An Efficient Interest-Group Based Search Mechanism in Unstructured Peer-to-Peer Networks

Jian Yang Yiping Zhong Shiyong Zhang
Department of Computing and Information Technology, Fudan University, Shanghai 200433, P.R. China
jianyang@guanghua.sh.cn

Abstract

Efficient search mechanism is a very important aspect of any peer-to-peer system. Current search mechanisms in both DHT-based structured peer-to-peer networks and unstructured peer-to-peer networks are either lack of flexibility or very inefficient. In this paper, we propose an efficient search mechanism based on interest-group, which is formed by nodes sharing similar interests. Metadata are used to describe and represent documents that nodes share with others. We deploy Metadata Slide Window and metadata replication to facilitate the search process. Preliminary analysis and simulation results prove the effectiveness and efficiency of the proposed search mechanism.

1. Introduction

With the rapid growth of Internet and computing power, peer-to-peer (P2P) networks have already gained huge success in file-sharing applications. However, the potential of P2P networks can reach far more than just file sharing. It is even predicted that P2P will shape the future of Internet. As for any heavily used large distributed systems, the effectiveness of a P2P network largely depends on the versatility and scalability of its search mechanism.

P2P networks became popular with the advent of Napster [1], a centralized architecture, where the shared items of all peers were indexed in a central server. Although queries in this centralized architecture are very efficient and versatile, the major problems are security and scalability.

There are also kinds of decentralized architectures of P2P networks. They can be roughly classified into two categories: structured and unstructured. For structured P2P networks, although they don't have a central server, they do have a significant amount of structure. Structured means the topology of the network is tightly controlled and files are not randomly placed at nodes but at specific locations which will make subsequent queries easier to satisfy. These systems usually deploy a kind of distributed hash table (DHT) scheme. Typical systems include [2] [3] [4] [5], etc..

Decentralized unstructured P2P networks have neither a central directory server nor any precise control over the network topology or file placement. Gnutella [6] is a typical example which is formed by nodes joining the network following some loose rules. Queries in such systems are usually forwarded to random neighbors.

2. Related Work

There are various search strategies developed for both structured and unstructured decentralized P2P systems. DHT-based P2P networks are suitable for exact-match queries but difficult to satisfy complex queries. However, some search mechanisms are designed to support complex queries in DHT based P2P networks, e.g. [7] [8]. These schemes can solve the complex query problem in DHT based P2P networks to some extent, but it is still unsatisfactory.

For Gnutella-like unstructured P2P networks, the basic search mechanism is flooding which may find desirable results at the cost of large amount of network traffic. To address the scalability issue, some improvements are made to the flooding scheme. Approaches in [9] and [10] try to resolve the problem by studying the topology of the network. And [11] and [12] exploit space for improvements over pure flooding, such as refining query stop condition and replacing flooding with random walk.

In this paper, we propose a search mechanism that is based on the idea that nodes sharing similar interests always store similar documents. Nodes sharing similar interests form an interest-group and queries are propagated within the interest-group first. Our search mechanism does not exert tight control over network topology like DHT-based P2P networks do. While obtaining high search efficiency, our search mechanism also retains many appealing properties of unstructured P2P networks such as support for complex query and resilience to peer failures.

In the next section, we describe the representation of document. The basic model and procedure of our search

Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing (ICCNMC'03)
0-7695-2033-2/03 $17.00 © 2003 IEEE

mechanism are presented in detail in section 4. We give and explain the simulation results in section 5. And we draw conclusions with future directions in the last section.

3. Representation of Document

Metadata can be simply defined as data about data. Metadata are used to describe the documents they represent and are checked to match given queries. There are a number of types of metadata. Document hash, for example, is an id generated from the document contents through some hashing algorithm and often used by DHT-based P2P networks. Statistical representation takes advantage of statistical information of document, e.g. TFIDF [13]. RDF and other keywords schemes are a kind of human assigned metadata.

We choose Resource Description Framework (RDF) [14] as basic tool for describing documents. There are various types of metadata. The reason we choose RDF is its capability to precisely describe documents and support for complex statements. RDF is a framework for describing and exchanging metadata. RDF descriptions are usually exchanged using the syntax of XML. Typically, one RDF statement consists of three parts, namely Resource, Property and Value, or Subject, Predicate and Object. A resource may have many properties and each property may have a different value.

In our proposed metadata scheme, a document is described by many of its properties, such as title, author, style, performer, creation time, version number, keywords, and corresponding document location, etc.. The more detailed description a metadata contains; the more precise result a query returns. Metadata is initially stored in the same node as that storing the corresponding document. One metadata corresponds to one document. However, metadata can be replicated among interest-group to boost the search process because of its much smaller size than that of document. We discuss it in detail in the next section.

4. Interest-Group Based Search Model

In this section, we present our interest-group based search mechanism. First, we give a description of the basic model. And then we present the Metadata Slide Window mechanism deployed in our search model. After that, we describe the detailed search procedure and finally we discuss the bootstrap procedure of the search model.

4.1. Basic Model

As pointed out by [15]: Data in the modern society are not placed at random. Nodes in a P2P network can fall into different groups according to their interests in documents. Because document can be described by metadata, interest-group in our search mechanism is based on metadata. Different nodes sharing the same metadata form an interest-group. Note that one node may belong to several interest-groups. The reason we choose metadata as basis of interest-group is that:

1). Metadata using RDF may have very affluent descriptions about document, which makes query in interest-group can be complex and semantically rich.

2). The size of metadata is usually very small, so metadata can be replicated among the interest-group. One node can store much more metadata than documents; it can retrieve a document directly from another node by using the document location information stored in the corresponding metadata.

The basic hypothesis is that queries can be satisfied with high probability by nodes sharing similar interests. If one node can satisfy a previous query, it is more likely that the node will satisfy a subsequent query. During some period, the node which satisfied more previous queries than other nodes also has more chances to satisfy subsequent queries.

We assume there are totally n nodes and m metadata (duplicated metadata are regarded as one) in the network. N denotes the set of nodes and M denotes the set of metadata in the network. $|N| = n$ and $|M| = m$. We denote by L the set of (node, metadata) pairs. Node can be represented by its IP address and metadata is represented by its network-unique ID in L. For simplicity purpose, we still use the terms of node and metadata when referring to L. Suppose that node $n_i \in N$ and metadata $m_r \in M$, if node n_i contains metadata m_r, then L contains (n_i, m_r) pair. We denote by C_r the interest-group that is formed by nodes sharing metadata m_r.

L_i denotes the set of (node, metadata) pairs that node n_i maintains. L_i is a subset of L. For example, if node n_i learns that $(n_k, m_t) \in L$, node n_i will insert (n_k, m_t) pair into L_i. Each node also has a metadata repository. The set of metadata stored in the metadata repository of node n_i is denoted by M_i. M_i initially stores the metadata that node n_i owns when it just joins the network. And L_i initially contains (n_i, m_r) pairs, where $m_r \in M_i$. With the process of query, M_i and L_i are merged with the same sets retrieved from other nodes. We also denote by N_{ir} the subset of nodes having metadata m_r that node n_i knows. N_{ir} can be expressed by $N_{ir} = \{n \mid (n, m_r) \in L_i\}$.

4.2. Metadata Slide Window

To reflect the dynamic feature of unstructured P2P networks and similar interests within a interest-group, we deploy a mechanism called Metadata Slide Window (MSW). As we have noted above, queries can be satisfied with high probability in the same interest-group as the query initiator belongs to. Nodes in the same interest-group share similar interests, which indicate they have similar metadata. Thus metadata can be used as a clue to find other nodes in the interest-group. If one node in a interest-group issues a query, it first asks the node which has some same metadata as it does. The query initiator may have many metadata, so it needs to select one which is more effective than others. MSW serves this purpose. We denote by |MSW| the size of the window and the value of |MSW| is adjustable. A larger MSW has more accurate prediction about which metadata is the most effective.

MSW is initially empty. With the process of query, MSW is gradually filled with metadata. The metadata which is used as a clue to complete a query successfully is inserted into the leftmost slot in MSW. And the metadata before the insertion in all slots slide one slot right. The metadata in the rightmost slot is dropped from the window.

One metadata occupies one or more slots in MSW. Different slot in MSW has different weight. Weight is used to calculate which metadata is to choose as a clue for next query. The leftmost slot has the heaviest weight |MSW|, the slot next to it has the second-heaviest weight |MSW|-1, etc., and the rightmost slot has the lightest weight 1. When one node needs to select some metadata in its repository to decide which node to forward the query to, it checks its MSW. The sum of weights of the slots occupied by same metadata in MSW is calculated and the metadata with the heaviest sum of weights is selected.

8	7	6	5	4	3	2	1
m_t	m_r	m_s	m_r	m_r	m_s	m_t	m_t

Figure 1. Example MSW of Node n_i

Figure 1 depicts an example MSW of node n_i with the window size of 8; values above the window slots are corresponding weights. In the example, there are 3 different metadata in MSW, m_t occupies slot 8, 2, 1 in MSW and the sum of weights is 11, m_r occupies slot 7, 5, 4 and the sum of weights is 16, and m_s occupies slot 6 and 3 and the sum of weights is 9. We can see metadata m_r has the heaviest sum of weights in MSW, so m_r is selected and N_{ir} is calculated. The query is then forwarded to a randomly chosen node n_j in the set of N_{ir}.

4.3. Search Procedure

Suppose node n_i needs some document which is described by metadata m_r. By using the basic model and Metadata Slide Window described above, the procedure of our search mechanism is as following:

1). Node n_i first matches the query against its local metadata repository M_i. If metadata m_r can be found locally, n_i completes the query successfully and goes to step 8); if not, n_i goes to step 2).

2). Node n_i checks its MSW. If MSW is empty, it chooses one metadata from its metadata repository M_i randomly; if MSW is not empty, the metadata in MSW with the heaviest or next heaviest sum of weights is selected. We denote by m_{sel} the metadata selected.

3). Using m_{sel}, n_i calculates the set $N_{i,sel}$ through the formula $N_{i,sel} = \{n \mid (n, m_{sel}) \in L_i\}$. $N_{i,sel}$ consists of nodes that store the metadata m_{sel}.

4). If $N_{i,sel}$ is not empty, then n_i randomly selects one node from $N_{i,sel}$; if $N_{i,sel}$ is empty, n_i repeats step 2) and 3) until $N_{i,sel}$ is not empty or finally after some unsuccessful retries n_i randomly selects one node from its neighbors in the P2P network. We denote by n_{next} the node selected.

5). Node n_i forwards the query to n_{next}. The search procedure is repeated from node n_{next} on until the query succeeds or some stop criteria are met and the query fails.

6). If the query succeeds, n_i goes to step 7). If the query fails, n_i goes back to step 4) and chooses another node in $N_{i,sel}$; if all nodes in $N_{i,sel}$ are selected and the query still fails, then n_i goes back to step 2) and chooses the metadata in MSW with the next-heaviest sum of weights, etc..

7). Suppose node n_{meta} is the destination node which stores metadata m_r. Node n_i contacts node n_{meta} and retrieves m_r.

8). From document location information stored in m_r, node n_i finds out the node n_{doc} which stores the desired document. Then node n_i inserts m_{sel} into the leftmost slot in its MSW as we described in section 4.2; retrieves the desired document from node n_{doc}; inserts (n_{doc}, m_r) pair into L_i; retrieves metadata repository M_{doc} and (node, metadata) pairs set L_{doc} from node n_{doc}. And M_{doc} is merged with M_i and L_{doc} is merged with L_i.

The search procedure above is for nodes which have been in the P2P network for some period and have known some other nodes. For a node which just joins the P2P network, the bootstrap and search procedure is described in section 4.4. Step 6) in above procedure deals with

cases in which query fails, in fact, the times we retry in case of query failure is limited. If the query still fails after maximum number of retries, we stop retrying and abort the query. Although the number of data units stored in M_i and L_i can be very large, the storage space of one node is limited. When the storage limit is reached, we update M_i and L_i by replacing the least recently used data unit except those describing local node or documents.

Stop criteria are used to prevent infinite probing and are typically used in the form of TTL as we do in our simulations. However, the value of TTL in a large distributed system is hard to determine. Too small TTL value means you can not find what you want though they are near from you; too large value means excessive search traffic in the network. More discussions about stop criteria can be found in [11] and [12].

4.4. Bootstrap Procedure

If one node is new to the P2P network, it knows nothing about the network and other nodes. It needs some kind of bootstrap mechanism, e.g. YOID [16] which we choose, to learn the outside world. The bootstrap mechanism contains some bootstrap nodes which maintain a partial list of the nodes they believe are currently in the network. The IP address of a bootstrap node can be resolved by DNS. Through bootstrap nodes, the newcomer obtains a random part of the active nodes in the system. And through the nodes it obtains during bootstrap and with the process of query, the new node gradually learns more nodes that belong to the same interest-group as it does. Figure 2 depicts part of the bootstrapping and interest-group joining procedure of node n_i.

Because the new node does not know much about the network and the interest-group it belongs to after bootstrapping, it is desirable for the new node to issue some queries for popular documents first to become more familiar with the system. The more information the node knows about the interest-group, the more chances that query for rare items succeeds.

Metadata replication is an important feature in our proposed search mechanism. Metadata repository grows with the search process. On every successful query, the metadata repository from document holder is replicated and merged with the metadata repository of document requester, which increases the success probability of the next query in this interest-group.

The Metadata Slide Window deployed in the search model further increases the success probability of the next probe because metadata is selected according to previous successes. The probe sequence of our search model with MSW is not purely random; it is kind of heuristic because it is refined by past experiences. Simulations in the next section also prove the efficiency and effectiveness of our search model.

```
//Part of bootstrapping and interest-group joining
//procedure of node n_i
…
//Using YOID bootstrap mechanism to discover the set
//of a random part of active nodes
N_neigh = bootstrap_YOID();
…
do {
   //Select a random node from the neighbors of n_i
   Node n_next = select_random(N_neigh);
   //Forward the query to the selected node. If all nodes
   //in the neighbors set are selected, then fail the query
   if(n_next != NULL)
      m_r = n_next.search_procedure(query);
   else
      {m_r = NULL; break;}
} while (m_r == NULL && max_num_of_retries() ==
FALSE)
//If query succeeds, then retrieve the document and
//information about other nodes
if(m_r != NULL) post_process_on_success(m_r);
…
```

Figure 2. Pseudo code of bootstrapping

5. Simulation Results

We choose anonymized Boeing proxy logs [17] as the basis of the data used in our simulations. In Boeing proxy logs, username is identified by client ID and hostname of an URI is identified by host ID. We choose the proxy log on the day of March 1, 1999. And we extract client IDs and host IDs from the chosen log. Due to the large size of the Boeing proxy log, we randomly extract 5000 client IDs from it, and the host IDs that are accessed by chosen clients are also extracted. After the extraction, we have total 5000 client IDs and 4382 host IDs, note that one host ID may be accessed by several client IDs. We use client IDs to simulate nodes in the system and host IDs to simulate documents stored by nodes.

We choose an enhanced random search model of Gnutella which includes 32 random walkers. A random walker is a search process which randomly probes neighboring node in the system until the query is satisfied or some stop condition is met. Using multiple random walkers can reduce the query delay at the cost of increased network traffic. We run the simulations on both our search model and the enhanced random search model of Gnutella. Stop condition in both models is setting TTL=12 and we also set maximum number of retries in

our search model to 32 which equals the number of random walkers in Gnutella.

In one simulation, we run queries issued consecutively for each one of the randomly chosen 1000 documents in the system. Each query is initiated from a random node. We run the simulation 10 times and measure following metrics to compare the performance of both models:

1). Success Rate: Success rate of query is defined as the number of successful queries over the number of total queries issued. This metric captures the effectiveness of the simulated search model.

2). Path Length: Path length is the number of hops a query travels after it finally finds the desired metadata. Path length suggests the efficiency of the simulated search model. With small path length, we can find desired metadata quickly.

The simulation results are showed in following figures. Figure 3 depicts success rate of the two simulated search models in each simulation. We group every consecutive 50 queries together and calculate average path length of successful queries in each group; results are outlined in Figure 4.

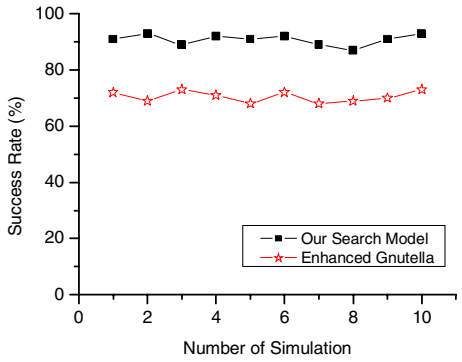

Figure 3. Success rate comparisons

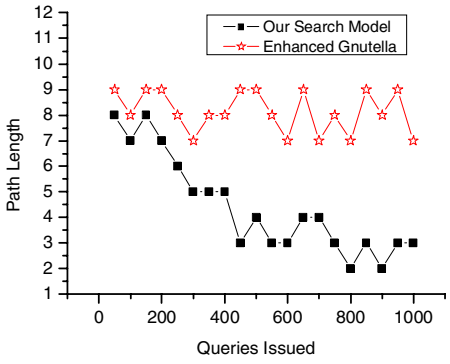

Figure 4. Path length comparisons

From Figure 3 we can see that the success rate of our proposed search model is higher than that of Gnutella in each simulation. In fact, the average success rate of our search model is almost 20% higher than that of Gnutella, which proves effectiveness of our search model over that of Gnutella. Figure 4 shows that at the beginning of the search process, because nodes in the system have not been familiar enough with each other, the average path length of successful queries of our search model is similar with that of Gnutella. However, with more queries issued, we can see that the average query path length of our search model becomes significantly shorter than that of Gnutella.

6. Conclusions and Future Work

This paper proposes an efficient search mechanism based on interest-group in unstructured P2P networks. Interest-group is formed by nodes sharing similar interests. We use metadata to describe and represent documents in the system. Unlike blind search in Gnutella, queries are propagated in interest-group first in our search model. By using Metadata Slide Window and metadata replication mechanisms, we can achieve much better performance than Gnutella. We describe in detail our basic search model, search and bootstrap procedure, and simulations. And the effectiveness and efficiency of our search mechanism is proved by the preliminary analysis and simulation results.

Although our search model has a significant advantage over random search model in Gnutella-like unstructured P2P networks, there are still many improvements to be made. For example, we plan to add reputation mechanism to our search model, which ranks nodes according to their content and services. We still need some better metadata update and replacement strategies. Also we need to study the relationships between different interest-groups and how they affect the search process and further refine our search model to make it more flexible.

References

[1] Napster Inc. Napster Homepage. In *http://www.napster.com/*, 2001.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of SIGCOMM'2001*, 2001.

[3] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of SIGCOMM'2001*, 2001.

[4] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of 18th IFIP/ACM*

International Conference on Distributed Systems Platforms, 2001.

[5] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for wide-area fault-tolerant location and routing. *U. C. Berkeley Technical Report UCB/CSD-01-1141*, 2001.

[6] Open Source Community. Gnutella. In *http://gnutella.wego.com/*, 2001.

[7] Matthew Harren, Joseph M. Hellerstein, Ryan Huebsch, Boon T. Loo, Scott Shenker, and Ion Stoica. Complex Queries in DHT-based Peer-to-Peer Networks. In *Proceedings of IPTPS'02*, 2002.

[8] Omprakash D Gnawali. A Keyword-Set Search System for Peer-to-Peer Networks. *M.I.T. Master Thesis*, 2002.

[9] L.A. Adamic, R.M. Lukose, A.R. Puniyani, and B.A. Huberman. Search in Power Law Networks. In *Physics Review, E64, pages 46135-46143*, 2001.

[10] A. Iamnitchi, M. Ripeanu, and I. Foster. Locating Data in (Small-World?) P2P Scientific Collaborations. In *Proceedings of IPTPS'02*, 2002.

[11] B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.

[12] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th annual ACM International Conference on supercomputing*, 2002.

[13] Salton, G., and Yang, C. On the specification of term values in automatic indexing. In *Journal of Documentation 29 (1973) 351-372*, 1973.

[14] W3C. Resource Description Framework. In *http://www.w3.org/RDF/*, 2002.

[15] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic indexing. Journal of the American Society for Information Science, 41(6):391–407, September 1990.

[16] P. Francis. Yoid: Extending the Internet Multicast Architecture. Unpublished paper, available at http://www.aciri.org/yoid/docs/index.html, Apr. 2000.

[17] J. Meadows. Boeing proxy logs. In *ftp://researchsmp2.cc.vt.edu/pub/boeing/*, March 1999.