Journal of Theoretical and Applied Information Technology
28th February 2013. Vol. 48 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FLEXIBLE SHARING IN DHT-BASED P2P NETWORKS USING METADATA OF RESOURCE

[1]GUO-YING WANG, [2]ZHENG HUA

[1]Information Engineering College, Zhejiang A&F University, Hangzhou, 311300, China

[2] School of Information and Statistics, Guangxi University of Finance and Economics, 530003, China

E-mail: [1]wgy@zafu.edu.cn , [2]gxhuazheng@yahoo.com.cn

ABSTRACT

DHT-based P2P systems have the ability to map a resource's identifier to its location, so a resource can be located by the unique identifier of the resource. In such systems users must know the identifier of a resource exactly to find the resource. However, a resource can not be described accurately and integrally only by an identifier for the variety and complexity of resources, and needs multiple properties, i.e. metadata, are needed. In this paper, we proposed a flexible sharing method for DHT-based P2P networks using the metadata of resources. In this method, inverted indexes are constructed with multiple properties of resources, hashed and placed into proper nodes based on DHT when publishing resources; multiple properties are given by users and hashed to corresponding nodes using DHT, then matched inverted indexes are collected together and joined when searching for resources. Some extra technologies such as synonyms library, combined properties and virtual nodes are also used in our method to improve the sharing flexibility and performance. We also implemented a prototype system of our sharing method, *MP3Share*, using which we evaluated the performance improvement of combined properties. To improve the efficiency of the whole system, we should using synonyms library and combined properties technologies during the publishing procedure other than the searching procedure.

Keywords: *Resource Sharing, Metadata, Distributed Hash Table (DHT), Peer-to-Peer (P2P)*

1. INTRODUCTION

Peer-to-Peer (P2P) technique has received great attention along with the development of network technology and the increasing amount of Internet users. In recent years, P2P technology has been applied to plenty of applications ranging from file sharing system to video streaming, distributed computing, communication and cooperation. All nodes in P2P networks share their storage space, CPU time and bandwidth, which is different from client/server systems. A P2P system has more effective communication because nodes in it interact with each other directly without the need of a central server as the medium. The capacity of server is a bottleneck of the whole performance of a client/server system, while a P2P system has not such bottleneck because it does not rely on any central server, which increases the extensibility and the reliability of the system.

Distributed hash table (DHT) is used for the observation and localization of resources in some structured P2P systems, such as Chord [1], CAN [2], Tapestry [3], which is called Chimera [4] now, and Pastry [5]. The identifiers of shared resources are mapped into a hash space in such systems, and the nodes are also hashed to the same space base on IP address. As a result, a mapping relation between resources and nodes is formed, according to which each node is responsible for some resources. Each node also stores some information about its neighbor nodes in the hash space for the purpose of message routing. Given the identifier of a resource, the responsible node can be located rapidly and precisely by hashing the resource identifier. The amount of information stored in each node for routing is only relative to the count of neighbor nodes in the hash spaces. Message exchange only occurred between neighbor nodes. Therefore, the DHT-based P2P system solves the problems of system scalability and the dynamic nature of nodes.

The identifier of the resource must be given to find a resource in a pure DHT-based system. However such strict limit may make it failed for users to get the desired resource in certain condition because of the increasing of the kind and amount of resources in networks. For example, when the songs

Journal of Theoretical and Applied Information Technology
28th February 2013. Vol. 48 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of a certain singer is wanted by a user, the user will no have access to these songs if he don't know all the song titles exactly, not to mention many songs in the network are not stored using titles. Even though all the song titles of the singer are known, it is a little ridiculous for the user to search for each of them by its title. On the other hand, a simple identifier is not sufficient and multiple properties are often needed to describe a resource completely and accurately because of the diversity and complexity of resources.

Therefore, a flexible sharing method for DHT-based P2P networks is proposed in this paper. In our method resources are published and searched on the basis of multiple properties of resources. A user can get to all resources matched search criteria as long as one or more properties of wanted resources are given, regardless of whether the user is aware of the identifiers of resources, regardless of the format of resources. This makes the types of resources that our method can publish and search broader and more flexible [6]. For example, to get all songs of a certain singer, a user needs not know the exact titles of every songs of the singer or the file formats of songs, and need only provide the singer's name.

2. MAIN IDEA

2.1. Architecture

Along with the decreasing of storage price and increasing of storage space, even a small group of users could share large amounts of data, so accurate and fast search capabilities is particularly important for the designing of a P2P system. Using the capability of DHT to map a key to a network node, a metadata-based inverted index module was designed on the basis of DHT, which implemented the publishing and search functions, as is shown in figure 1. Furthermore, some technologies such as synonyms library, combined properties and virtual nodes are used to improve the functionalities and performance.

Traditional search engines often use pre-generated inverted indexes to shorten the search time. An inverted index is a list of correspondences between each keyword and the identifier of a document containing the keyword, i.e., <keyword, document>. We import this idea into our flexible sharing method. For each resource described with multiple properties, we define the inverted indexes as the correspondences between each property and the identifier of a resource, i.e., <property, resource>.

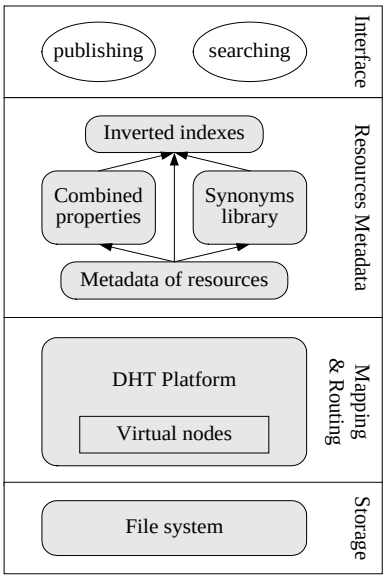

Figure 1. Architecture Of The Proposed Sharing Method For DHT-Based P2P Networks

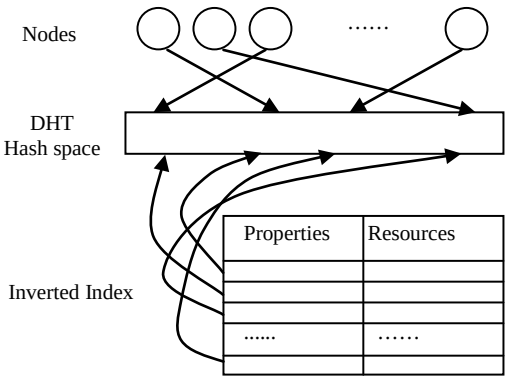

Figure 2. Mapping Between Nodes And Properties Inverted Indexes Of Resources Using DHT

As to the retrieval of inverted indexes, traditional search engines often search for target documents in a centralized way. While inverted indexes can be stored and retrieved in a distributed way for P2P systems, which can speed up the searching and reduce the pressure of individual nodes.

Although the existing DHT-based P2P systems have not search abilities but a mapping relation, DHT technology provides a mechanism of mapping a single key to a node in the network. Using the mapping ability of DHT, we can map and store the inverted indexes of each property of the resource in the corresponding nodes when a resource is published, and route the search requests of given properties to responsible nodes when searching. The corresponding relations between properties of resources and nodes are shown in figure 2.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In our method, when publishing a resource, not only the identifier of the resource but also the properties of it are hashed. Inverted indexes are constructed using these properties, and then these inverted indexes are sent to and stored in responsible nodes using the DHT layer. When searching a document, the user provides one or more property values and these property values are hashed respectively. Then the corresponding nodes stored each keyword inverted index are located using the hash results on the basis of the DHT layer. All resources met the user's request can be found in the inverted index of these properties.

2.2. Construction of Inverted Index

For a resource described with multiple properties, the values of properties are main factors of the resource. For example, some property values of a song such as singer, album, lyricist, and composer can be used to describe the song well. A property value may stand for different relation to a resource if its relative property is different, and different properties may have the same value. Therefore, if these property values are used indiscriminately to construct inverted indexes, some search results may deviate from the users' requirement. For example, when a user wants to search for songs singed by a star, the search result may include songs not singed but composed by the star.

Therefore when constructing inverted indexes in our method, not only the values of properties are used, but also the combination of names and values of properties are used. Using this method we can distinguish different properties with the same values. Suppose resource R has n properties P_1, P_2, ..., P_n, and the corresponding property values are V_1, V_2, ..., V_n, respectively. Then the inverted index is constructed like the following correspondence $<P_i, V_i> : R$ (i = 1, 2, … , n) rather than the correspondence between V_i (i = 1, 2, … , n) and R.

3. FLEXIBLE SHARING ALGORITHMS

3.1. Description of Algorithms

When publishing a resource, every properties of the resource are used to construct inverted index entries, which mapping each property to the resource identifier. Then each entry was placed to the corresponding node using DHT layer according to the hash result of the property in each entry, which is shown as algorithm 1.

For example, when publishing the resource R, at first each property pair $<P_i, V_i>$ (i = 1, 2, … , n) was hashed and the hash result H_i was obtained. And then inverted index entries $<H_i : R>$ (i = 1, 2, … , n) was constructed and placed to N_i (i = 1, 2, … , n). Suppose N_i is the corresponding node of H_i based on DHT mapping shown in figure 2.

When searching for resources, users must provide one or more property name-value pairs, and then each pair is hashed and the result is used to find the corresponding node storing the inverted index entry of the property pair based on DHT shown in figure 2. Finally the inverted index entries of all provided properties are collected and some *JOIN* operations such as union and/or intersection were performed. The result is all resources matched given criteria. The algorithm is show as algorithm 2.

If resources having value V_1 for the property P_1 and value V_2 for the property P_2 are wanted by a user, first, $<P_1, V_1>$ and $<P_2, V_2>$ are hashed and the result is H_1 and H_2 respectively. Then node N_1 and N_2, the corresponding node of the hash result H_1 and H_2 according to figure 2 are located and searched for matched inverted index entries, and interjection of the search results S_1 and S_2 was send to the user. For less exchange, the node doing *JOIN* operation is one of N_1 and N_2 in which more matched entries have been found than in the other.

Algorithm 1: Basic_Flexible_Publishing

```
Input:
    RID – Resource identifier;
    n – Number of properties in metadata of resource;
    P[] – Array of property names;
    V[] – Array of properties values
Begin
  for i:=1 to n do
    PV:=Combine(P[i],V[i]);
    H:=Hash(PV);
    Node:=LookUpNode(H);
    Insert(Node,H,RID);
  end for;
End.
```

Algorithm 2: Basic_Flexible_Searching

```
Input:
    x –Number of property values use provided;
    P[] – Array of property names;
    V[] – Array of properties values
Begin
  for i:=1 to x do
    PV:=Combine(P[i],V[i]);
    H:=Hash(PV);
    Node[i]:=LookUpNode(H);
    S[i]:=LookUpRIDs(Node[i],H);
    Send(Node[i],Node[join],S[i]);
  end for;
  S[join]:=Join(N,S[]);
  return(S[join]);
End.
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.2. Analysis of Algorithms

The process of publishing a resource in pure DHT-based P2P system is shown as algorithm 3. This algorithm hashes the resource identifier, finds the corresponding node, and finally stores the hash result and resource identifier in the node.

Algorithm 1 repeats the process in algorithm 3 for n times so as to hash n properties and place n hash results to n corresponding nodes. So the bandwidth needed to publish a resource in our method is n times of that in a pure DHT system. The time complexity of publishing a resource needs in Tapestry is $O(logN)$ [3], so that of our method is $O(n\ logN)$, and the n means the count of properties of the published resource and N means the count of nodes in the system. The CPU time and bandwidth needed for publishing resources in our method are more than that in DHT systems. We can use n processes or threads in parallel to shorten the time, and each process or thread publishes one property, because of the independency of publishing all properties. In this paralleled way, publishing a resource needs the same time to Tapestry, $O(logN)$.

Algorithm 4 is the resource searching algorithm in DHT-based P2P system. First the identifier of wanted resource given by a user is hashed and then the corresponding node is looked up for matched identifier.

Compared with algorithm 4, the algorithm 2 not only repeated these operation x times, but also collected x sets of matched inverted index entries to a node and computed the *JOIN* result. Here x means the count of properties given by user as criteria. In Tapestry routing from a node to another needs time $O(logN)$, so sending matched entries from one nodes ($node[i]$) to another one ($node[join]$) needs the same time. As a result, the time complexity of searching for a resource in our method is $O(x(logN + logN)) = O(x\ logN)$, and x means the count of properties provided by a user as the search criteria and N means the count of nodes in the system. Similarly, hashing each property of search criteria, looking up each node and collecting matched inverted index entries can also be executed in parallel so as to shorten the search time.

Algorithm 3: DHT_Publishing

```
Input:
    RID – Resource identifier;
Begin
  H:=Hash(RID);
  Node:=LookUpNode(H);
  Insert(Node,H,RID);
End.
```

Algorithm 4: DHT_Searching

```
Input:
    RID – Resource identifier;
Begin
  H:=Hash(RID);
  Node:=LookUpNode(H);
  S:=LookUpRIDs(Node,H);
  return(S);
End.
```

4. ENHANCEMENT OF SHARING

To make the proposed method more feasible to practice, some key strategies are needed to improve the functionality and the performance: (1) metadata of a resource should be created and formatted by user when the resource is published; (2) considering the similarity between some property values, a library of synonyms should be constructed in advance and used during the publishing and searching procedure, which can improve the semantic performance; (3) combined multiple properties should be used when the multiple properties are often used together during publishing or searching procedures, which can relieve the load of transferring inverted indexes and doing *JOIN* operation between inverted indexes; and (4) considering the imbalance of capability of nodes and size of inverted indexes, some load balancing technologies should be taken.

4.1. Acquisition of Metadata

Every resource in networks can be described by some properties, but these properties do not exist in written form with resource usually. Therefore, a resource has to be described, i.e., the value of each properties of the resource is provided, by its publisher when it is published. According to metadata provided by the publisher, the inverted index of each property is published to the corresponding node for other users to search. The more detailed the metadata provided by the publisher is, the more ways can be used to find the resources by other users. The more accurate the metadata is, the better the search results match the requests of users.

Even so, these metadata are not saved. Once the users withdraw a shared resource and want to re-publish it, they have to describe the resource once again. On the other hand, if the saved metadata information has not been published along with the resource, other users have to re-describe it when they obtain the resource and want to share it too. Besides, there may be large or small differences between the original metadata and new description, which will affect the access to the resource. Although the metadata describing resources can be saved and published with the resource together as a

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

supplement, the preservation and publishing of metadata will undoubtedly increase the system operation and maintenance costs.

If the metadata can be embedded in the resource and the creator of the resource fill in property values one-time when the resource is created, the metadata of the resource can be published or accessed together with the resource. Therefore, an additional processing overhead is no longer needed. There is no need for publisher to describe a resource once again, and what the publisher needs to do is to read metadata from the resource, which is written by the creator. The method will also make description of the same type of the resource more standard and uniform. For example, the most popular music file format in network, MP3, can be embedded with song title, singer, album and other information in the file as ID3 tags.

4.2. Semantics of Inverted Indexes

Method described above adopts exact match of properties to achieve the mapping from properties to the location of inverted indexes. However, synonyms may lead to malfunction in practice. If a resource is published with a specific word *W* as a property value, the resource will not be found when other users search resources with a word *W'* which have the same meaning with *W*. For example, if a song is published using the name of a singer in English, while the user search it using Chinese, the song will not be found even if it exists in the system and matches the user's criteria semantically. Therefore a synonyms library should be maintained to improve the semantics.

The synonyms library can be used in either publishing procedure or searching procedure. In the first case, when constructing an inverted index of a description as a property value during a resource-publishing procedure, the publisher program looks up the synonyms library, constructs the inverted indexes of other synonyms of the description and publishes them. Then the resource can be found during searching procedure afterward, no matter which description in the synonyms library is provided. In the other case, even if a user publishes a resource without using the synonyms library, other users can also find the resource with other keywords during search procedure, if the searcher program looks up the synonyms library and search with all synonyms of the keyword user provided.

Using synonyms library may lead a longer process time. However once a resource is published, there may be a great number of users who will search and access the resource at some time later. That is, a publishing procedure corresponds to many searching procedures. From this point of view, we should use the synonyms library in the publishing procedure other than the searching procedure. The cost of using synonyms library during publishing procedure is counterbalanced by the good retrievability performance of the large amount of searching procedure afterwards.

4.3. Sharing with Combined Properties

In a searching procedure with multiple properties, the inverted indexes corresponding to each property are collected to a node and execute some *JOIN* operations. Obviously, the system should send the fewer inverted index entries to the node where the more inverted index entries locate in.

To improve the efficiency of searching procedure, the transmission and computation overhead of inverted indexes sets should be reduced. There are several ways to reach the objective: first, the necessity to transfer inverted indexes should be reduced, which can be achieved by pre-computing or caching relevant information; second, the inverted indexes sets should be compressed to reduce the network communication required; third, in the premise of meeting users' requirement, only part of the inverted indexes are transferred. In our method, we emphasized on the first method, i.e., combined keywords.

If we combine several keywords together and hash them to place the corresponding inverted indexes to a node in network during the publishing procedure, the inverted indexes of each keyword need not transfer any more when a user searches a file which contains these keywords. Instead, the combination of pre-built inverted indexes can be returned to the user directly.

Publishing a resource with combined properties may lead to an increase of burden of publishing, for not only the inverted index of each single property but also the inverted indexes of combined properties need be published. However, once a resource is published, there may be a great number of users who will search and access the resource at some time later. Using combined properties can reduce the amount of inverted indexes transferred and the computation of *JOIN* operation during the searching procedure, which can save the expense of communication and computation.

To take advantage of the convenience of combined properties in the large amount of search

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

procedures and avoid too much overload in the publishing procedure, there should be a trade-off. Combined properties should be published or not according to the potential probability that the properties to combine are used together: combined properties that may be used frequently in search procedures should be published, while seldom used combined properties should not be published.

4.4. Load Balancing

A key property of P2P systems is heterogeneous, and the capacities of nodes are different. Therefore, the designer of P2P networks needs to consider load balancing. In our method, inverted indexes of some properties may be larger than that of the others. For example, the inverted index of a popular star is much larger than the inverted index of a single album. On one hand, a large inverted index properties may be hashed to a node with relatively poor performance (storage space, speed of CPU computation, network bandwidth etc.). On the other hand, due to the character of hash function, the distribution of hash values of properties may be unevenly. Hence, some nodes may be assigned too more hash value of the property than the average. As a result, these nodes may be heavily loaded and become bottlenecks of whole system.

Virtual nodes technique can be used to alleviate imbalance to some extent. Based on the capacities of nodes and the real time load, a number of logical virtual nodes are set up in a physical node dynamically. It makes work loads of nodes relative to the capacities, and reduces the blindness of the distribution of inverted indexes.

4.5. Enhanced Sharing Algorithms

Integrated these enhancing approaches into the basic flexible sharing algorithms, we reach the enhanced sharing algorithms. Algorithm 5 shows the enhanced publishing algorithms, in which metadata are filled if it is necessary, and synonyms library and combine properties are used when the options are turned on.

Load balancing using virtual nodes is usually accomplished in the deploying stage other than by the algorithm, so this technology is not included in algorithm 5.

It is easy to understand that metadata of resources should be filled during the publishing procedure other than searching procedure. As to the synonyms library and combined properties, we preferred to use them during the publishing procedure other than the searching procedure, for that there may be many searching procedure by different users after a publishing procedure for a resource. This means that using these two technologies in publishing procedure can obtain more efficiency. For these reasons, we do not describe the enhanced flexible searching algorithm any more, in which the way of using the two technologies is similar to that in algorithm 5.

Algorithm 5: Enhanced_Flexible_Publishing

```
Input:
    RID – Resource identifier;
    n – Number of properties in metadata of resource;
    P[] – Array of property names;
    V[] – Array of properties values
Begin
  num := n;
  for i:=1 to n do
    // Fill metadata if it is necessary.
    if V[i] = NULL then
      Prompt user to fill V[i];
    end if;
    // Using synonyms library
    if synonyms_library = true then
      S[] = LookUpSynLib(V[i]);
      for k:=1 to length(S[]) do
        P[++num] = P[i];
        V[num] = S[k];
      end for
    end if
  end for;
  // Using combined properties
  if combine_properties = true then
    CPlist[] = LookUpComProps();
    CVlist[] = GetValues(CPlist[]);
    for k:=1 to length(CPList[]) do
      P[++num] = Combine(CPlist[k]);
      V[num] = Combine(CVlist[k]);
    end for
  end if

  Basic_Flexible_Publishing(RID, num, P[], V[]);
End.
```

5. PROTOTYPE AND EVALUATION

5.1. Prototype System

We implemented a prototype system, *MP3Share*, on the basis of Tapestry [3] which is a development platform for DHT-based p2p systems. *MP3Share* provided the functionalities of synonyms library, combined properties and virtual nodes, as well as the interface for users to fill the metadata of mp3 files.

We adopted six common properties of songs as the metadata for publishing and searching song files, such as title, album, singer, year, composer and lyricist. We formatted the metadata using ID3 tags of mp3 files of randomly chosen 250 songs.

In *MP3Share*, we construct a synonyms library of singer name including some keyword sets: each

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

keyword set means a singer and each element of a set means a different expression of the singer's name. For example, a keyword set is {"Madonna", "Madonna Louise Ciccone"}, another keyword set is {"bsb", "Backstreet Boys"} and so on. A more useful case is that each keyword set includes the names of the singer in different language such as English, Chinese and so on.

Considering that songs in an album of a singer are usually wanted together, we use combined properties of <singer, album> in *MP3Share*.

When a user publish a song mp3 file with *MP3Share*, the program reads the ID3 tags of the mp3 file, constructs the inverted indexes with the six key-value tuples mentioned above, constructs the inverted indexes with the synonymous words of the singer's name in the synonyms library, constructs the inverted index with the combined properties of singer and album of the song, and finally hash each of these inverted indexes to get a hash value according to which a proper node is determined to save one copy of the inverted index.

The virtual nodes functionality is supported by the platform, Tapestry. The number of virtual nodes that one real node provided can be set in the configure file of Tapestry.

5.2. Experimental Evaluation

In our experiment, we deployed *MP3Share* into three PCs in a LAN. To simulate the complexity of reality, 5, 5, 8 virtual nodes are configured to run in the three PCs respectively.

The basic publishing and searching capability are main aspects for a resource-sharing system to test. We also tested the additional functionalities such as synonyms library, virtual nodes and combined properties. Synonyms library and virtual nodes mainly enhance the usability of the system, while combined properties is used to improve the performance of the system. So we examine the time costs of publishing and searching resources with *MP3Share*, mainly for the using of combined properties functionality.

We published the mp3 files of 250 songs using *MP3Share* with combined properties and without them for 10 times respectively. When not using combined properties, we need publish 6 inverted indexes of the 6 properties for each song. While we need publish 7 inverted indexes of the 6 properties and the <singer, album> combined property of each song when using combined properties. We examined the time cost both using the <singer, album> combined property and not using it, which is shown in figure 3. The average time cost of publishing without combined properties is 4639.2 milliseconds, and that of publishing with combined property is 5536.8 milliseconds. The ratio of these two time costs is 0.8379 which is close to 6/7, that is, the time cost of publishing is in proportion to the count of inverted indexes published approximately.

After publishing with combined properties, we look for songs of the Madonna's album "Ray Of Light" using *MP3Share* with the <singer, album> combined property and without it for 10 times respectively. The searching time costs are shown in figure 4. The average cost of searching without combined properties is 13.12 milliseconds, and that of searching with combined properties is 9.03 milliseconds. This means that using combined properties saves 31.1% processing time, including saving of communication time of inverted indexes and computation time of the *JOIN* operations.

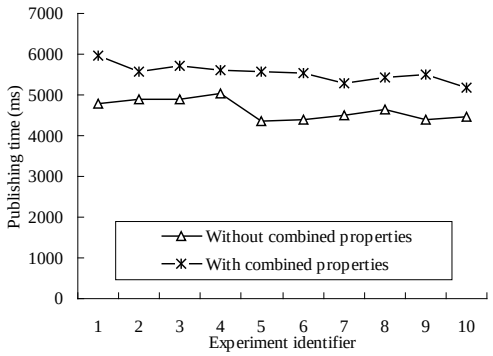

Figure 3. Publishing Time Of 250 MP3 Song Files With And Without Combined Properties

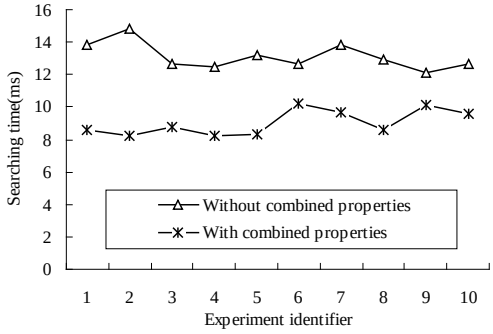

Figure 4. Searching Time For Madonna's Album "Ray Of Light" With And Without Combined Properties

From figure 3 we can see that each combined property can lead to additional publishing time cost, while on the other hand it can also lead to a saving of searching time cost according to figure 4. Because the relation between publishing procedure and search procedures of a resource is 1 : *n*, the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

benefit of combined properties is in a dominant position, if we using this technology in each publishing procedure.

6. RELATED WORKS

6.1 Semantics in P2P Networks

Huang extended GES and presented a class-based semantic searching scheme (CSS) in Gnutella-like unstructured P2P systems, and designed a totally class-based protocol and a partially class-based search protocol, both of which are more efficient than GES in all cases [7]. Rudomilov studied P2P unstructured information retrieval (IR) with heterogeneous documents on independent nodes, and proposed a combination of CSS and GES models as semantic Gnutella-like Information Retrieval system using class-based approach [8].

Cerqueus focused on semantic heterogeneity and interoperability in unstructured P2P information retrieval systems, and proposed two protocols: the GoOD-TA protocol that reduces the semantic heterogeneity related to the topology by putting closer peers that can understand each others and the DiQuESH protocol, a distributed top-k algorithm that ensures some interoperability [9]. Cao Presented a dynamic semantic data replication scheme called DSDR for classic k-random search in unstructured peer-to-peer networks. During its k-random search each peer periodically updates its local view on the semantic overlay of the network based on observed queries (demand) and received information about provided items (supply), in particular their semantics [10].

6.2 Multiple Keywords Searching

When searching resources using multiple keywords, it needs to collect the set meeting each keyword and compute the *JOIN* of these keywords sets. There are three ways to improve the efficiency, i.e., pre-computed *JOIN* result of keyword sets, transfer top-k the each keyword set, and compressing the keyword sets that have to be transferred.

Gnawali research the searching methods in peer-to-peer networks using pre-computed keyword set [11], which is similar to the method of combined properties in our method.

Users usually do not need to get all the results which match the search criteria at a time. So we can transfer parts of the inverted indexes in batches, produce a part of the results, and return them to users, then stop or continue to transfer the inverted indexes to generate new results according to the need of users. The method will not only reduce the transmission cost, but also shorten the response time of searches.

Researchers have studied that different compression methods, such as bloom filter [12], gap compression [13], PLSA compression [13], and so on, are used to reduce the communication overhead required for the keyword sets.

6.3 Load Balancing

Virtual nodes technique is a common way to alleviate the load unbalancing to some extend. In addition, Surana [14] discussed in more detail about load balancing issue in the P2P system, found that heterogeneity of the system can improve scalability by reducing the necessary number of virtual servers per node as compared to a system in which all nodes have the same capacity.

7. CONCLUSIONS

In the paper, we proposed a flexible resource sharing method for DHT-based P2P networks. Based on the capability of DHT mapping a single key to a node, our method has good scalability and dynamicity that DHT-based P2P systems possessed. In addition, unlike pure DHT-based P2P systems, which have only mapping function other than search ability, our method has rich capabilities of searching for resources providing the property conditions. It can meet various search requests of users. The function of our method only depends on the various properties describing resources and has nothing to do with the identifier, format of resources. Therefore, the resource type that our method can publish and search are widely scaled and flexible. Besides, synonyms library technology is used in the system to enhance the search semantics. Furthermore, combined properties technology is adopted to improve the efficiency of searching with multiple properties. At last, virtual nodes technology is used to improve system's load balancing.

ACKNOWLEDGMENT

The work of this paper is supported by Zhejiang Provincial Natural Science Foundation of China (LY12F02016) and the 2011 Maintain School Safety and Stability Fund Project of Guangxi province (class A).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES:

[1] Ion Stoica, Robert Morris, David Liben-Nowell, etc. "*Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications*". IEEE/ACM Transactions on Networking, Vol. 11, No. 1, Feb. 2003, pp. 17-32.

[2] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. "*A scalable content-addressable network*". Proc. ACM SIGCOMM'01, 2001, pp. 161-172.

[3] Ben Y. Zhao, Ling Huang, Jeremy Stribling, etc. "*Tapestry: A Resilient Global-scale Overlay for Service Deployment*". IEEE Journal on Selected Areas in Communications, January 2004, Vol. 22, No. 1, Pgs. 41-53.

[4] http://current.cs.ucsb.edu/projects/chimera/, visited in Oct. 2012.

[5] Antony Rowstron, Peter Druschel. "*Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems*". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, Nov. 2001, pp. 329-350.

[6] G. Wang, J. Cheng, "*Design of a Semantic Resources Sharing Framework for Structured P2P Networks*", in Proceeding of ISCID'10, 2010, Vol. 2, pp. 201-204.

[7] J. Huang, X. Li, J. Wu, "*A semantic searching scheme in heterogeneous unstructured P2P networks*", Journal of Computer Science and Technology, 2011, vol. 26, No. 6, pp. 925–941.

[8] I. Rudomilov, I. Jelínek, "*Class-based Approach in Semantic P2P Information Retrieval*", in Proceeding of Federated Conference on Computer Science and Information Systems, 2012, pp. 279–283

[9] X.Cao and M. Klusch, "*Dynamic Semantic Data Replication for K-Random Search in Peer-to-Peer Networks*", in Proceeding of IEEE 11th International Symposium on Network Computing and Applications, 2012, pp. 20-27.

[10] T. Cerqueus, S. Cazalens, P. Lamarre, "*An Approach to Manage Semantic Heterogeneity in Unstructured P2P Information Retrieval Systems*", in Proceeding of IEEE International Conference on Peer-to-Peer Computing, 2012.

[11] O. D. Gnawali. "*A Keyword Set Search System for Peer-to-Peer Networks*". Master's Thesis, Massachusetts Institute of Technology, Jun. 2002.

[12] P. Reynolds and A. Vahdat. "*Efficient peer-to-peer keyword searching*". Proc.Middleware 2003, Jun. 2003, pp. 21-40.

[13] J. Li, B. T. Loo, J. M. Hellerstein, and etc. "*On the Feasibility of Peer-to-Peer Web Indexing and Search*". Proc. 2nd International Workshop on Peer-to-Peer Systems, Feb. 2003, pp.207-215.

[14] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, I. Stoica. "*Load balancing in dynamic structured peer-to-peer systems*". Performance Evaluation In P2P Computing Systems, Vol. 63, March 2006, pp. 217-240.