國立交通大學

網路工程研究所

碩 士 論 文

在 JXTA 同儕網路上建構關鍵字搜尋服務

Keyword Search for Enhancing JXTA Discovery Service in Peer to Peer Networks

研 究 生：張琮炫

指導教授：王國禎　教授

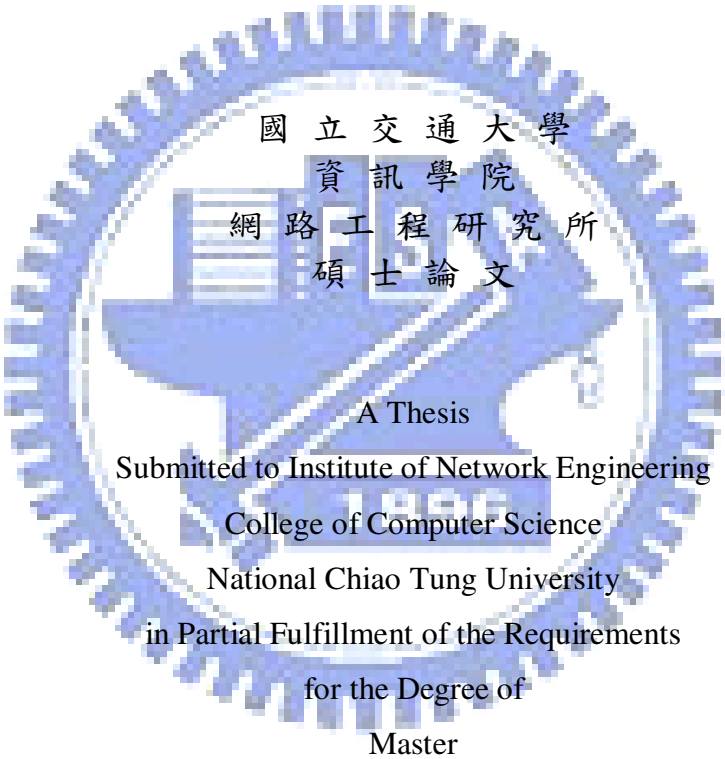
中 華 民 國　九 十 七　年 六 月

在 JXTA 同儕網路上建構關鍵字搜尋服務

Keyword Search for Enhancing JXTA Discovery Service in Peer to Peer Networks

研 究 生：張琮炫　　Student：Tsung-Hsuan Chang

指導教授：王國禎　　Advisor：Kuochen Wang



國 立 交 通 大 學

資 訊 學 院

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

在 JXTA 同儕網路上建構關鍵字搜尋服務

學生：張琮炫　　指導教授：王國禎 博士

國立交通大學 資訊學院 網路工程研究所

摘 要

JXTA(Juxtapose)是一組傳輸協定，可提供任何可以上網的設備在同儕網路(P2P)架構下進行傳遞訊息及協同運作。JXTA 具有成為下一世代同儕網路平台的潛力，並且已被應用於即時通訊、檔案分享及即時合作平台等應用程式。但是 JXTA 缺乏關鍵字搜尋的功能，導致查詢(query)必需使用搜尋對象的全名才有辦法找到。這樣的搜尋方式對使用者是很不方便的。本篇論文提出了一個 JXTA 關鍵字搜尋(JKS)服務並且支援中文關鍵字切割(CKP)。JKS 有兩個目標，第一個目標是設計新的發佈(publishing)和搜尋機制(discovery)，以達成 JXTA 關鍵字搜尋服務。第二個目標是設計中文關鍵字切割方法，以提升完全符合查詢條件的搜尋結果數量(不含重覆)。根據一些知名入口網站資料所做的模擬結果顯示，當資料大部份為中文

時，JKS 較 KAD 增加了 427%完全符合查詢條件的搜尋結果數量。在網路流量方面，當資料全為英文時，JKS 較 KAD 少 70%。當資料主要為中文時，JKS 較 KAD 少 11%。評估結果顯示 JKS 能夠應用於 JXTA 上的應用軟體，諸如檔案分享、多媒體串流分享及網路服務搜尋等。

關鍵詞：檔案分享，JXTA，關鍵字搜尋，同儕網路。

Keyword Search for Enhancing JXTA Discovery Service in Peer to Peer Networks

Student：Tsung-Hsuan Chang **Advisor：Dr. Kuochen Wang**

Department of Computer Science

National Chiao Tung University

Abstract

JXTA is a set of protocols which provide different network devices to communicate and collaborate in a P2P (peer-to peer) manner. It is emerging as the next-generation P2P platform and has been adopted by many applications, such as instant messaging systems, file sharing systems, and real-time collaboration platforms, etc. However, JXTA is lack of keyword search. Without the support of keyword search, the query must contain the exact full name of a desired resource. It is inconvenient for users in this aspect. This thesis proposes a mechanism called *JXTA Keyword Search (JKS)* with the support of *Chinese Keyword Partition* (CKP). JKS has two objectives. First, it is providing keyword search upon JXTA. New publishing and discovery schemes are proposed for keyword search. Second, it is designing a Chinese Keyword Partition method to enhance the number of exact matches. Experimental results, based on real data obtained from some well-known portal sites, show that the number of exact matches of JKS is 81% larger than that of KAD when the majorities of resources and queries are Chinese. JKS is 237% and 12% less than KAD in terms of bandwidth cost when resources and queries are all English and the majorities of resources and queries are Chinese, respectively. The research results are applicable for P2P applications (e.g., file sharing, multimedia streaming sharing, internet service discovery, etc.) built on JXTA.

Index Terms — File sharing, JXTA, keyword search, peer to peer network.

Acknowledgements

Many people have helped and encouraged me with this thesis. I appreciate my thesis advisor, Dr. Kuochen Wang, for his intensive advice and instruction. I would like to thank all the classmates in the *Mobile Computing and Broadband Networking Laboratory* (MBL) for their assistance. I am grateful to my family in church for their encouragement. I also want to thank my mother for her tender solicitude and consideration. This work was supported by the National Science Council under Grant NSC96-2628-E-009-140-MY3.

Finally, I thank my God for his continuing blessing and guiding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Peer-to-peer (P2P) networks have attracted great attention and popularity. There are many kinds of P2P applications like file sharing (e.g. eMule, Bitcomet), IPTV (e.g. PPLive, Joost), synchronous collaboration [1], etc. JXTA is a set of protocols which provide different network devices to communicate and collaborate in a P2P manner. JXTA is a common framework provided for P2P application development. There are several advantages of JXTA. (1) It accepts heterogeneous devices like PCs and mobile devices [2], [3]. (2) There is a J2ME version for mobile devices to build applications which could access JXTA networks. (3) JXTA is platform independent and can be implemented by multi-programming languages [4]. (4) It also provides firewall routing capabilities [2]. (5) It provides a consistent bottom platform for the P2P network application developers [5]. It is used by many projects to establish their applications [6], [7], [8]. It is emerging as a next-generation P2P platform [9].

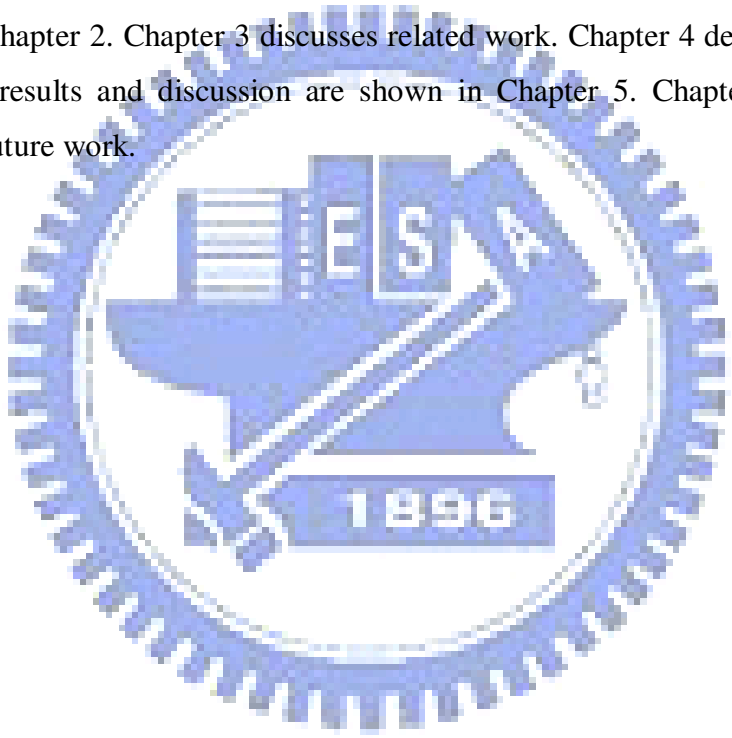
There are two types of P2P overlay networks: s*tructured* and *unstructured* [10]. Most structured P2P networks are based on distributed hash table (DHT) technology. In DHT, the index of a file is stored in selected peers. These peers are usually selected according to the hash value of the file name. In DHT, each peer is responsible for some section of hash value space. When another user wants to search a specific file, he/she needs to input a file name. A query would be directed to the responsible peer by the hash value of a user input.

The advantage of DHT is its cost-effectiveness on query routing. Given a key, it guarantees to find an object within bounded cost. But it is only useful when the user has the exact file name. Most of the time, the user may only know partial information [11]. It is hard to find out something we don't know its full name. For example, "MLB baseball game" can't be found by the query "MLB", "baseball" or "game". Thus, augmenting DHT with a keyword-based search capability is a valuable extension [12].

JXTA uses loosely-consistent DHT (LC-DHT) as the underlying query routing mechanism. The operation is like DHT which stores file index in a selected peer according to some kind of hash value. It is also lack of keyword search.

The study of this thesis is related to a NSC (National Science Council) project "Robust Peer-to-Peer Networks and Their Applications on Real-time Multimedia." This project is aiming at providing multimedia streaming on a heterogeneous P2P network and establishing a service-oriented platform based on this P2P network. The focus of this thesis is in the aspect of search technology for P2P networks. There are two objectives for this research. The first objective is designing the keyword search function upon JXTA. For example, the user can search "MLB game" or "MLB baseball" in place of "MLB baseball game." The second objective is designing a Chinese Keyword Partition (CKP) method to enhance the number of exact matches.

The rest of this thesis is organized as follows. The preliminary knowledge of JXTA is presented in Chapter 2. Chapter 3 discusses related work. Chapter 4 depicts design approach. Experimental results and discussion are shown in Chapter 5. Chapter 6 gives concluding remarks and future work.



Chapter 2

Preliminary

2.1 Introduction of JXTA [6]

JXTA is started by Sun Microsystems in 2001. It is a set of protocols which provide different network devices to communicate and collaborate in a P2P manner. Network devices can join the JXTA network by applications they run if it conforms to the JXTA set of protocols and could parse XML. The JXTA network is able to cross barriers like firewalls and Network Address Translation (NAT) [13].

JXTA consists of a specification of XML-based protocols, as shown in Fig. 1, that provide basic services common to most P2P applications, such as resource publish and discovery, and inter-peer communication [14].

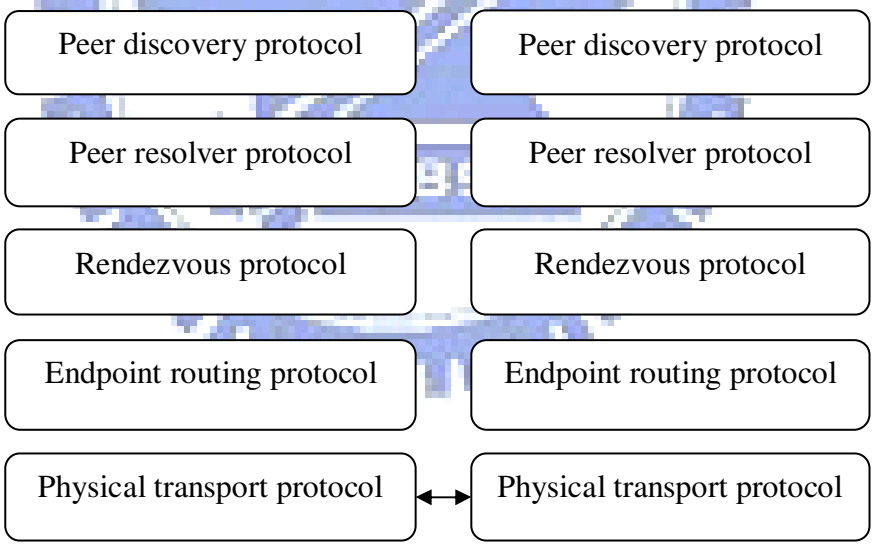


Fig. 1. Partial view of JXTA protocol stack [14].

Fig. 1 presents the stack of JXTA protocols. Above the physical transport protocol, the endpoint routing protocol (ERP) is used by peers to find routing paths to destination ports on other peers. The rendezvous protocol (RVP) is used for an edge peer to advertise local resources. A rendezvous peer uses RVP to propagate messages and maintain topology. The peer resolver protocol (PRP) enables peers to send a generic query to one or more peers and

receive a response (or multiple responses) to the query. The peer discovery protocol (PDP) is used for publishing and discovering resources [15].

An advertisement is an XML document. JXTA uses advertisements to describe resources. Resources can include files, documents, pipes, and media etc. but can also include real world resources such as sensors and printers [15]. An example of advertisements is showed in Fig. 2. While publishing, some pair of <field, key> would be extracted from the advertisement. In Fig. 2, a field means “Id”, “Type”, or “Name”. The corresponding value for a field is the content enclosed between the start tag and the end tag.

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
<Id>
urn:jxta:uuid-59616261646162614E504720503250338E3E
786229EA460DADC1A176B69B731504
</Id>
<Type>
JxtaUnicast
</Type>
<Name>
TestPipe
</Name>
</jxta:PipeAdvertisement>
```

Fig. 2. An example of a pipe advertisement [15].

The JXTA network is composed of connected peers. There are several kinds of peers. Two of them are shown in the Fig. 3. They are edge peers and rendezvous peers. The other kinds of peers are not discussed in this thesis. Most peers in a JXTA network are edge peers. An edge peer can perform search, process discovery requests from others, but doesn’t involve forwarding discovery requests. Rendezvous peers have a list of other rendezvous peers and communicate with each other in order to maintain the list. They help to propagate discovery requests to other peers.

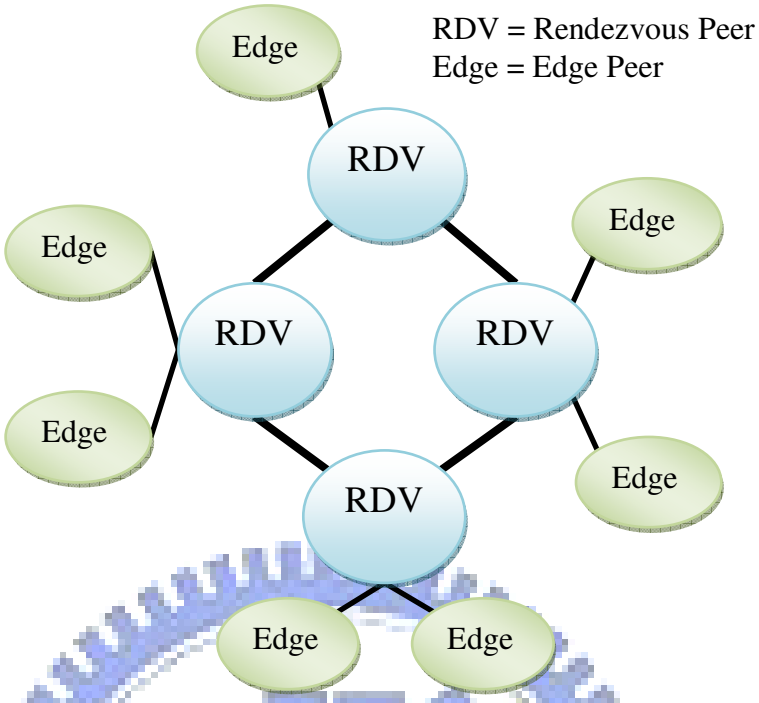


Fig. 3. JXTA network organization.

2.2 JXTA routing, publish and discovery service

Previous section gives an overview of JXTA. Here we briefly describe JXTA routing, publish and discovery service [16]. The routing mechanism in JXTA is LC-DHT. The paper in [17] provides detail description of LC-DHT.

Routing

In JXTA, each Rendezvous peer has a Rendezvous Peer View (RPV) which is an ordered list of known rendezvous peers by their peer IDs. A RPV is maintained through exchanging rendezvous peer information or through well-known seeding rendezvous peers [17]. Moreover, the message received from or sent to other rendezvous peers would update the RPV [16]. The RPVs on different rendezvous peers are not always consistent due to peer joining and leaving. But they will eventually converge to a consistent RPV when changes in the network are not too frequent [16].

For routing a key in a traditional DHT, suppose *h(key)* is the hash value of the key. The key is routed to a peer whose ID is close to the hash value of the key. In LC-DHT, the proportion

of *h(key)* to the maximum hash value is computed first. A rendezvous peer is selected from the RPV according to the proportion. Then the key is routed to the selected peer.

Publish

The file owner allows others to search its files by publishing the index entries. Each entry contains a pair of <*field, value*> which is extracted from an advertisement. An advertisement is created from each file and contains information of the file. Table 1 shows some pairs of <*field, value*> extracted from an advertisement shown in Fig. 2.

Table 1. Some pairs of <*field, value*> extracted from an advertisement in Fig. 2.

Field	**Value**
ID	urn:jxta:uuid-59616261646162614E504720503250338E3E786229EA460DADC1A176B69B731504
Type	JxtaUnicast
Name	TestPipe

In the edge peer, the step of publish is listed below:

1. Create an advertisement of files and store it into the local advertisement database.
2. Extract index entries from the local advertisement database and send these entries to a connected rendezvous peer.
3. In a rendezvous peer: Retrieve entries from index messages and store them into the index entry database.
4. In a rendezvous peer: For each entry of index messages, use a routing mechanism to send each entry to a different rendezvous peer and then the received peer stores it.

Discovery

Discovery service helps users to find out who owns files they are looking for. JXTA discovery service allows user input one set <*Field, Value*> (e.g. <name, “A Whole World”>) and finds out advertisements which contain it.

Since an RPV is not always consistent, it is possible to route a query to an incorrect rendezvous peer. JXTA designed a limited-range *walker algorithm* to cope with it. In this case, the query walks along rendezvous peers in both directions of the RPV. The walker is stopped when a result is found, the end of RPV is reached, or a defined hop count is run out [16].

In an edge peer, the step of discovery is listed below:

1. Create a query message that contains user input, and then send the query message to a connected Rendezvous peer.
2. In a rendezvous peer: search its local advertisement database.
   2a. Find results: return found advertisements to the query peer.
   2b. No result: go to step 3.
3. In a rendezvous peer: search its index entry database.
   3a. Find results: forward the query to the peers which have published the index entry. The owner peer then returns the desired advertisement to the query requester.
   3b. No result: if the rendezvous peer is the responsible peer go to step 5; otherwise, go to step 4.
4. Forward the query to a responsible rendezvous peer. The receiving peer will start from step 2.
5. Start the walker algorithm (described above).

The JXTA publish and discovery scenario is illustrated in Fig. 4. The number nearby an arrow line is corresponding to the step described above.

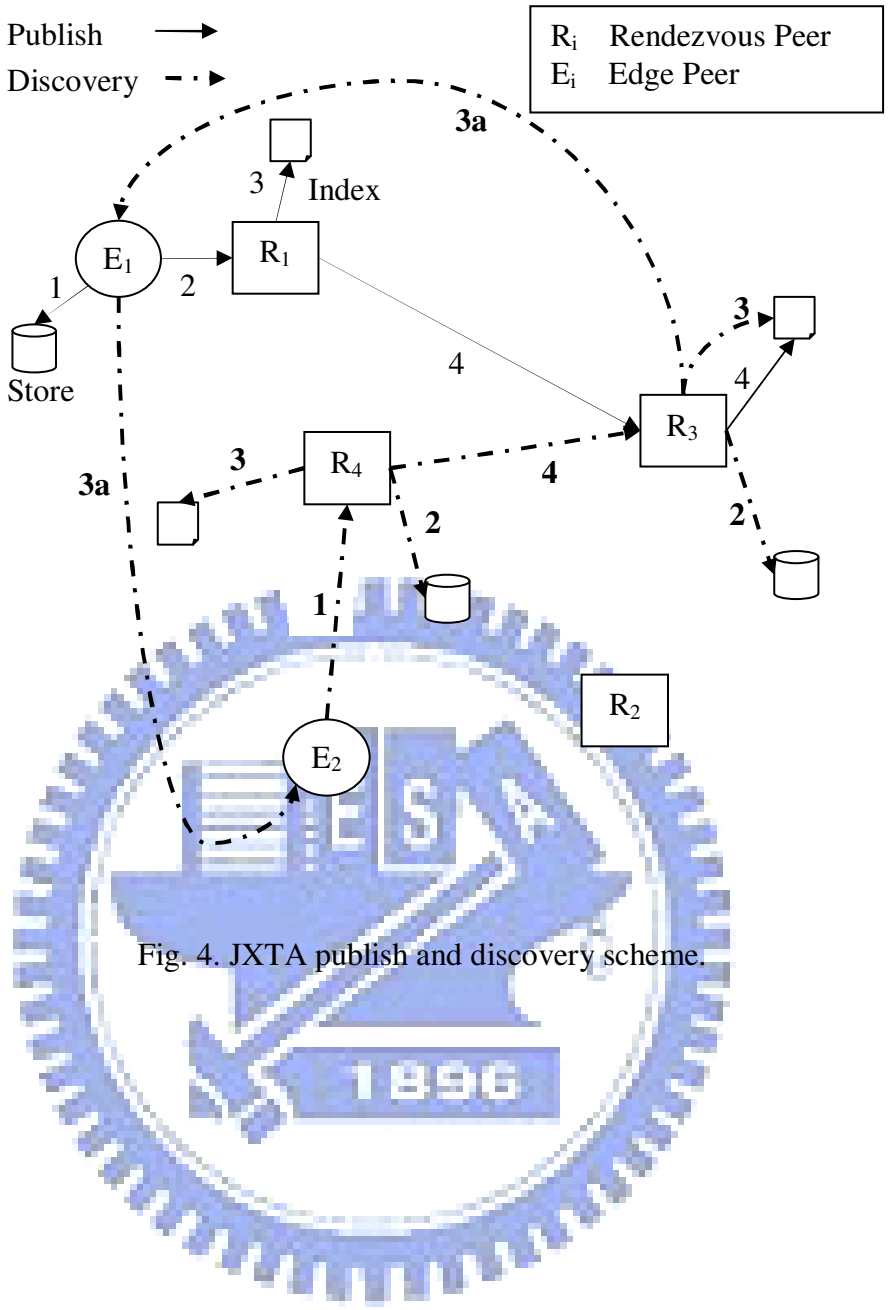


Fig. 4. JXTA publish and discovery scheme.

Chapter 3

Related Work

3.1 JXSE-CMS [18]

JXSE-CMS (content manager service) provides file sharing and retrieving services on JXTA. It also provides metadata search, but this function can't be used in the remote search because an index is found only when a query is routed to the rendezvous peer which contains the index. JXSE-CMS define its own content advertisement which could contain *content id* (*cid*), length, description, name and etc. The *cid* field contains the unique 128-bit MD5 checksum of the file. The index of an advertisement is published by the underlying JXTA functions. JXSE-CMS also provides list and request functions. The former is used for asking a peer listing shared files. The later is used for retrieving files. The latest JXSE-CMS version is 2.5.

3.2 Rich metadata searches [19]

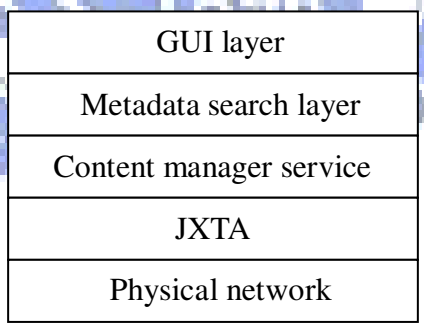


Fig. 5. Metadata architecture.

In Fig. 5, the metadata search layer is implemented based on the existing content manager service (CMS). The abundant metadata is added into an advertisement so that much more searchable attributes could be supported in a query expression. The query could include author, publisher, etc. But this mechanism can't work in a recent JXTA version in the internet because the resource is stored in a selected peer according to some kind of a hash value. The metadata search only works in a local network.

3.3 KAD [20]

There are many DHT-based methods that were proposed but very few of them are wildly deployed [21]. KAD is one of these and is based on Kademlia networks. Several papers pointed out that there are several million users using KAD [21]. There are four major clients supporting KAD [22]. These clients are eMule, aMule, MLDonkey, Lphant. The most popular one is eMule, and it is kept improving. The latest version of eMule is 0.49a [23]. In the following, we briefly describe about operations related to the keyword search in KAD. The content below including keyword partition, publishing objects and searching objects.

Keyword partition

The keyword sets are extracted by ripping names of objects. A name is ripped by some special characters, e.g. space, '.', ',', ':', '-', '_', '*', '?', etc. The keywords that consist of two or less letters are deleted [23]. For example, the keyword set of "A Whole World" is {"Whole", "World"}.

Publish objects

A peer enables other peers access its owning files by publishing references. There are two kinds of references: *source key* and *keyword key* [21].

1. Source key

A source key is computed by hashing the content of the file. It is used to search the peers owning the file.

2. Keyword key

A keyword key is computed by hashing the keyword gained from the keyword partition described above. It is used to search the files whose name contains the keyword.

The 2-level publishing scheme publishes these two kinds of references. The benefit of the 2-level scheme is decreasing number of publishing references [20].

The references are stored in the peers whose KAD ID agrees in the first 8 bits with the reference [24]. To avoid the references lost due to peer leaving, each reference is replicated in at least 10 peers [20].

The 2-level publishing scheme is described as follows:

1. Source publishing

It finds and stores a source key in the responsible peers. The address information of a publishing peer is stored with the key.

2. Keyword publishing

It finds and stores a keyword key in the responsible peers. The source key and related information of a published file (e.g. size, type, name, etc.) are stored with the key.

Fig. 6 describes the publishing scheme.

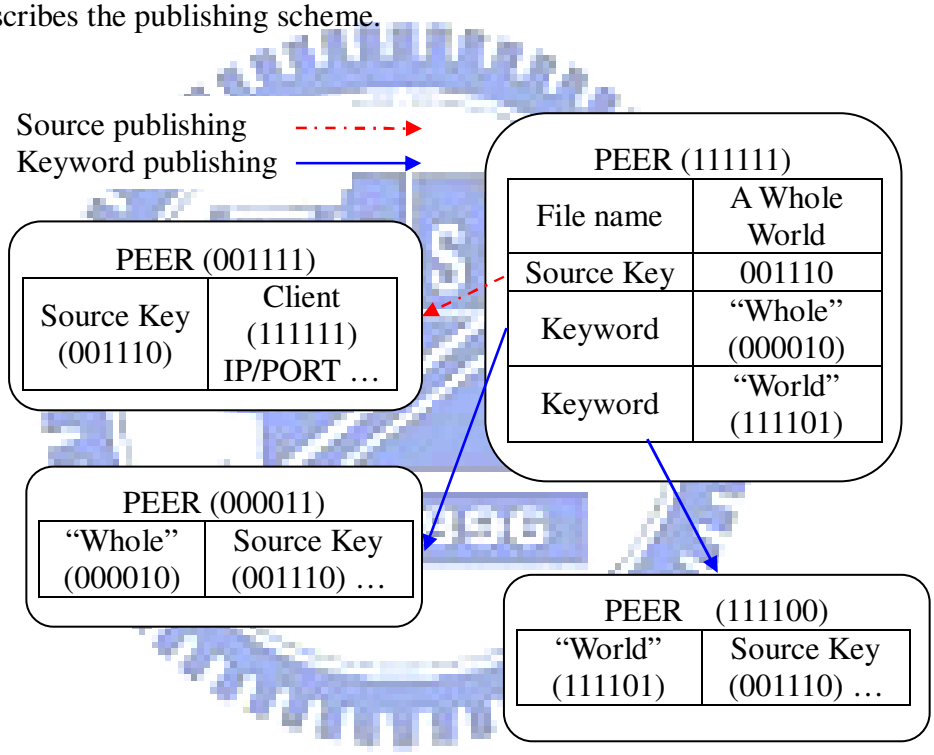


Fig. 6. KAD publishing scheme.

Search objects

Like publishing, searching files is also a 2-level scheme: *keyword search* and *source search.*

1. Keyword search

To start a search for a file, the user needs input at least one keyword of the name of the file which he/she desires. The keyword length must be longer than three letters. The hash value of

the first keyword is computed. The rest of keywords and additional attributes are packed in form of a search tree. A query consists of a hash value of the first keyword and a search tree. The query is routed to the peers that have a KAD ID close to the hash value. The matching results are responded from the peers and carry the information of source keys and files.

2. Source search

A user chooses a desired file from returned results. Then the source key of the file is used for searching the peers who have the file. This process is like keyword search. The returned results would be added into the download queue of the file.

Table 2 shows the comparison of the existing methods described above and the proposed JKS.

Table 2. Qualitative comparison of existing approaches.

Approach	JXSE-CMS [18]	Rich [19]	KAD [20]	JKS (Proposed)
P2P network	JXTA	JXTA	Kademlia	JXTA
Search capability	Metadata	Metadata	Keyword + Attribute	Keyword + Attribute
Query key	Content id or exact file name[1]		Keyword + Attribute	
Special feature	N/A	N/A	N/A	Chinese Keyword Partition (CKP)
Number of exact matches	Low	Low	Medium	High
Bandwidth cost	Low	Low	High	Medium
Storage cost	Low	Low	Low	High

[1] Before searching for a file, a user needs to know the content id of the file or the exact file name.

Chapter 4

Design Approach

The proposed JKS has two objectives: (1) adding a keyword search function upon JXTA, (2) designing a Chinese Keyword Partition (CKP) method into JXTA. The latest Java implementation of JXTA is JXSE 2.5. JKS is designed for the source code of JXSE 2.5.

4.1 Keyword search

The solution of how to add keyword search functionality upon JXTA is described in two parts: publish and discovery.

Publish

The publish scheme allows to publish several keywords for a single file. It involves adding keyword partition, a new advertisement, and modifying the publishing behavior of each rendezvous peer. The aim of keyword partition is retrieving some keywords from a file for publishing so that other users could search this file out through these keywords. Fig. 7 shows the algorithm of keyword partition.

Keyword Partition Algorithm:

INPUT *file_name*

OUTPUT *Keyword_Set*

Keyword_Set = Split *file_name* by special characters
{space, '.', ',', ':', '-', '_', '*', '?', etc.}

For each *Keyword* in *Keyword_Set*

  IF *Keyword.length* < 3 **THEN**

    Remove *Keyword* from *Keyword_Set*

  ENDIF

return *Keyword_Set*

Fig. 7. Keyword partition algorithm.

JXSE-CMS defines a content advertisement which is extended from a JXTA advertisement. The publish scheme uses the content advertisement to describe files. For example, Fig. 8 shows the content advertisement for the file “World Map.”

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:ContentAdvertisement>
<jxta:ContentAdvertisement>
    <name>
        World Map
    </name>
    <cid>
        md5:f74af6b8a1cce51d53c01ae56c61e32c
    </cid>
    <length>
        26
    </length>
    <address>
    jxta://uuid-59616261646162614A78746150325033A6B
1A47B8C2A4A26AED3E4B9932C78E803/CMS/jxta-NetG
roup
    </address>
</jxta:ContentAdvertisement>
```

Fig. 8. An example of content advertisement.

To let users search files by file attributes (e.g. author, album …), the related pairs are added into advertisements, for example, <Author> David, or <Album> 2008 Fantasy. These attributes are not index entries but are used when determining whether a query and an advertisement match.

The index entries are extracted from “name” and “cid” field of the content advertisement. Then the entries are sent to the connected rendezvous peer. When the rendezvous peer receives entries, it stores the entries with field “cid” into the index entry database. If the entry field is “name,” the keywords are extracted by the keyword partition algorithm. These keywords are added into entries and also store into the index entry database with field

“keyword.” Then the entry with field “name” is discarded.

In the original JXTA source code, the rendezvous peer distributes each entry to other rendezvous peers to store. The 2-tier hierarchical architecture allows to further decrease network traffic and storage cost. If there are multiple same entries, only one entry is sent. Besides, the sources of entries are all changed to the rendezvous peer. So during the discovery phase, the query would be routed to the rendezvous peer and then be forwarded to the real publisher.

The steps of file publishing are listed below:

1. Create content advertisement and save in a local advertisement database.
2. Extract index entries from the local advertisement database and pack these entries into an index message.
3. Send an index message to the connected rendezvous peer.
4. In the rendezvous peer: Retrieve entries from the index message and store them into the index entry database. If an entry field is “name,” it is not stored. Keywords are obtained by ripping file name and keyword pairs are stored into the index entry database.
5. In the rendezvous peer: For each entry of the index message, remove duplicated entries and change the source to the rendezvous peer. Then use the routing mechanism to send each entry to a different rendezvous peer and store it.

The file publishing scenario is illustrated in Fig. 9. The number nearby each arrow line is corresponding to the step described above.

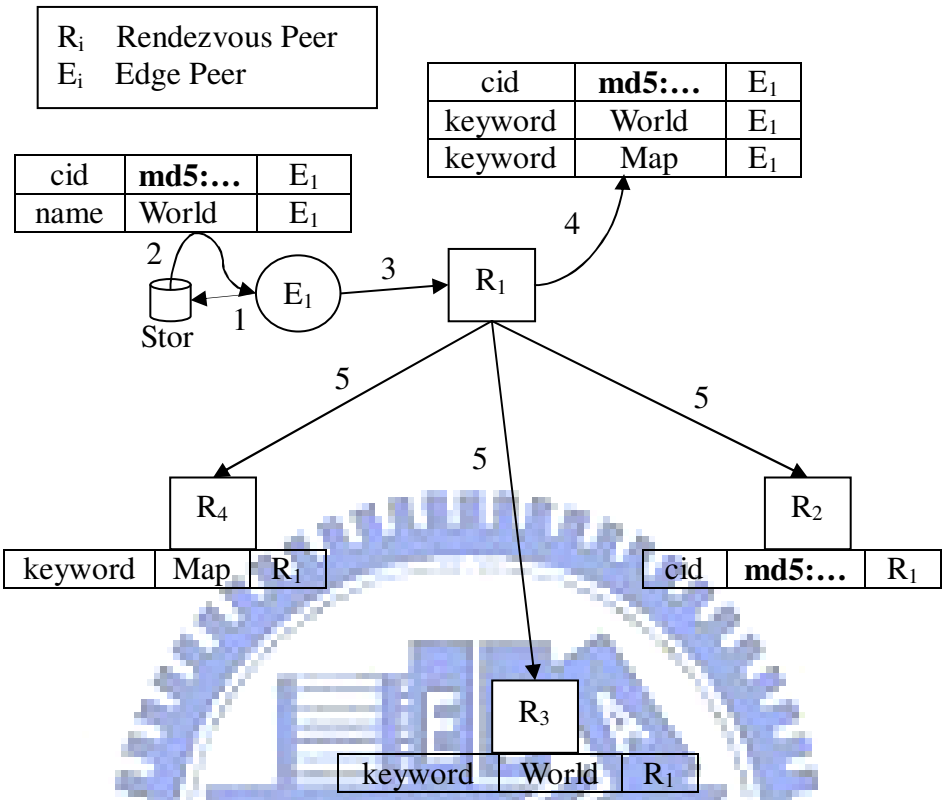


Fig. 9. Flowchart of file publishing.

Discovery

The discovery service enables users to lookup desired files by keywords and file attributes, and to seek to decrease the produced network traffic. To enable keyword and attribute discovery, the format of a query is changed. In the original JXTA, a query includes *type*, *threshold* (maximum response number), *attr*, *value*, *query ID*, *source peer ID*, etc. Attr and value is the pair of <field, value> which was mentioned before. Fig. 10 shows partial view of a query for the file with name "A Whole New World."

```
…
<Type>
2
</Type>
<Threshold>
100
</Threshold>
<Attr>
name
</Attr>
<Value>
A Whole New World
</Value>
…
```

Fig. 10. Partial view of a query.

A new query format is defined by adding three new kinds of tags into the original query format. These tags are: <Number>, <Append> and <Attribute>. <Number> is used to ask how many entries of <Attr, Value> in the peer. <Append> contains keywords and Boolean operation. <Attribute> contains file attributes. Both <Append> and <Attribute> are used to filter the results, searched by <Attr, Value>, to refine results.

To start a query, a user inputs some keywords and file attributes. To avoid generating large network traffic, a query is created for each keyword and added tag <Number>. Queries are first routed to peers who might contain index entries for the keyword. The receiving peer checks if the tag <Number> is set and number of entries for the keyword is returned. After receiving the number of entries for each keyword, the keyword with minimum number is set in tag <Value>. The <Attr> contains "keyword." The rest of keywords is set in tag <Append>, and file attributes are set in tag <Attribute>.

Then, a query is routed to the peer who might contain an index entry of <"keyword", Value>. The peer forwards the query to peers who published the entries. To decrease network traffic, only ten peers are selected for forwarding. Besides, only one query is sent if two entries are published from the same peer. After the query is forwarded to the entry publisher,

the <"keyword", Value> specified in the query is used to retrieve advertisements which contain it. If either <Append> or <Attribute> is set, it is used to filter the results. The matching results are returned to the requester.

The step of the discovery scheme is listed below:

1. Rip keywords from a query (use the keyword partition algorithm described in the publish scheme).
2. Pack each keyword into a different query and send it to the responsible peer to ask for the index number.
3. Choose a keyword with minimum index number as the query keyword. If the number is zero, the discovery process terminates. Pack the rest of the keywords and attributes into a query. Send the query to a connected rendezvous peer.
4. In the connected rendezvous peer: Send a query to the peer who is responsible for storing the index entries of the query keyword.
5. The peer receiving the query uses the query keyword to find out peers who published the index entry containing the query keyword. The processing peer then forwards the query to top ten found peers.
6. The peer receiving the forwarding query is a rendezvous peer who is connected to the real publisher. The peer finds out peers who published the entry containing the query keyword and forwards the query to the found peers.
7. The peer receiving the forwarding query finds out the advertisement which matches the requirement of the query. After that, the peer send found advertisements to the query peer.

A user chooses a desired file from returned results. If there are not enough of file owner peers acquired from returned results, the *cid* of the file is used for searching the peers who have the desired file. This process is like the KAD source key search. The discovery scheme is illustrated in Fig. 11. The number nearby an arrow line is corresponding to the step described above.

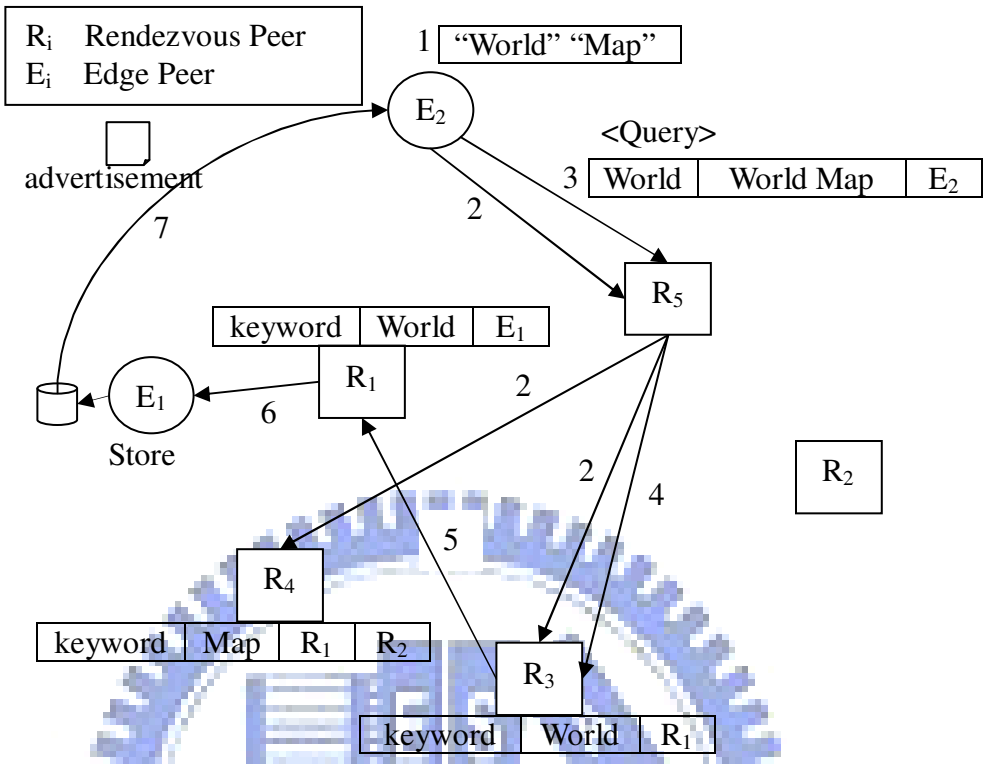


Fig. 11. Flowchart of discovering the file "World Map".

4.2 Chinese keyword partition (CKP)

The aim of CKP is to let files with Chinese names increase possibility of being searched out. In other words, increase the number of exact matching results for a query containing Chinese keywords. The character characteristic of Chinese sentences makes it difficult to be partitioned by the keyword partition algorithm in Fig. 7 because it does not have spaces between each character. Users who publish files need to manually add some special delimiters between Chinese characters. But some users may not know how to separate or are not diligent to change all files' names. It results in only a subset of files with Chinese names being searched out if a user uses a complete file name to query.

CKP provides a method to automatically distinguish Chinese keywords from Chinese file names. CKP involves a Chinese keyword list (CKL) and needs to modify the keyword partition algorithm described before. Each rendezvous peer needs to maintain a CKL. The list contains records in form of <Keyword, Frequency>. The Keyword is Chinese. When a rendezvous peer receives a query from other peers, it gets a keyword set from the query by

using the CKP algorithm. For each keyword in the keyword set, the rendezvous peer checks whether the keyword is contained in the CKL. If there is no record for the keyword, it adds a record <keyword, 1> into the CKL. Otherwise, it adds 1 to Frequency of the old record.

CKL needs to be maintained to control the use of storage. When the number of records in the CKL exceeds 2,000, all records with frequency less than 3 are removed to avoid a large memory space requirement. When the total frequencies of all records exceed 100,000, the frequencies of all records are divided by 2. If the original frequency is 1, the record is deleted. If the number of records in the CKL is less than 200, a peer uses its RPV to ask for records from other rendezvous peers. Each transmission is limited to 200 records selected from the top of the CKL. A peer would continually ask for records until the number of records exceeds 200.

An array TK (Top Keywords) contains 200 keywords with the most frequencies in the CKL. It is updated once per hour and after asking for record transmission from one of rendezvous peers. A modified keyword partition algorithm, CKP algorithm, is presented in Fig. 12.

Chinese Keyword Partition (CKP) Algorithm:

INPUT *file_name*

OUTPUT *Returned_Keyword_Set*

Key_Set = Split *file_name* by special characters

{space, '.', ',', ':', '-', '_', '*', '?', etc.}

For each *Keyword* in *Key_Set*

Split *Keyword* if it is consist of Chinese and non-Chinese and then add results into *Keyword_Set*

For each *Keyword* in *Keyword_Set*

IF *Keyword* is not Chinese **THEN**

IF *Keyword.length* > 2 **THEN**

Add *Keyword* into *Returned_Keyword_Set*

ENDIF

ELSE //*Keyword* is Chinese

Add *Keyword* into *Returned_Keyword_Set*

FOR *i* = 1 to 200

IF *Keyword* contains *TK[i]* **THEN**

Add *TK[i] into Returned_Keyword_Set*

ENDIF

ENDIF

return *Returned_Keyword_Set*

Fig. 12. Chinese keyword partition (CKP) algorithm.

From the CKP algorithm, if the file name contains any keyword of TK, the keyword is retrieved and added into the keyword set. The keyword will be packed into an index entry and be published so that another user could search the file out by the keyword. The overhead of CKP can be divided into three parts: network traffic, storage cost, CPU load.

Network traffic

It is produced by record transmissions from other rendezvous peers. It only occurs when the number of records in the CKL is less than 200. If the number of records exceeds 200, it seldom goes down to the number less than 200 again because the list keeps growing while queries keep coming. If there are enough queries in the network, the demand for transmission will be very low and the network traffic is little.

Storage cost

It is mainly produced by the CKL. Suppose each record in the CKL is 50 bytes (46 bytes for Keyword (string) and 4 bytes for Frequency (Integer)). When the CKL has the maximum record number 2,000, the size of the CKL is 2,000 * 50 bytes = 100 KB. 100 KB is small compared to the memory size currently (512MB, 1G …).

CPU load

CPU load is evaluated by elapsed system time between the beginning and end of executing the CKP algorithm in Fig. 12. A simple test program was written to run the CKP algorithm 100, 1000, 10000, 100000 times, and the elapsed system time was recorded, respectively. The program was implemented in Java and run on a PC (AMD Athlon(tm) 64 X2 Dual Core Processor 3600+ (2.01 GHz) CPU, 1 GB RAM, Windows XP). Fig. 13 shows that it takes approximately 6.5 second to run CKP algorithm 100000 times. From Fig. 13, the overhead of running CKP is little.

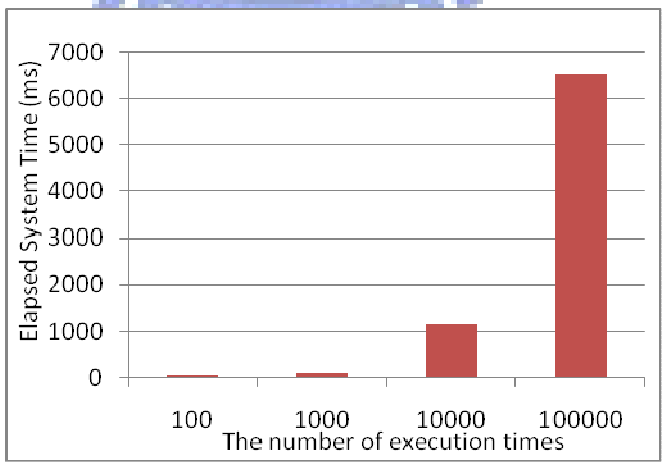


Fig. 13. The elapsed system time for running the CKP algorithm various number of times.

Chapter 5

Evaluation and Discussion

5.1 Experiment environment

Table 3 gives descriptions about four kinds of data collected for experiments. English queries were selected from query logs released by AOL [25]. Mixed queries were obtained from Yahoo TW [26]. Mixed queries consist of Chinese and non-Chinese queries with a ratio of approximately 5:2. Two kinds of files were produced in the same way. First, 2000 queries with most frequencies were selected from query data. Second, each query obtained five websites by Google SOAP search API [27]. The total number of websites is 10,000 and the titles of websites are treated as file names in the experiments. Third, each file name is assigned a replicate number according to Zipf's law. The sum of the replicate number of each file name is 100,000. Finally, file names were distributed among peers according to Zipf's law at runtime. In addition, all peers in the system are stable (no leaving or joining) and the replication mechanism for publishing is removed for easy evaluation. Table 4 shows some system parameters used in the experiments.

Table 3. Experiment data.

Data Type	Source	Amount	Date
English query	AOL query log [25]	10000	released on 2006.8.4
Mixed query	Yahoo	10000	2008.5.29
English file	Google	100000[1]	2008.5.29
Mixed file	Google	100000[1]	2008.5.29

[1] The original amount obtained from Google is 10,000; The amount 100,000 is produced by replicating each file several times according to Zipf's law which the n^{th} most frequency (number of replication) is 1/n as often as the first.

Table 4. The system parameters used in the experiments.

Notation	Description	Value
	File popularity distribution	Zipf's law [28]
	File distribution among peers	Zipf's law
N_K	Number of KAD peers	5,000
N_{JE}	Number of JXTA edge peers	5,000
N_{JR}	Number of JXTA rendezvous peers	25 [29]
T_n	Number of keywords in TK	200

5.2 Experimental results

We wrote simulation program in Java SE 6 to simulate and evaluate the performance of JKS, JXTA and KAD in terms of number of exact matches, bandwidth cost and storage cost. Table 5 gives definitions for measured metrics.

Table 5. Definitions of measured metrics.

Measured metrics	Definition
Publish cost	Average number of visited peers per file publishing
Discovery cost	Average number of visited peers per query
Bandwidth cost	Publish cost*10 + discovery cost
Storage cost	Total number of published references
Number of exact matches	Average number of exact matches per query (no duplicate)

Number of exact matches (no duplicate)

In the first experiment, it is to show how the Chinese Keyword Partition method affects the number of exact matches. All queries are sent from one peer. After receiving responses, each result is count as one but duplicate results are removed. Fig. 14 shows simulation results using two kinds of data (English and mixed). JXTA is zero in both data because it needs a query that exactly match to the file name to search out files.

For the number of exact matches, the proposed JKS is 1% less than KAD when data is English and 81% more than KAD when data is mixed. It is because JKS publishes more keywords according to most popular query keywords (TK). Besides, if a file name contains a string with a mix of Chinese, English or number (e.g. “SHE 五月天” or “104 人力銀行”), the string is partitioned and more keywords are published. The growth of published references increases with the number of exact matches. In some extent, CKP implies a way to achieve Chinese keyword search.

Fig. 14. Number of exact matches (no duplicate).

Bandwidth cost

In the second experiment, it aims to show the network traffic produced by the keyword search. The bandwidth cost consists of publish cost and discovery cost. Since the number of publish messages is ten times bigger than the number of discovery messages [30], [31], the bandwidth cost is computed as follows:

$$\text{Bandwidth cost} = \text{Publish cost} * 10 + \text{Discovery cost} \quad (1)$$

Fig. 15 shows publish cost using two kinds of data (English and mixed). In terms of average visited peers per file publishing, JXTA is 2 in both data because JXTA publishes one index entry for each file and each index entry is transmitted twice. One is from an edge peer to a connected rendezvous peer; the other is transmitted to the responsible peer. JKS is 474% less than KAD when the data is English and is 163% less than KAD when the majority of data is Chinese. There are two reasons for this. One is because JKS uses a 2-tier hierarchical architecture. Many edge peers publish their index entries to a connected rendezvous peer. The rendezvous peer publishes the index entries of keywords partitioned from the file name. When there are duplicate index entries, only one is published. The other reason is that attributes and file information are attached to the keyword key published in KAD. So when there is a same keyword appeared in two file names, the keyword is published twice in KAD but only once in JKS. However, when the majority of data is Chinese, the performance improvement margin is smaller. The reason is that CKP can retrieve more keywords from the file name so that more index entries are published.

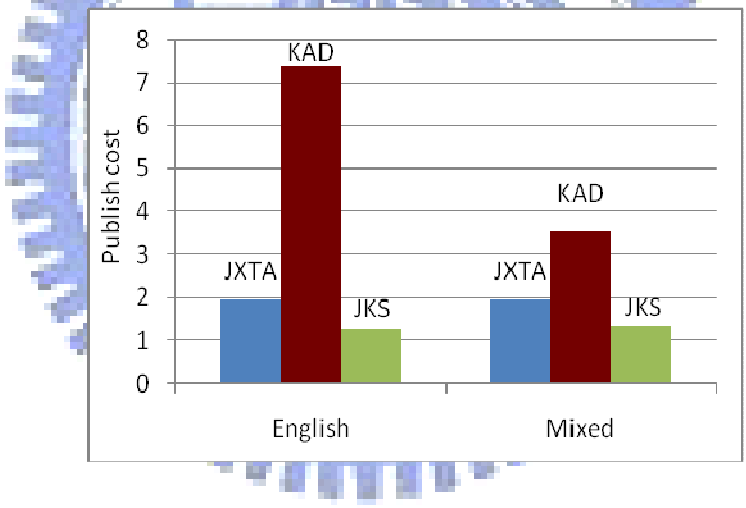


Fig. 15. Publish cost.

Fig. 16 shows discovery cost using two kinds of data (English and mixed). In terms of average visited peer per query, JXTA is around 26 in both data because when a query finds nothing in the responsible peer it goes walking. In the walking, it visits all other rendezvous peers which are 24 in the experiment. In terms of discovery cost, JKS is 89% and 95% more than KAD when the data are English and Chinese, respectively. The reason is as follows. KAD is a 2-level search scheme that searches the source key in the first level. The abundant information published with the keyword key gives the query sufficient information to determine whether it matches the query. It avoids routing a query to too many peers to find

out whether there are matching files.

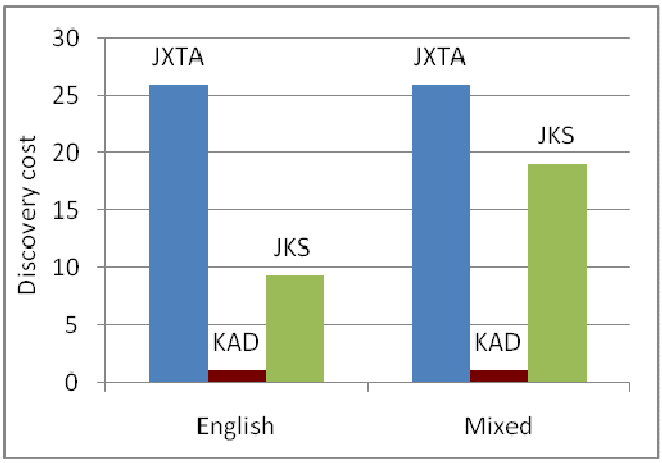


Fig. 16. Discovery cost.

Fig. 17 shows bandwidth cost computed according to equation (1). JXTA has almost the same value in both kinds of data (English and mixed). The bandwidth cost of JKS is 237% less than that of KAD when the data is English, and is 12% less than that of KAD when the majority of data is Chinese. The result shows JKS is better than KAD in both kinds of data.

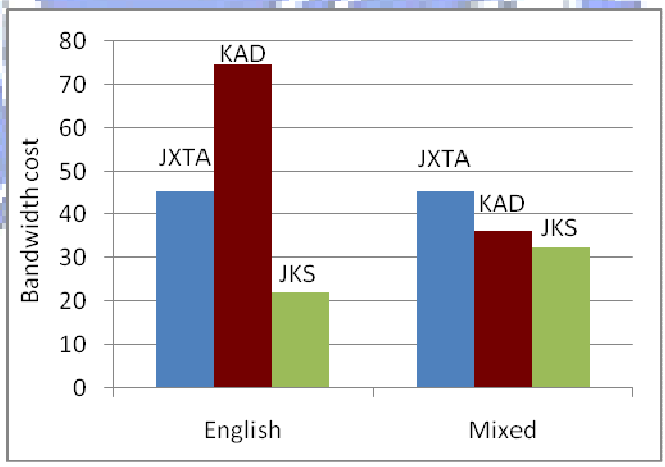


Fig. 17. Bandwidth cost.

Storage cost

In this experiment, the aim is to show the storage overhead. The storage cost is calculated after all files are published. In JXTA, all index entries stored in the index entry database are counted. In KAD, all stored references are counted. Fig. 18 shows storage cost using two kinds of data (English and mixed). The result of JXTA is near twice in the number of files in

both data. It is because each entry is stored in a connected rendezvous peer and a responsible peer. JKS is 70% more than KAD in both kinds of data. There are two reasons. One is the index entry is stored in a connected rendezvous peer and a responsible peer. The other reason is due to that KAD is a 2-level publish scheme. Assume that there is a file and the number of keywords in the file name is 3. KAD stores four references for the file (three keyword keys and one source key). Now there is another identical file published. This file produces only one stored reference (source key). The keyword keys in a file are stored only once for all identical files. This feature saves a lot of storage cost while there are many identical files in the network. Nevertheless, the memory price is very cheap. Today, one 2G memory only cost s US$33 [1].

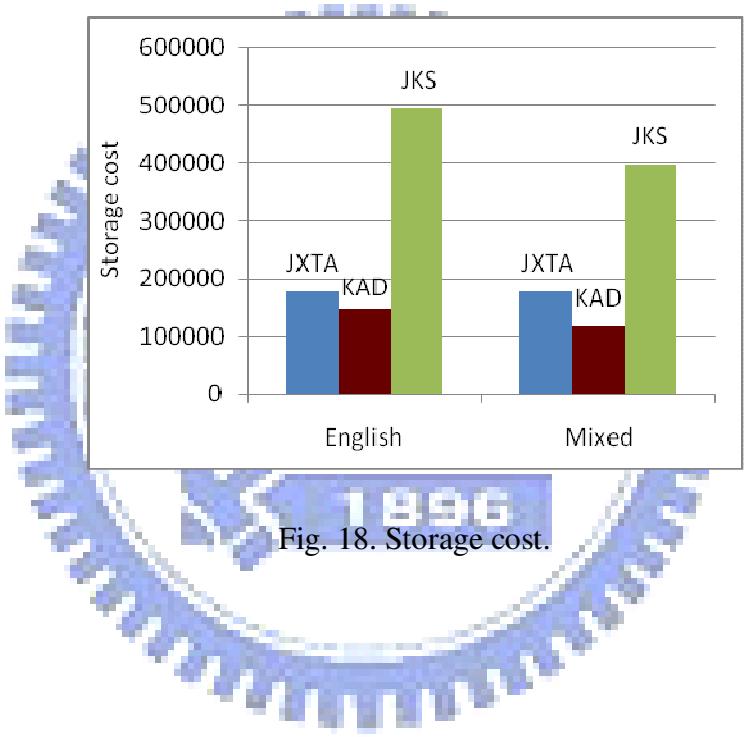


Fig. 18. Storage cost.

Chapter 6

Conclusion and Future Work

6.1 Concluding remarks

In this thesis, we have presented the design and evaluation of keyword search with CKP (Chinese Keyword Partition) upon JXTA. Experimental results have shown that JKS is 237% and 12 % less than KAD in terms of bandwidth cost when resources and queries are English and the majorities of resources and queries are Chinese, respectively. The overhead is that JKS produces 70% more storage cost than KAD. CKP helps JKS to achieve 81% more number of exact matches compared to KAD when the majority of data is Chinese. The keyword search function can be used by many P2P applications, like video on demand (VOD) streaming, file sharing etc. In addition, our CKP design is simple and has a great potential to be extended to other languages.

6.2 Future work

There are some issues that could be further studied. (1) What is the best size of TK (top keywords)? The size of TK influences the performance of bandwidth cost, storage cost and number of exact matches. Generally speaking, the more the size of TK, the more the bandwidth cost, storage cost and number of exact matches. (2) Extend CKP to multiple languages. Since there are several languages that could not be partitioned by special characters, CKP reveals a way to carry out multiple language keyword partition by utilizing most popular keywords retrieved from queries. (3) Improve load balancing. In the design of JKS, rendezvous peers bear heavy storage cost compared to KAD peers. It is because not only JKS produces more storage cost than KAD, but also all loads are stored in rendezvous peers. It causes extremely imbalanced loading between rendezvous peers and edge peers. A modification may be taken by moving some loads from rendezvous peers to their respective connected edge peers.

Bibliography

[1] N. Nakamura, S. Takahama, L. Barolli, J. Ma, and K. Sugita, "A Multiplatform P2P System: its implementation and applications," in *Proc. of the 19th International Conference on Advanced Information Networking and Applications*, vol. 1, pp. 171-176, 2005.

[2] A. Akram and R. Allan, "Comparison of JXTA and WSRF," in *Proc. of the 7th IEEE International Symposium on Cluster Computing and the Grid*, pp. 761-766, May 2007.

[3] S. Venot and Y. Lu, "On-demand mobile peer-to-peer streaming over the JXTA Overlay," in *Proc. of the International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pp. 131-136, November 2007.

[4] J. E. Riasol and F. Xhafa, "Juxta-Cat: a JXTA-based platform for distributed computing," in *Proc. of the 4th International Symposium on Principles and Practice of Programming in Java Mannheim*, pp. 72-81, 2006.

[5] Y. Liu, "Research on the mobile P2P VOD system of JXME," in *Proc. of the 2007 Annual Conference on International Conference on Computer Engineering and Applications*, pp. 80-83, 2007.

[6] "JXTA community projects," [Online]. Available: https://jxta.dev.java.net.

[7] "JXTA company spotlight," [Online]. Available: https://jxta.dev.java.net/companyarchive.htm.

[8] "SourceForge," [Online]. Available: http://sourceforge.net/search/?type_of_search=soft&type_of_search=soft&words=jxta.

[9] T. Kim, H. Lee, and H. Cheon, "Implementation of a service oriented architecture based on JXTA for new business models (ICCAS 2007)," in *Proc. of the International Conference on Control, Automation and Systems* , pp. 2402-2406, October 2007.

[10] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys & Tutorials*, *IEEE*, vol. 7, pp. 72-93, 2005.

[11] Y.-J Joung, L.-W. Yang, and C.-T. Fang, "Keyword search in DHT-based peer-to-peer networks," *IEEE Journal on Selected Areas in Communications,* vol. 25, pp. 46-61, January 2007.

[12] L. Liu, K. D. Ryu, and K.-W. Lee, "Keyword fusion to support efficient keyword-based search in peer-to-peer file sharing," *IEEE International Symposium on Cluster Computing and the Grid*, pp. 269-276, April 2004.

[13] E. Halepovic, R. Deters, and B. Traversat, "Performance evaluation of JXTA rendezvous," in *Proc. International Symposium on Distributed Objects and Applications*, pp. 1125–1142, October 2004.

[14] G. Antoniu, L. Cudennec, M. Jan, and M. Duigou, "Performance scalability of the JXTA P2P framework," in *Proc. IEEE International Parallel and Distributed Processing Symposium*, pp. 1-10, March 2007.

[15] "JXTA Java Standard Edition v2.5: Programmers Guide," [Online]. Available: https://jxta-guide.dev.java.net.

[16] N. Théodoloz, "DHT-based Routing and Discovery in JXTA," Master's Thesis, School of Computer and Communication Sciences, February 2004.

[17] M. Abdelaziz, B. Traversat, E. Pouyoul, "Project JXTA: A Loosely-Consistent DHT Rendezvous Walker," Mar. 2003. [Online]. Available: http://www.jxta.org/docs/jxta-dht.pdf.

[18] "JXSE CMS," [Online]. Available: https://jxse-cms.dev.java.net.

[19] X. Xiang, Y. Shi, and L. Guo, "Rich metadata searches using the JXTA content manager service," in *Proc. of the 18th International Conference on Advanced Information Networking and Applications*, vol. 1, pp. 624-629, 2004.

[20] R. Brunner, "A performance evaluation of the Kad-protocol," Master's Thesis, University of Mannheim and Institut Eurecom, 2006.

[21] M. Steiner, T. En-Najjary, and E. W. Biersack, "A global view of KAD," in *Proc. of the 7th ACM SIGCOMM Conference on Internet Measurement*, pp.117-122, 2007.

[22] "Wiki KAD," [Online]. Available: http://en.wikipedia.org/wiki/KAD.

[23] "eMule," [Online]. Available: http://www.emule-project.net/home/perl/general.cgi?l=1.

[24] D. Carra and E. W. Biersack, "Building a reliable P2P system out of unreliable P2P clients: the case of KAD," in *Proc. of the ACM CoNEXT Conference*, No. 28, 2007.

[25] "AOL query log," [Online]. Available: http://www.gregsadetsky.com/aol-data.

[26] "Yahoo," [Online]. Available: http://tw.buzz.yahoo.com/live_kw.php.xml.

[27] "Google Soap Search API," [Online]. Available: http://code.google.com/apis/soapsearch.

[28] D. Stutzbach, S. Y. Zhao, and R. Rejaie, "Characterizing files in the modern Gnutella network," *Multimedia Systems*, vol. 13, pp. 35-50, Sep. 2007.

[29] X. Jin, W.-P. K. Yiu, and S.-H. G. Chan, "Supporting multiple-keyword search in a hybrid structured peer-to-peer network," in *Proc. of the IEEE International Conference on Communications*, pp. 42-47, June 2006.

[30] M. Steiner, W. Effelsberg, T. En-Najjary, and E. W. Biersack. "Load reduction in the KAD peer-to-peer system," in Proc. of *5th International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2007.

[31] M. Steiner, T. En-Najjary, and E. W. Biersack, "Exploiting KAD: possible uses and misuses," *ACM SIGCOMM Computer Communnication Review*, vol. 37, pp. 65-70, 2007.

[32] "NOVA," [Online]. Available: http://www.nova.com.tw.