet to evolve gradually into a so-called *small-world network* [Kleinberg] with small network diameter, making the greedy search heuristic highly efficient.

Not surprisingly, the principles behind Freenet fits well with the abovementioned semantic-based search approach using semantics-preserving key mapping. The main issue needs to be addressed is the uneven distribution of the keys because the documents themselves are semantically clustered. No other drastic architectural change is needed even when data dimension is high, as long as the network can still self-organize into some small-world network. To achieve the goal, in our architecture each machine is associated with a point in the semantic space based on the documents it hosts (for example by choosing their

geometric center). Since the set of documents hosted by a machine changes over time, the machine's position also changes accordingly. In our architecture, each machine simply learns the positions of other machines continuously based on the information passing by, as in the original Freenet, and the greedy search heuristic remains the same. To retain small-world network characteristics, the size of the routing table in each peer is limited. Each peer then attempts to keep track of the other peers closer to it in its routing table, possibly by kicking out more distant ones. In addition, the peer also makes room in its routing table to maintain a certain number of distant peers in order to provide "short-cuts" for queries; this is essential to keep the overall network diameter low.

We have conducted extensive simulation using synthetic data and the TREC data set. The simulation results showed that our system can self-organize into a small-world network and provide efficient semantic-based search. Several performance comparisons are also conducted against pSearch; the results show that our system can quickly converge and perform much better even with a smaller number of search steps.

The rest of the paper is organized as follows: In Chapter 2 we will describe the background about semantic search and small work network, and survey related research on semantic-based P2P network. In Chapter 3 we will provide an overview about the architecture of our system. In Chapter 4 we will present an improvement in our system. In Chapter 5 we perform various experiments via simulation and discuss the results. In Chapter 6 we discuss some issues and outline possible future work, and then conclude the thesis in Chapter 7.

Chapter 2. Background and Related Work

2.1. Key-based Search

2.1.1. Napster and Gnutella

Napster [Fanning] is a well-known online music file sharing service which is the first P2P file sharing network employed extensively. Instead of storing the files on a central computer, the files are placed on users' machines. Napster utilizes a central server to maintain the indexes of the files that are currently being shared by connected peers. Each peer maintains a connection to the central server, to which it sends file requests. Upon receiving a file request, the central server then informs the peer where to find the requested file, and it is the peer who initiates file download directly from other peer. An illustrative example is shown in Fig. 2-1 (a). As a centralized network, Napster is unreliable due to the single point of failure, and its scalability is also limited.

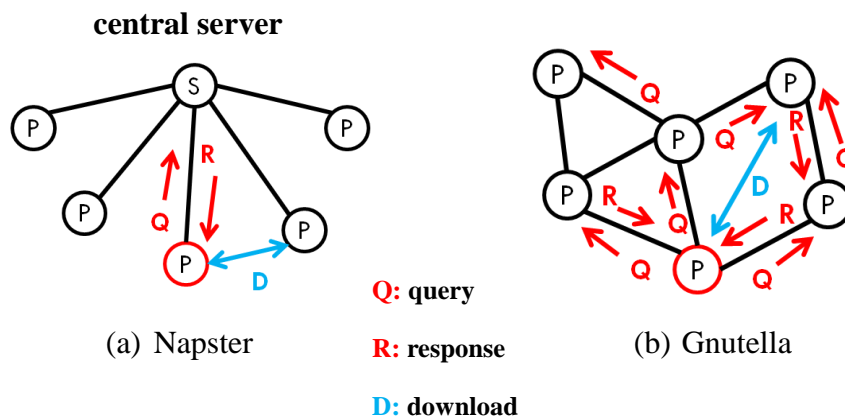


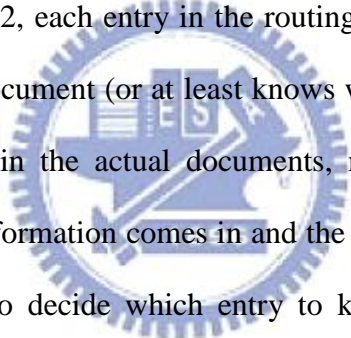
Fig. 2-1. Locating files in Napster and Gnutella. (a) Napster. (b) Gnutella

Different from Napster, Gnutella [Frankel] is a file sharing system employing a fully decentralized architecture. To locate a file, peer initiates a controlled flood by sending query

to all of its neighbors. Upon receiving a query, a peer checks if any locally stored files match the query. If so, the peer sends a query response back towards the query originator through the overlay. If the file is not found, the peer continues to flood the query through the overlay. An illustrative example is shown in Fig. 2-1 (b). Although Gnutella has better reliability and is fault tolerance, it incurs excessive query traffic and is not scalable.

2.1.2. Freenet

Freenet is an unstructured and fully decentralized anonymous P2P network originally designed by Ian Clarke. In Freenet, each file is identified by the binary key by applying a hash function (e.g. SHA-1). A Freenet peer maintains a routing table about documents and their sources. As illustrated in Fig. 2-2, each entry in the routing table contains the document key and the peer that contains the document (or at least knows where the document comes from). Some of the entries also contain the actual documents, meaning the peer is caching the document locally. When new information comes in and the routing table is full, some kind of replacement policy is applied to decide which entry to kick out. A common policy is to replace the entry that has not been accessed for the longest time.



Key	Ref	Data
abc	124	D1
wrw	142	D2
etw	344	D3
fsv	465	D6
sdfg	125	D8
fgf	198	
dhd	153	
zvs	463	
ewt	576	
jeo	244	

Fig. 2-2. An illustrative example of Freenet’s routing table

Users search for documents by giving the document keys explicitly. When searching a document, if the peer receiving the query does not contain the document, it looks up the documents in its routing table and find the one with *closest* keys (e.g. using simple bit-by-bit comparison), and forwards the query to the peer that own or knows more about the document. Observation shows that Freenet indeed evolves into a small-world network gradually, such that the greedy search heuristic mentioned above works quite efficiently.

2.1.3. Small-World Networks

Small-world networks are an important concept and need further description. A small world network [Kleinberg] consists of many clusters. Peers among a cluster have intensive connection, called *short-range link*. In addition, there are also shortcuts, called *long-range links* that connect different clusters. To meet the small-world criteria, the network should have two properties: (1) the intensive connection among a cluster is far higher than between different clusters. (2) Two peers from two different clusters can reach each other in a small number of hops. In this network, a search can be performed in $O(\log^2 N)$ steps, where N is the number of peers in a network. An illustrative example is shown in Fig. 2-3, in which the black links represent short-range links while the red dotted links represent long-range links.

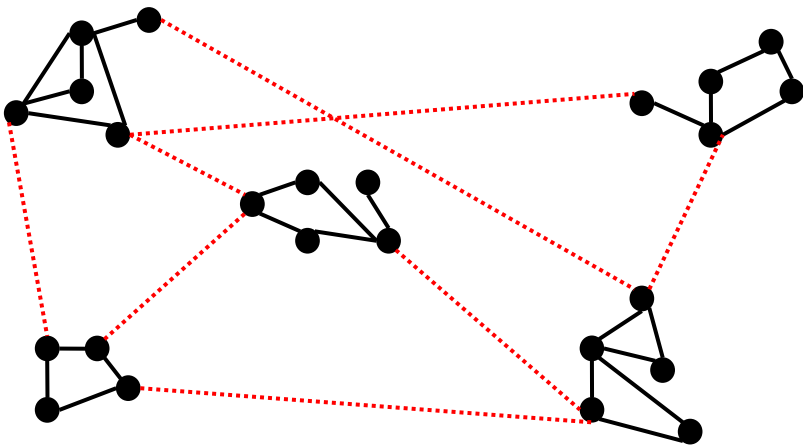


Fig. 2-3. An illustrative example of small world

2.1.4. Content addressable Network (CAN)

CAN is a well-known structured network using the distributed hash table (DHT) to organize the P2P overlay. CAN partitions the k-dimensional Cartesian space into *zones*. Each zone is assigned to a peer. A data object is mapped a point in the Cartesian space using certain hash function (e.g. SHA-1), and is managed by the peer whose zone contains the point. When a peer joins the network, it randomly selects a point in the Cartesian space and routes to the zone that contains the point. Then the current owner of this zone would split the zone into two smaller zones and hands over one of the subzone to the new peer. Two peers are neighbors if their coordinate spans overlap along d-1 dimensions. During search, the peer always routes message to the neighbor whose zone is the closest to a query. For a k-dimensional space partitioned into n equal zones, the average routing path length is thus $(k/4)(n^{1/k})$ and each peer maintain $2k$ neighbors. An example 2-dimension CAN is shown in Fig. 2-4, in which there are six peers A-G in the network. Each peer has a zone in Cartesian space. Peer A has four neighbors B, C, D and E. When peer G request a data object with point (x, y) , it would send request to F, and F forwards the request to E, and finally E forwards the request to A.

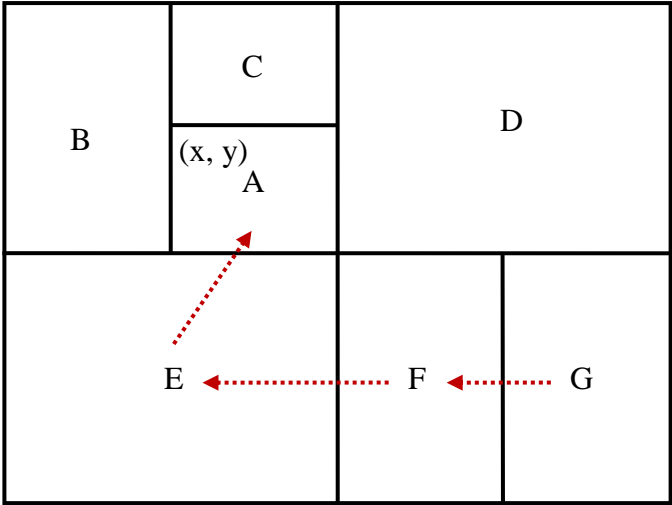
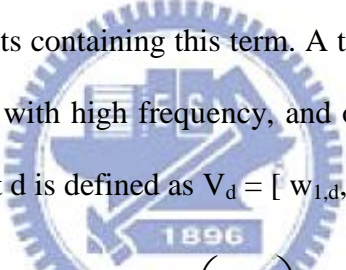


Fig. 2-4. A 2-dimension CAN

2.2. Semantic-based Search

2.2.1. Semantic Vector

Various digital data, including documents, pictures and multimedia, can be presented with k -element vectors. Each element represents a particular feature or attribute. In the information retrieval community, Vector Space Model (VSM) is well-known for document indexing. In VSM, the k -element vectors are called term vectors, and each element corresponds to an important word or phrase in the document or query. The weight of an element is computed using the *term frequency - inverse document frequency* (TF-IDF) scheme. The term frequency is the number of times a term occurs in a document, and the document frequency is number of documents containing this term. A term is importance for a document if a term appears in a document with high frequency, and only appears in a few documents. The semantic vector of document d is defined as $V_d = [w_{1,d}, w_{2,d}, \dots, w_{k,d}]$, where


$$w_{t,d} = tf_t * \log\left(\frac{D}{df_t}\right) \quad (1)$$

However, VSM suffers from synonyms and noise in the documents. Latent Semantic Indexing (LSI) is a well-known approach that overcomes this problem by using the singular value decomposition (SVD) to identify patterns in the relationships between the terms. More importantly, LSI transforms the high-dimension space of term vectors into a lower-dimension *concept* space, such that documents that are semantically closer tend to be close to each other in the concept space after the transformation. Many P2P networks use this feature to support semantic search.

2.2.2. pSearch

pSearch is a semantic-based P2P network proposed in [Tang, et al]. It is a structured network that organizes a semantic overlay using LSI and content-addressable network (CAN) technology. The major feature of pSearch is to combine the efficiency of DHT systems and accuracy of information retrieval algorithm.

pSearch uses LSI to obtain documents' semantic vectors, which in turn are maintained using a CAN overlay. When the dimension is high, however, CAN suffers from the problem of high maintenance overhead, because the neighbor links a peer needs to maintain increases drastically. To address this problem, pSearch propose a dimension reduction technique, called *rolling index*. Rolling index partitions a small portion of a semantic vector into p subvectors. Each subvector consists of m dimensions corresponding to an m-dimensional space. An example is shown in Fig. 2-5, in which the first six elements of a semantic vector is partitioned into three 2-dimensional vectors and are separately mapped to three 2-dimensional CAN spaces.

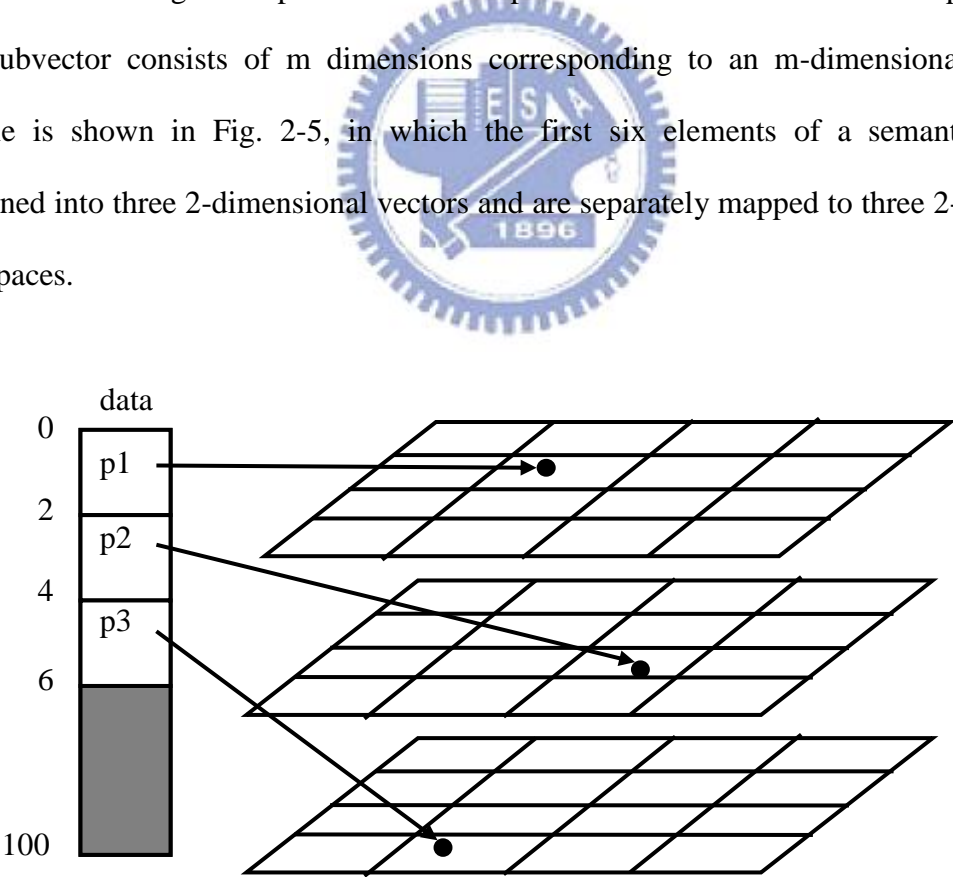


Fig. 2-5. Rolling index in pSearch

Because each semantic vector needs to publish and search p times, rolling index still incurs high publishing overhead and search cost. Because CAN is a structured network, imbalance distribution of document over the peers may still happen.

2.2.3. Semantic Small World

Semantic small world (SSW) is presented by [Mei Li et al] in 2004 and is closely related to pSearch. It is also a structured network using CAN as the underlying overlay. SSW introduces two new features: semantic clustering and small world. In the SSW overlay, a zone not only represents a peer, but also consists of several peers to share the responsible of managing a semantic zone. To reduce average search path, SSW adopts the small world property: each peer maintains short-range links and long-range links. An illustrative example for SSW is shown in Fig. 2-6.

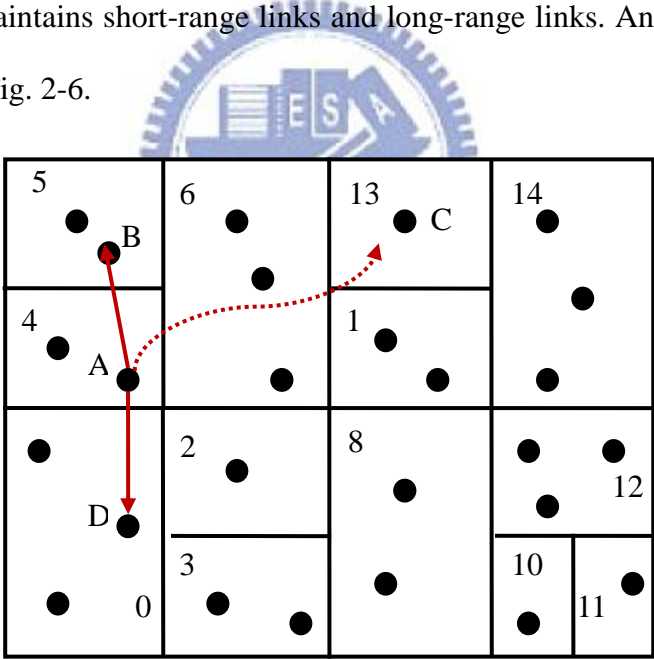


Fig. 2-6. An illustrative example for SSW

Like pSearch, to address the high maintenance overhead associated with high dimensional semantic space, SSW proposes the dimension reduction technique, called *SSW-ID*. The idea is to linearize the peer clusters from high dimensional semantic space to a *one-dimensional naming space*. Specifically, SSW assigns a unique ClusterID to each cluster

using a virtual binary tree (GPT). An illustrative example of SSW-1D is shown in Fig. 2-7. The root node of GPT represents the initial space. Each partition generates two subtrees. The left subtree attaches bit 0 and the right subtree attaches bit 1. The subspace corresponds to a leaf node in the GPT. With this set up, the tree label of a leaf node is used as the clusterID for a cluster. SSW-1D is constructed as a ring consisting of clusters. Two peer in two neighbor clusters connect via a short-range link. Each peer has several long-range links to connect to other clusters.

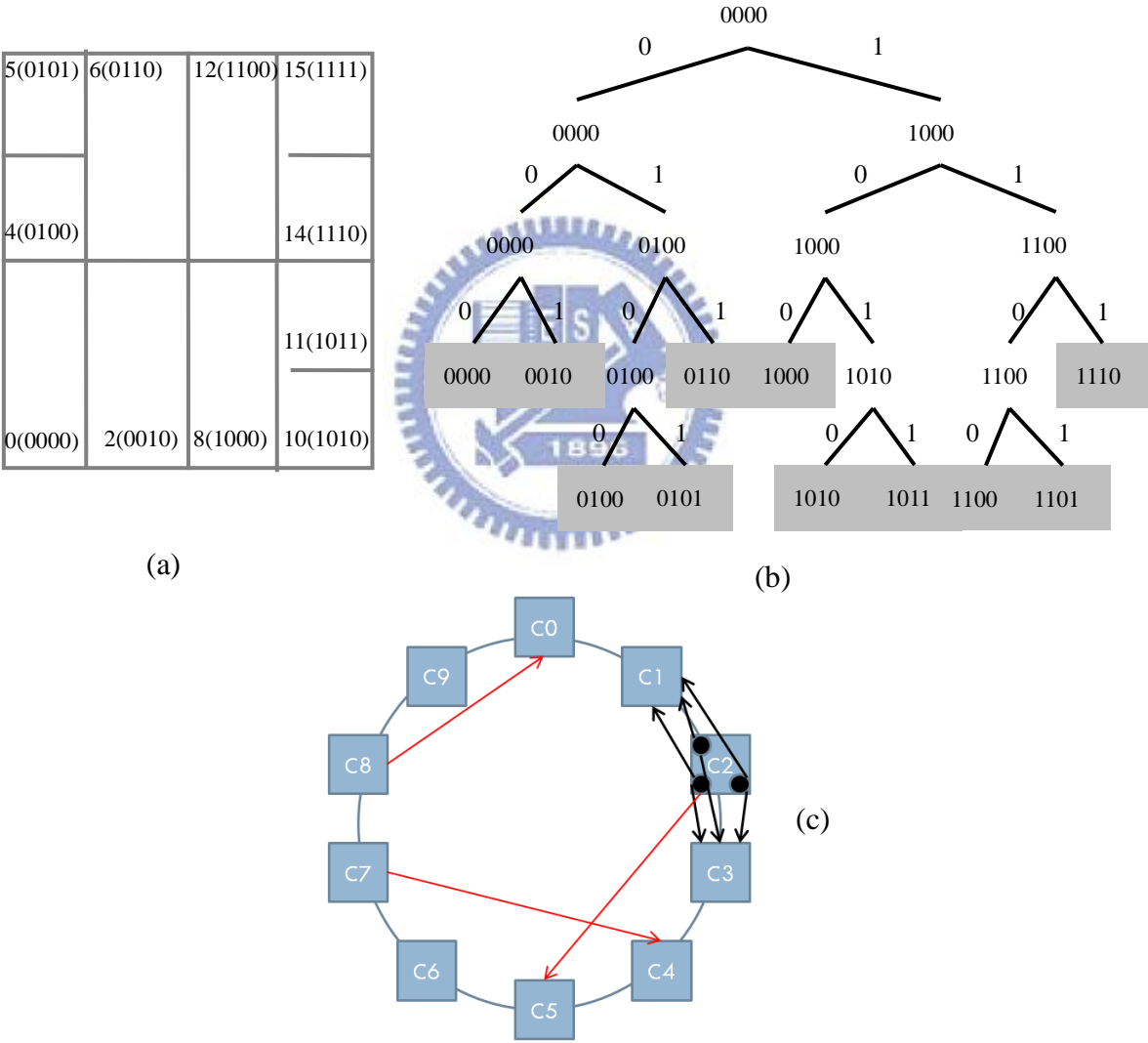


Fig. 2-7. An illustrative example of SSW-1D. (a) clusterID. (b) GPT. (c) SSW-1D

Although SSW reduces the high maintenance overhead associated with CAN and the high publishing cost required by pSearch, the imbalance distribution of documents still exists

in SSW. Furthermore, as a structured network based on DHT, implementing SSW is still a complex task.



Chapter 3. Self-Organizing P2P Network Architecture

3.1. Overview

We design an unstructured P2P overlay supporting semantic search. Our approach adopts many design principles from the P2P networks discussed in earlier chapters, notably the Freenet architecture and the use of LSI. In the overlay, each document is assigned a k -dimensional semantic vector obtained using the LSI transformation. Each peer is also assigned a point in the semantic space based on the documents it contains. Initially, a peer may contain documents with diverse topics such that their corresponding semantic vectors show not many patterns. Our goal is to make each peer focused. When a peer finds out that there is a document closer to its center than some of the documents in its routing table, the one in the routing table will be replaced with the new document if the routing table is full. Because changing the routing table in turn changes the peer's position, it is as though the peer continuously moves in the semantic space, with the document set it contains becoming more focused gradually. The idea is illustrated in Fig. 3-1.

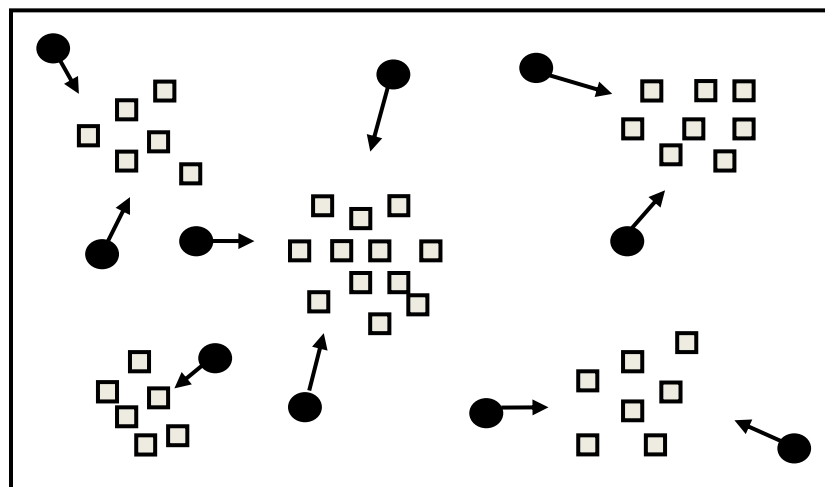


Fig. 3-1. The key idea for self-organizing

3.2. Overlay Construction

Document table and routing table. Document table records information about the documents managed by a peer. There are two types of information in the document table: document link and document cache. Document link contains is the reference information, including the semantic vector and information about the owner's information. In case the document owner is the peer itself, the actual document content is stored in the document cache. Routing table records information of all neighboring peers known to the peer. Two types of links are maintained: short-range links and long-range links. Short-range links always reference to the peers whose (current) positions are within a given range in the semantic space (i.e. within the peer's proximity). Short-range links enable efficient retrieval of relevant documents in a few hops. In contrast, long-range links reference distant peers in the semantic space, allowing a query to be routed quickly to the target regions.

Peer positioning. As mentioned previously, each peer is assigned a semantic vector that may change based on the network state. Although there are many ways the semantic vector can be determined, in this thesis we focus on the case that the semantic vector should only be based on the documents it contains as well as other peers' positions. Ideally, we want to position the peer at the center of the documents. We achieve this by simply using the *median* of documents' semantic vectors, which prevents the position of the peer from changing too much within a short period of time, especially when a document with large distance from the peer's current position is swapped out the routing table. The detailed equation as follows:

$$P_{sv} = [p_1, p_2, \dots, p_k] \quad \text{where} \\ p_i = \underset{d \in D_{link}}{\text{median}}(d_i) \quad (2)$$

where D_{link} is the set of document links P manages. Fig. 3-2 shows a simple example about the calculation of the medium.

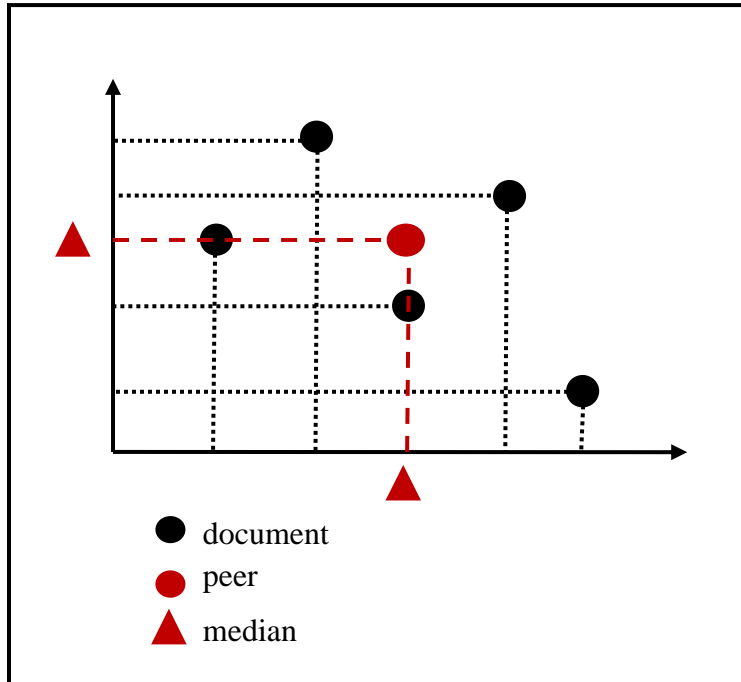


Fig. 3-2. Peer positioning using medians

3.3. Semantic Search

When performing a semantic search, the user needs to provide both a range (a sphere with a center and a radius) in the semantic space and the number of documents to be returned. In addition, the maximum number of hops (i.e. TTL value) should also be given. The search procedure will continue collecting documents that fall into the given range until either the required number of documents has been collected, or the TTL is reached. Clearly, the quality of the query depends on the document distribution and the parameters provided by the user. However, in reality, it is possible for the user to provide only the center, and the system can fill in the other parameters probably, possibly after conducting several “sampling” queries and gaining sufficient knowledge about the distribution of documents around the center.

Below we describe the search procedure in more details.

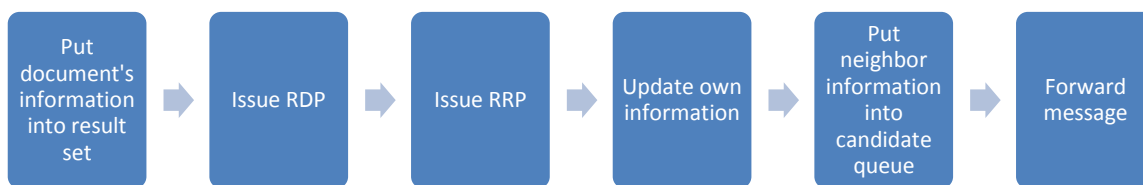


Fig. 3-3. The flow chart of the request procedure

As shown in Fig. 3-3, the procedure mainly consists of six steps:

Put document's information into result set. When a query is initiated, it contains an empty result set used to hold documents collected so far, as well as the peers that contain the documents, respectively. The result set has an upper bound limiting the number of documents it can contain. We can conveniently set the limit to the required number of documents given in the query. When a peer receives a query, it first looks up its own document link table and whether to put some related documents it finds into the result set. If the result set is full, the peer can replace documents in the result set with better ones in its document link table. If the result set is not full, there are two choices. First, a document is put in the result set only if it falls in the query region. In the second choice, the result set is simply filled as many documents as possible, but following the same replacement strategy mentioned above. The latter approach is what we focus on because, doing so propagate more document information as well as peer information along the query path, allowing more peers to learn the network topology and their inter-relations in the network more quickly.

Replace Document Procedure (RDP). This procedure mainly replaces a document link in the document table whenever a peer gains knowledge about a new document in the network. Often the knowledge comes from the query (actually, its result set) passing by. RDP is performed to make the documents of a peer more focused. However, RDP does not drop a document link when its document is cached in the peer, to prevent the document from being

dropped from the network all together. In RDP, if the document table is not full, the new document is simply added to the document table. If the document table is full, RDP then calculate the distance (similarity) between the peer and the document, where the distance between two semantic vectors is computed using Euclidean distance:

$$d(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_k - q_k)^2} \quad (3)$$

RDP then compares the distance with d_{\max} , i.e. the document that is most distant from the peer. If the new document is closer than d_{\max} , then replace d_{\max} with the document. Algorithm 3-1 shows the pseudo codes of RDP:

Algorithm 3-1: Algorithm for RDP

RRP(Peer p, Document d) :

- 1: **if** document table isn't full **then**
- 2: add d to document table
- 3: **else**
- 4: **if** $d(d, p) < d(d_{\max}, p)$ where d_{\max} has maximum distance in the document table of p **then**
- 5: replace d_{\max} with d
- 6: **end if**
- 7: **end if**

Replace Routing Procedure (RRP). This procedure mainly updates and replaces routing table in a way similar to RDP. In this case, a peer gains new knowledge about a new peer also from the query passing by. This also implies that whenever a peer receives and/or redirects a query, it attempts to “inject” information about itself, including its current position. However, if we only keep near-by peers in the routing table, the network may gradually evolve into disjoint clusters. To avoid this, we divide the routing table into short-range link set and long-range link set. The replacement of short-range links is similar to RDP. In contrast, when

the new peer under consideration falls outside the short-range link set, RRP checks to see if adding the new peer can help the peer “expand” its horizon. This is done by computing, for each peer in the long-range link set as well as the new peer, its *total* distance between it and the rest, then keep the combination which has the maximum total distance. Of course, if original combination of long-range link set has the maximum total distance, no replacement is done. Algorithm 3-2 shows the RPP pseudo code:

Algorithm 3-2: Algorithm for RRP

```

RPP(Peer p, ResultSet rs ) : for each peer np in rs
1: if np exists in routing table then
2:   update np's information in the routing table
3: else
4:   if  $d(np, p) < d(s, p)$  where s has maximum distance among the
      short-range link set then
5:     replace peer s with peer np
6:   else
7:     calculate total distance of all combination between np and
      all long-range links
8:     if long-range link set has minimum total distance when
      replacing peer l with peer np then
9:       replace peer l with peer np
10:    end if
11:  end if
12: end if

```

Update own information. This procedure mainly re-calculates the new semantic vector, because after RDP and RRP are performed, the peer’s position may change.

Put neighbor information into candidate queue. In request procedure, the search path is along the greedy path. The request message is always forwarded to the peer that it has not

been visited and its position is closest to query. So the request message uses a candidate queue to store top k peers' information closest to query.

Forward the message. The search procedure stops if the result set has accumulated required number of documents, or the query has traversed TTL hops given in the query. In either case the query will be sent back along the request path. Otherwise, the peer will forward the query to the peer in its routing table that is closest to query. The algorithm 3-3 shows the complete pseudo code for request procedure as follows:

```
Algorithm 3-3: Algorithm for search procedure
search(Peer p, Query Q ):
Q.RS: result set, Q.CQ: candidate queue, Q.TTL: TTL
Q.S: similarity threshold, Q.TS: travel stack
1: for all d in the document table do
2:   add d to Q.RS
3: end for
4: for all document d in the Q.RS do
5:   do RDP(p, d)
6: end for
7: for all known peer r in Q do
8:   do RRP(p, r)
9: end for
10: re-calculate position for p
11: for all n in the routing table do
12:   add n to Q.CQ
13: end for
14: if Q.TTL = 0 or all d(r.d, Q.q) <= Q.S then
15:   return Q back to previous peer according to Q.TS
16: else
17:   forward Q to the peer i among Q.CQ that is closest to Q
15: end if
```

In order to reduce the maintenance overhead, we make use of the query as much as possible by packing document and peer information into the query. This is also the case when the query is returned along the search path towards the source, where all the peers along the path can still learn knowledge about new documents and other peers and update themselves accordingly. Fig. 3-4 shows the steps done when a query is returned:

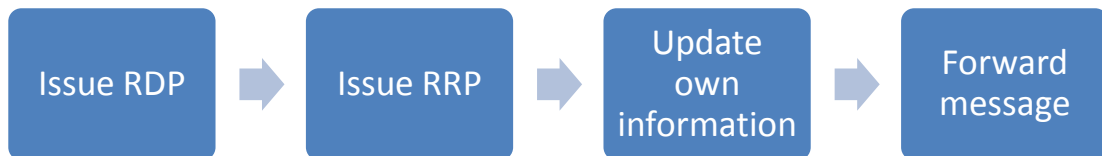


Fig. 3-4. The flow chart of the response procedure

When peer p receives the returning query, it first executes RDP and RRP again and re-calculates its own position. To send response message back along the search path, the peer still update its own entry in the query (mostly in the travel stack carried by the query). Algorithm 3-4 shows the pseudo code for the response procedure as follows:

Algorithm 3-4: Algorithm for response procedure
 response(Peer p , Query Q):
 4: **for** all document d in the $Q.RS$ **do**
 5: do RDP(p , d)
 6: **end for**
 7: **for** all peer r in Q **do**
 8: do RRP(p , r)
 9: **end for**
 5: re-calculate own position
 6: pop peer i from $Q.TS$ and forward Q toward peer i

3.4. Document Cache Swapping

In the search procedure, we use RDP to re-organize document links in the document table. Because the peer does not store the actual content locally, we can drop a document link if it falls outside the peer's range of focus. Because the actual document is still stored somewhere in the network, it can still be reached. However, if for a document there are few links to it from other peers (so no one knows about it other than its owner), and the document is kept by a peer whose position is far from the document's, it is less likely for the document to be reached using the simple greedy search heuristic. Accordingly, we augment the P2P network with some document swapping mechanism so that any document has chance to be moved to the peer closer to it. We implement two simple swap procedures: Two-side beneficial swap procedure (TBS) and one-side beneficial swap procedure (OBS), which are performed periodically. That is, from time to time, a peer will try to exchange the documents it caches but outside its interest region, with another peer which has documents closer to its interests .

Two-side beneficial swap procedure (TBS). This procedure lets two peers exchange documents that are mutually beneficial. From time to time, when peer P learns that a document link D (assume its source peer is Q), is closer to P's own position than some of the documents cached in P. Peer P can pick, among such documents, one that is closest to Q, say E, and sends a request-for-swap message to Q with D and the suggested document E. Peer Q accepts the swap request if document E is indeed better than some of its cached documents (other than D). If the request for document swap is granted, peer P and Q can subsequently exchange the documents E and D and update their routing tables correspondingly.

One side beneficial swap procedure (OBS). This procedure is also used for a peer to get rid of a cached document it does not want. Unlike TBS where the peer knows the actual

document it wants and can contact the document's source peer, OBS is initiated by the peer unilaterally. In OBS, peer P picks one of its cached document, say D, that is farthest to P's position; peer P then issues another kind of request-for-swap message which will be forwarded in the same path like initiating a query for document D from P. However, when a peer receives such a message, it can propose one of its own cached documents closest to P's position by packing this proposal into the message. When the message eventually comes back to P, the message may collect multiple proposals from the peers in the search path. Peer P can pick the document that is most suitable for itself, and contact the document's source to complete the document swap as in TBS.



Chapter 4. Peer Movement

In Chapter 3, we describe the basic self-organization mechanism for our P2P network. Each peer tries to make its collection of documents focused by replacing and swapping documents on its own. Doing this continuously, it is possible that many peers will simply converge to a small region in the semantic space. Consequently, the document links maintained by these peers may overlap significantly. Consequently, more and more documents will have few or no corresponding document links maintained by the other peers. This will degrade the search performance for the whole network. To address this problem, we can make a peer to take into consideration its relations with other close-by peers when the peer changes its position. We study a flocking behavior [Reynolds] in the bio-inspired field and adopt some of their ideas in our system.

The flocking behavior was first simulated on the computer in 1986 by Craig Reynolds. He models the flocking behavior into three simple rules: Separation, Alignment and Cohesion.



Rule 1: Separation. This rule prevents a bird from collision with other birds when steering.

Rule 2: Alignment. This rule makes a bird steer towards the average moving direction of its neighboring birds.

Rule 3: Cohesion. This rule makes a bird steer towards the average position of its neighboring birds.

Fig. 4-1 illustrates the three basic rules. The gray circle is the sight range of the central bird (full triangle). In the gray circle, the other birds (hollow triangle) would affect the central

bird with the three basic rules. Finally the movement of the central bird is the sum of three forces generated by three rules.

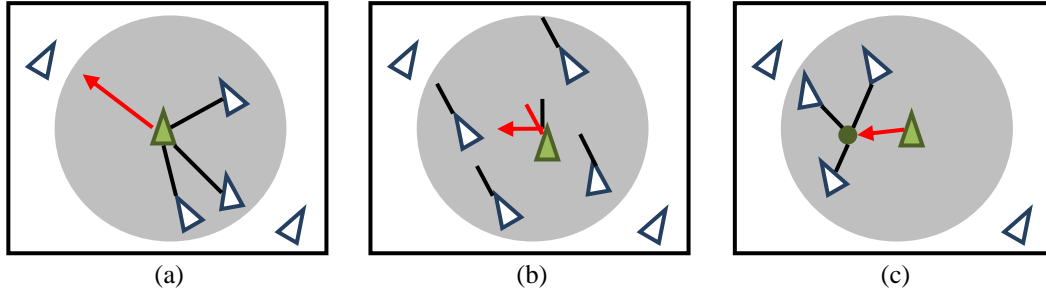


Fig. 4-1. Three basic flocking rules. (a) separation rule, (b) alignment rule, (c) cohesion rule

Following the flocking concept, we define two forces to affect the peer's movement: separation force and interest force:

Separation force. This force can separate two peers when they are too close to each other in the semantic space. There is a repulsive force between a peer and each of its neighbors. We define the repulsive force to be inversely proportional to the square of distance between two peers, as shown below:

$$\vec{V}_s = \sum_{n \in N} \frac{\vec{p} - \vec{n}}{d(p, n)^2 + c} \quad (4)$$

where N is the set of neighbors for peer p , c is a positive constant used to avoid singularity.

Interest force. This force makes each peer move to its interest zone in the semantic space. The interest zone for peer p is defined as the area surrounding by top $\alpha\%$ document links closest to p 's position in the document table. The idea behind the definition of interest zone is that if there is a document cluster near peer p , the center of the zone is often closer to the cluster center than p is, unless peer p is already at the cluster center. Therefore, we define

the attraction force between the peer p and the center of the interest zone, and make it proportional to the distance between them. The interest force is calculated as below:

$$\vec{V}_i = \overrightarrow{d_\alpha - p} \quad (5)$$

where d_α is the center of top α % document links closest to p . In our system, α is 40.

The force vector is the sum of separation force and interest force, which represents the peer's moving direction. Unlike Algorithm 1 where documents are replaced purely based on distance, here we try to kick out the document that does not help the peer move in the force vector's direction the most. To achieve this, we define the improvement score for each of the document in peer p as below:

$$s(d, p, \vec{v}) = \frac{(\overrightarrow{d - p}) \cdot \vec{v}}{|\vec{v}|} \quad (6)$$

where p is the peer's position and \vec{v} is the peer's force vector.

Chapter 5. Experiment

In this chapter, we evaluate the self-organizing effect of our system and compare our system with pSearch via simulation using both synthetic and real data set. When simulating pSearch, we take four groups of subvectors with dimension $2.3 \ln N$ where N is number of peer in the network and all peers are joined sequentially in a circle when simulation starts. The synthetic data set includes 30000 documents divided into 30 clusters. The real data set contains 26337 documents derived from the federal register (1994) in TREC version 4. We use Terrir system developed by University of Glasgow to obtain term vectors and use SVDLIBC developed by Doug Rohd to compute the SVD and obtain semantic vectors. The main metrics we use are average search hops and average precision. The other parameters used in the simulation are summarized in Table. 5-1.

Descriptions	Default
Dimensionality of semantic vector	100
Number of peers in the network	1000
Number of initial data per peer	40
Max. document table size per peer	120
Max. routing table size per peer	30
Number of results for a query	25
Similarity threshold for a query	10
Max. TTL for a query	120

Table. 5-1. The detailed parameters used in simulation

5.1. Performance Results for 2-Dimensional Synthetic Data

To demonstrate the behavior of our network, we use a 2D synthetic data set first. The data set consists of 4200 documents grouped into 6 clusters. And there are 120 peers in the network. Fig. 5-1 (a) shows the initial position of all peers and the distribution of all documents in the semantic space. Initially, 40 % of all documents a peer managed distributes over the same cluster. A gray square point represents the semantic point of a document. A black point represents the position of a peer. Fig. 5-1 (a) shows that all documents are divided into 6 clusters and initial positions of the peers are also clustered. Fig. 5-1 (b) shows the network distribution after a period of time. From the figure, we see that all peers have already dispersed around each cluster gradually.

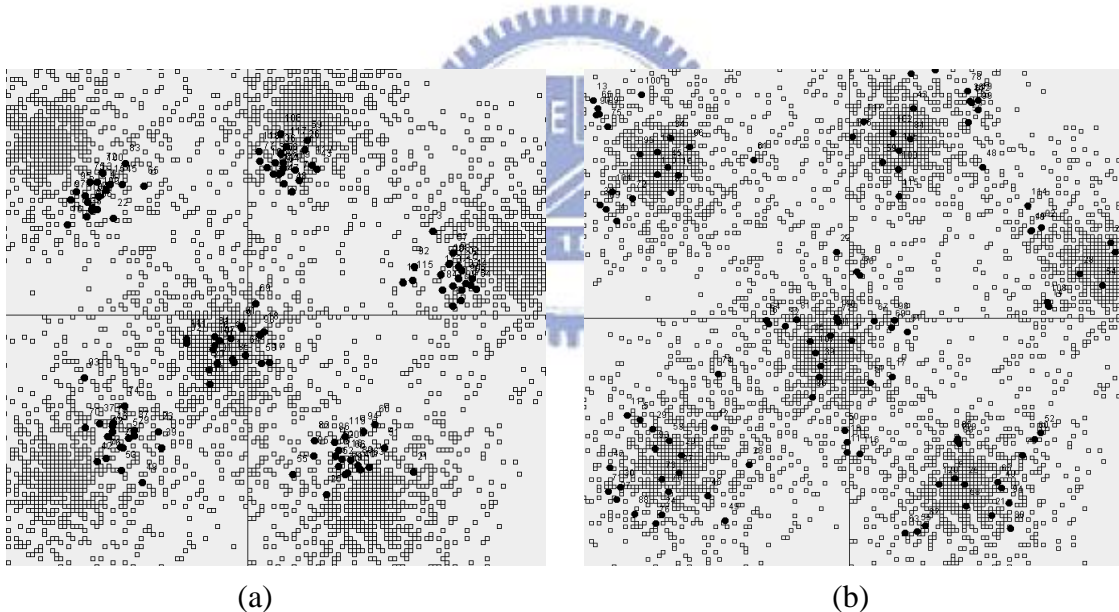


Fig. 5-1. The distribution of peers and documents. (a) initial distribution (b) the distribution after 2000 queries.

Fig 5-2 shows the change of documents owned by a particular peer over time. The peer's initial state is shown in Fig. 5-2 (a). Each black point represents a peer and the black circle represents the average distance of the top 60% documents managed by that peer; the squares represent the documents owned by the peer. Fig. 5-2 (b) and (c) shows the change of

document distribution for the peer over time, and the black circle becomes smaller gradually, showing that the peer is more focused about the documents it keeps.

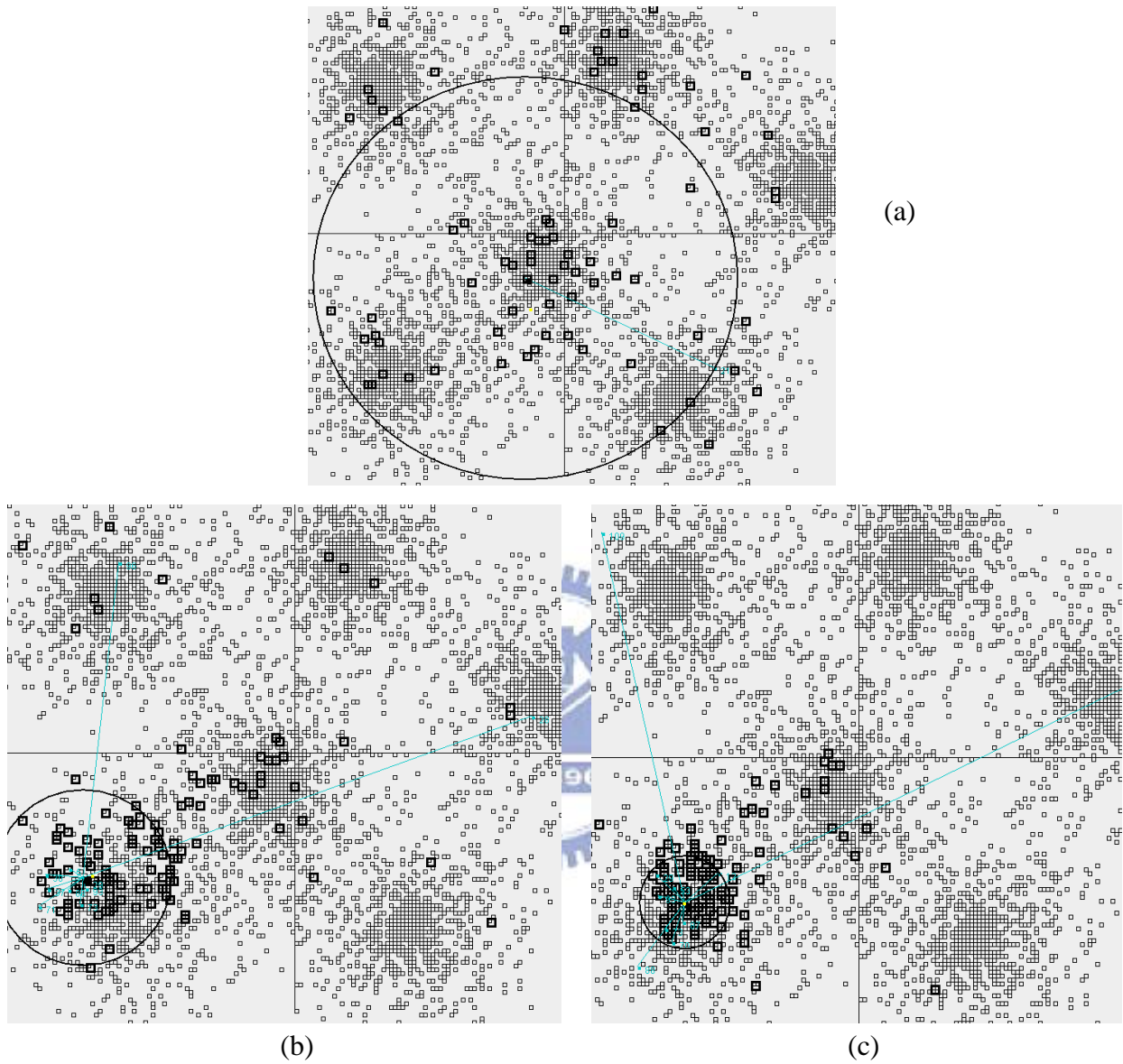


Fig. 5-2. Distribution of documents peer managed. (a) Initial (b) After 2000 queries (c) After 4000 queries

Fig. 5-3 shows that our system is self-learning by observing a query at T and $T+n$ time. In Fig. 5-3 (a), a peer S initiates a query for a region depicted by the black circle. It shows that the query has been forwarded to a peer very close to the region after three hops. But because the peer is not focused enough yet, the query is still passed around afterwards. At $T+n$ time,

the same query is sent, but because the peer has become focused, the requested documents are collected more quickly.

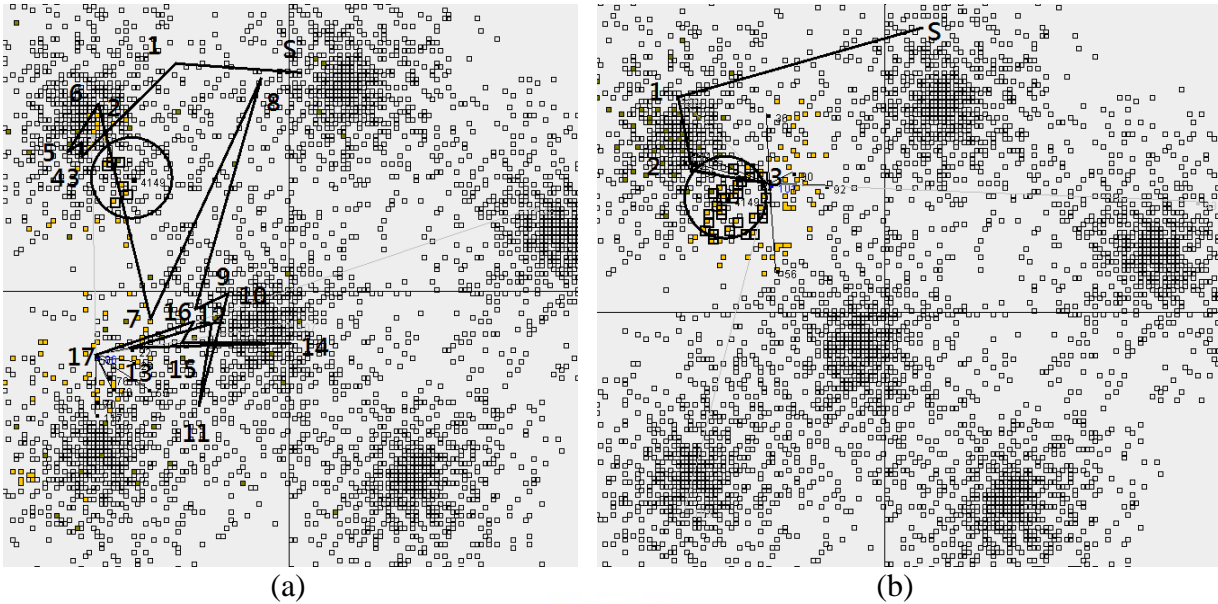


Fig. 5-3. The change of search path length. (a) At T time. (b) At T + n time

Fig. 5-4 shows the effect of the flocking. In fig. 5-4 (a), we can find that most peers concentrate in six small regions, and the regions they cover overlap too much. Fig. 5-4 (b) shows the case after we add the flocking mechanism. We can find that all peers scatter effectively around six document clusters.

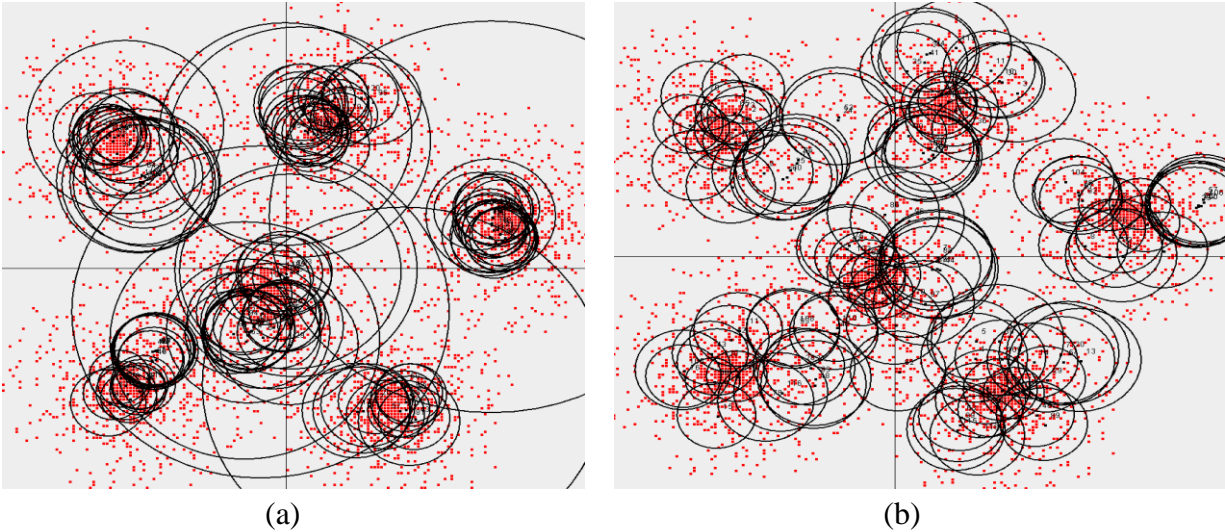


Fig. 5-4. The effect of flocking. (a) Without flocking (b) With flocking

5.2. Performance Evaluation

In section 5.1, we show that our system can make the peer move to an area and manage documents around this area gradually under the simulation of 2D data set. In this section we will carry out additional simulation using synthetic and real data set.

5.2.1. The Effect of Data Distribution

Fig. 5-5 shows the effect of data distribution per peer under synthetic and real data set. To control the initial assignment of the documents to the peers, the documents are grouped into 30 clusters first. For synthetic data this is easily done by controlling the generation of document points. For the real data, we divide the documents into 30 clusters using the K-Mean algorithm in advance. With the document grouping, each peer is given a specific group initially, $x\%$ of the documents it contains is drawn from the group, where we set x to be 0 (means totally randomly), 40, and 60. The results show that, although in the case of random document assignment ($x = 0$) the system needs more hops to finish a request in the beginning, but the required hops drop quickly as the system evolves, and eventually the search efficiency approximates the case where $x = 40$ and 60.

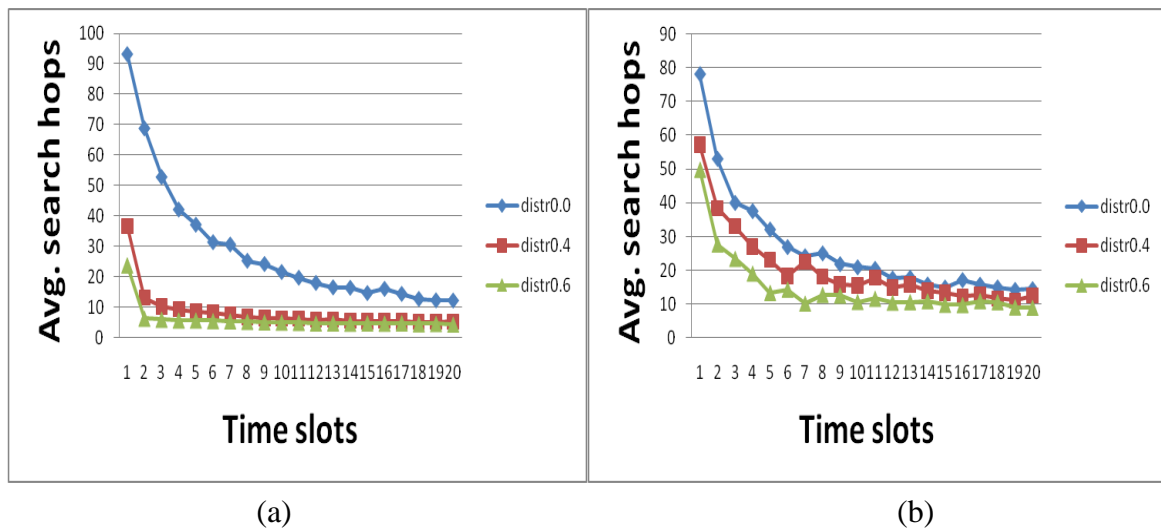


Fig. 5-5. The effect of data distribution. (a) Synthetic data set. (b) Real data set

5.2.2. The Effect of Flocking

Fig. 5-6 shows the effect of flocking on average search hops. In Fig. 5-6, we can find that the self-organizing behavior without flocking is limited since many peers concentrate on the small region. Fig. 5-7 shows the degrees of document overlap with and without flocking. In the figure, documents are ranked, along the x-axis, by the number of peers that index them. As shown in the figure, without flocking, the top 100 documents are managed by over 150 peers, and after about the 11000-th document the documents are not indexed by more than one peer.

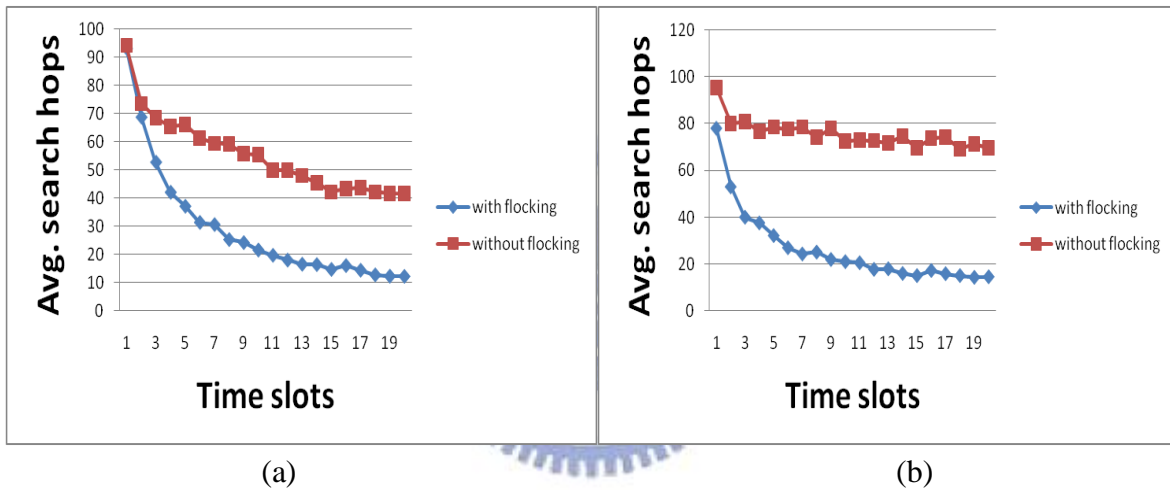


Fig. 5-6. The effect of flocking. (a) Synthetic data set. (b) Real data set

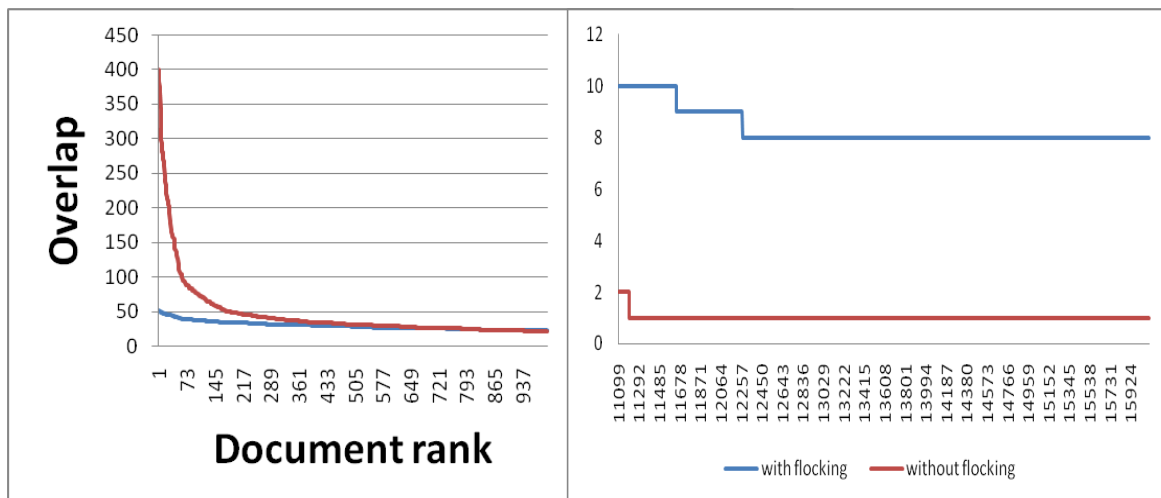


Fig. 5-7. Document overlap.

5.2.3. Search Performance

Fig. 5-8 shows the system performance compared with pSearch. In this experiment, we request the same document from the same peer at regular time intervals. In order to observe the recall rate of a request, we assume that the user knows the number of documents related to a document and asks for that number of documents. The recall metric is defined as follows:

$$R = \frac{|ret \cap rel|}{|rel|} \quad (7)$$

where *ret* is set of the returned result and *rel* is set of the documents related to the query. We observe the recall rates of our system and pSearch when the TTL is set to 20 hops and 10 hops, respectively. In the Fig. 5-8 (a), although our system has lower recall before two time slots when TTL=20, but the recall surpasses pSearch after 3rd time slot. Because pSearch is a structured network that does not adjust the network organization over time, the recall will remain the same. When TTL=20, pSearch only has 60 % recall. But our system has 75 % recall at 3rd time slot; our system even reaches 90 % recall at 19th time slot. Furthermore, pSearch only has 30 % recall when TTL=10, but our system can reach about 70 % recall when TTL=10. Fig.5-8 (b) shows the result quality under real data set and it still has better recall than pSearch after the system evolves sufficiently.

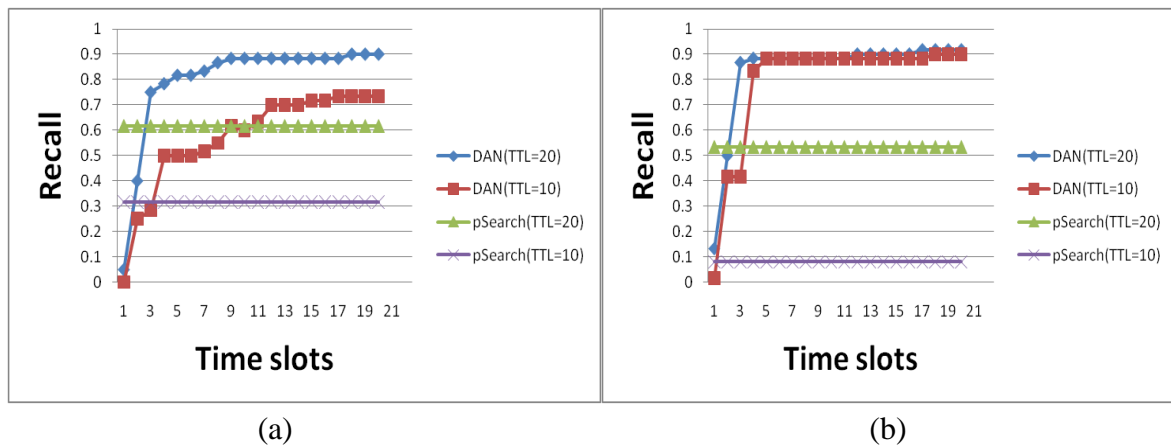


Fig. 5-8. Result quality. (a) Synthetic data set. (b) Real data set

Fig. 5-9 provides additional evidence that our network evolves into an efficient small-world network, by showing the recall hop by hop. In the Fig.5-9 (a), we observe the requests of 1st (T=1), 2nd (T=2), and 10th (T=10) time slot under TTL=20 and synthetic data set in Fig.5-8. When T=1, because the network has not stabilized yet, the recall is almost 0 % among 10 hop. When T=2, because the network has been self-organizing, the recall increases slowly. When T=10, the network has self-organized sufficiently, and a request can already collect about 80 % of the documents after just three hops. Fig.5-9 (b) shows the effect of small world under the real data set, and it also has similar behavior.

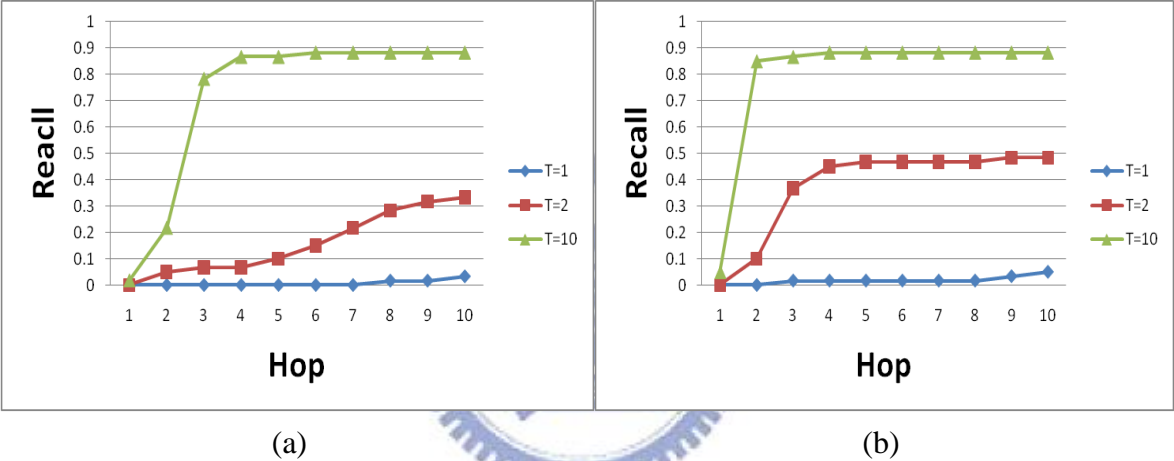


Fig. 5-9. Recall per hop. (a) Synthetic data set. (b) Real data set.

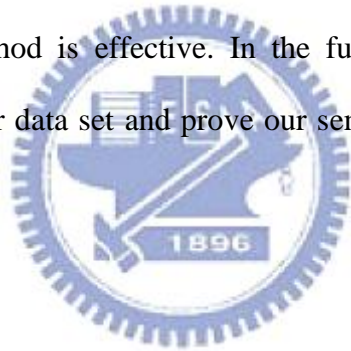
Chapter 6 Discussion and Future Work

We have shown that our system can improve search performance by letting similar documents be concentrated on few peers naturally. Fig 5-8 shows that, a request can recall about 90 % of the relevant documents after six hops when the network self-organizes sufficiently. By observing a lot of the simulation results, we also detect that a request is forwarded to fewer and fewer peers which have the documents related to the query in about ten hops.

We have applied the flocking idea in our system to address the problem that peers may overlap each other too much. However, we find that some peers are still very close to each other on the semantic space, and documents may still be distributed unevenly even when the network evolves for a long time. This is because even though two peers may be close to each other, neither peer knows the fact because it is not aware of the other (in its routing table), unless there is a suitable query passing them by. An idea to address this problem is to select a leader to manage several near peers. Each peer can then send its information, including his position, or even the documents it manages to the leader. This way, the overlapping problem can be reduced because two peers have more chance to know that they are overlapping too much through their common leader.

Chapter 7 Conclusion

We have proposed a novel self-organizing P2P network for semantic search. We adopt key principles from Freenet to make peer caches and replace document during the search. With the help of semantic-preserving document-key mapping, peers learn to concentrate on the documents in focused regions in the semantic space. In addition, peers also maintain links to other peers, including both close-by peers and distant peers in the semantic space, so that the whole network can evolve into a small-world network that permits efficient search. We also studied the flocking model and adopt some of its ideas into the system to make peers scatter more effectively on the semantic space. Simulation using synthetic data and TREC data all confirms that our method is effective. In the future, we still need to take more experiments under a large-scalar data set and prove our semantic overlay is efficiency under various situations.



References

- [Berry, et al] M. Berry, Z. Drmac, and E. Jessup. “Matrices, Vector Spaces, and Information Retrieval”, *SIAM Review*, 41(2):335–362, 1999.
- [Clarke, et al.] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, “Freenet: A Distributed Anonymous Information Storage and Retrieval System”, *In International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, vol. 2009 of Lecture Notes in Computer Science, Springer, pp. 46-66.*
- [Clarke] Ian Clarke, “A Distributed Decentralized Information Storage and Retrieval System”, Division of Informatics, University of Edinburgh, 1999
- [Cui, et al.] X. Cui, J. Gao, and T. E. Potok, “A Flocking Based Algorithm for Document Clustering Analysis”, *Journal of System Architecture* 52 (2006) 505-515.
- [Deerwester, et al] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, “Indexing by Latent Semantic Analysis”, *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- [Fanning] Shawn Fanning, Napster. (2000), available at <http://www.napster.com/>.
- [Frankel] J. Frankel and T. Pepper, The Annotated Gnutella Protocol Specification v0.4
- [Kleinberg] J. Kleinberg, “The Small-World Phenomenon: An Algorithm Perspective”, *Proc. 32nd Ann. ACM Symp. Theory of Computing (SOTC '00)*, pp. 163-170, May. 2000.
- [Kundur, et al] Deepa Kundur, Zhu Liu, Madjid Merabti, and Heather Yu, “Advances In Peer-To-Peer Content Search”, *IEEE International Conference on Multimedia and Expo, 2007*, pp. 404-407.
- [Li, et al] M. Li, W.-C. Lee, A. Sivasubramaniam, and J. Zhao, “SSW: A Small-World-Based Overlay for Peer-to-Peer Search”, *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEM, VOL. 19, NO 5, MAY 2008.*
- [Reynolds] C. Reynolds, “Flocks, herds, and schools: a distributed behavioral model”, *Computer Graphics* 21 (4) (1987) 25-34.
- [Rowstron, et al] A. Rowstron and P. Druschel, “Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems”, *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001, Heidelberg, Germany*

- [Ratnasamy, et al] S. Ratnasamy, P. Francis, M. Handdley, R. M. Karp, and S. Schenker, “A Scalable Content-Addressable Network”, *Proc. ACM SIGCOMM '01*, pp. 161-172, Aug. 2001.
- [Sandberg] Oskar Sandberg, “Distributed Routing in Small-World Networks”, *Proceedings of the eighth Workshop on Algorithm Engineering and Experiments*, Dec. 2005, pp. 144-155.
- [Shen, et al] Heng Tao Shen, Yanfeng Shu, and Bei Yu, “Efficient Semantic-Based Content Search in P2P Network”, *IEEE Transaction on Knowledge and Data Engineering*, Vol. 16, No. 7, July 2004, pp. 813-826.
- [Stoica, et al] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable Peer-to-Peer Lookup Service for Internet Applications”, *Proc. ACM SIGCOMM '01*, pp. 149-160, Aug. 2001.
- [Tang, et al] C. Tang, Z. Xu, and S. Dwarkadas, “Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Network”, *proc. ACM SIGCOMM '03*, pp.175-186, Aug. 2003.
- [Zhao et al] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiawicz, “Tapestry: A Resilient Global-Scale Overlay for Service Deployment”, *IEEE J. Selected Areas in Comm.*, vol. 22, no. 1, pp. 41-53, Jan. 2004.

