http://www.zju.edu.cn/jzus; E-mail: jzus@zju.edu.cn
ISSN 1009-3095 *Journal of Zhejiang University SCIENCE* V.4, No.5, P.549-554, Sep.-Oct., 2003

A "cluster" based search scheme in peer-to-peer network

LI Zhen-wu(李振武)[†1], YANG Jian(杨舰)[2], SHI Xu-dong(史旭东)[1], BAI Ying-cai (白英彩)[1]
([1] *Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200030, China*)
([2] *Department of Computer Science and Engineering, Fudan University, Shanghai 200433, China*)
[†]E-mail: li-zw@cs.sjtu.edu.cn

Received Nov.21, 2002; revision accepted Jan.8, 2003

Abstract: This paper presents a "cluster" based search scheme in peer-to-peer network. The idea is based on the fact that data distribution in an information society has structured feature. We designed an algorithm to cluster peers that have similar interests. When receiving a query request, a peer will preferentially forward it to another peer which belongs to the same cluster and shares more similar interests. By this way search efficiency will be remarkably improved and at the same time good resilience against peer failure (the ability to withstand peer failure) is reserved.

Key words: Peer-to-peer network, Cluster, Overlay, Structured feature, Request pathlength
Document code: A **CLC number:** TP393

INTRODUCTION

A good search scheme is a critical component in peer-to-peer network. An effective search scheme should have good scalability, high search efficiency, good peer failure tolerance and support to keyword partial-match.

Existing searching schemes can be classified into three categories:

1. Centralized search scheme

This scheme employed by Napster (2001) has wide search scope and supports keyword partial-match. But centralized index servers become the bottleneck. Thus scalability is not available.

2. Decentralized "unstructured" search scheme

This kind of search scheme is totally decentralized. It has good scalability and support keyword partial-match. Loose "overlay"("overlay" network topology is not tightly controlled) makes them have good resilience against peer failure. But because of the random query forwarding, search efficiency is low. In order to improve search efficiency, Gnutella (2001) employed limit flooding. But this caused big network flow.

3. Decentralized "structured" search scheme

This kind of search scheme (Stoica *et al.*, 2001; Ratnasamy *et al.*, 2001; Druschel *et al.*, 2001; Zhao *et al.*, 2001) is based on Distributed Hash Tables(DHTs), which has advantage of good scalability and guarantees that files will be found. But strict control on "overlay" network causes large resources consumption and sensitivity to peer failure. In addition, the use of DHTs leads to unavailability of keyword partial-match.

This paper presents a "cluster" based search scheme in peer-to-peer network. Its idea is based on the fact that data distribution in an information society has structured feature. We designed an algorithm to cluster peers that share similar interests. When receiving a query request, a peer will preferentially forward to another peer which is in the same cluster and has more similarities. This scheme has good convergence, high search efficiency, and good resilience against peer failure.

SYSTEM MODEL

Our scheme is based on the conclusion drawn from Deerwester *et al.* (1990). Data in the modern society are not placed at random. Different data in the same entity share certain interests. If some entities share some data they are more likely to share some other data. So our scheme establishes the following hypothesis: If peer1 can answer the previous query request of

peer2, it is more likely to answer the next query request of peer2; If several peers all have succeeded in answering the previous query request of peer2, the peer that has recently answered more request queries of peer2 has the greatest probability to satisfy the next query request of peer2. So if peer2 forwards its query request to the peer that has the greatest probability to satisfy its query request, its query efficiency will be greatly improved. On top of the high efficiency, our scheme does not require extremely tight control, so it has good resilience against peer failure. And because our scheme does not employ hash, keyword partial-match is supported, too.

We assume that the system has n peers and m documents. Relation between peers and documents is represented by a 2-tuple $P(PEER, DOCUMENT)$, $PEER \in [1, n]$, $DOCUMENT \in [1, m]$. When peer i owns document k, $(i, k) \in P$.

Definition 1. P_i is used to represent the state set that peer i maintains. It is composed of $(PEER, DOCUMENT)$ pair, which means the relation of document and peer. It is a subset of P. If peer i knows peer j owns document k and peer t owns document u, then peer i inserts (j, k) and (t, u) into P_i, so (j, k) and $(t, u) \in P_i$.

Definition 2. D_i is used to represent document set which is composed of document index (which is used to uniquely identify a document). These documents are those peer i knows other peers own.

Definition 3. P_{ik} is used to represent a peer set that is stored in peer i as state information and all peers in P_{ik} must own document k.

Our scheme is based on "loose overlay network". Each peer maintains two kinds of state information: state set and popular document window W. We take peer i as an example.

Peer i maintains two kinds of state information.

1. State set P_i

From P_i we can derive another two sub-state sets:

1) Document set D_i.

2) Peer set $P_{ik}(k \in D_i)$ that own document k.

According to the above example, peer i maintains:

$D_i = \{k, u, z\}$

$P_{ik} = \{j, t\}$, $P_{iu} = \{t\}$, $P_{iz} = \{s\}$

2. Popular document window W

Popular document window W is used to record what documents are most commonly shared by local peer with other peers. It is made up of some document elements. Each element is a document index (which is used to uniquely identify a document). All elements are assigned different weight according its position in W. From left to right, weight is $|W|$, $|W|-1$, $|W|-2$, …, 2, 1. For example, peer i has the window shown in Fig.1.

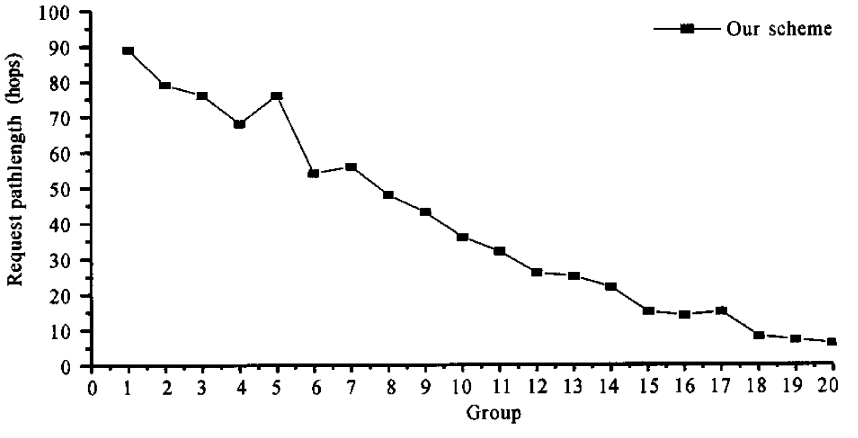

Fig.1 Request pathlength versus query process

Algorithm for peer i to calculate the popularity of a particular document is described below.

Peer i groups all elements in window W according to the document index value. Elements that have the same document index are grouped into the same group. We assume some Group is represented by G. G is a 2-tuple $G(DOCUMENT, WEIGHT)$. $DOCUMENT$ represents document index and $WEIGHT$ represents its weight in window W. We suppose the weight set of group G is W_G. Then the sum of the weight of group G which is represented by $W(G)$ equals $\sum_{W_i \in W_G} W_i$. Then peer i orders all the groups according to the sum of weight. Peer i will decide that the group with the max sum of weight has the most popular document index. Then peer i chooses one peer from peers that own the most popular document to forward query.

Definition 4. A cluster is a peer set which contains peers that share one or several documents. Because peers in the same cluster share some documents, they have common interests. So if a peer forwards its query to the peer in the same cluster, the query efficiency will be higher than just choosing a random peer.

We take peer i as an example, all peers in $P_{ik}(k \in D_i)$, $P_{jk}(j \in P_{ik})$, $P_{tk}(t \in P_{jk})$... form a cluster. One peer may belong to more than one cluster because it may share documents with more than one peer.

SEARCH ALGORITHM

When a peer j, which has joined the "cluster", has a query request for document k, it employs the following algorithm:

1. Peer j calculates the sum of the weight in its popular document window W using the algorithm described in Section 2. We suppose the chosen popular document index value is r. Go to 2.

2. Peer j chooses a peer at random from peers which own document r. We suppose the chosen peer is t, $t \in \{x \mid (x, r) \in P_j\}$:

If peer t owns document k, peer j shifts its window W one unit to the right and inserts r into the most left position in window W. Peer j retrieves document k from peer t and inserts (t, k) into its state set P_j. At the same time peer j gets the peer set P_{tk} from peer t and inserts (x, k) $(x \in P_{tk})$ into its state set P_j. The query ends.

If peer t does not have document k, peer t repeats the above query process until document k is found or some predetermined depth is reached. If document k is found, the query ends. If document k is not found, then go to 3.

3. Peer j chooses one group which has the max sum of the weight from the rest of the groups and gets its document index. We suppose the chosen document index is r. Go to 2. This process is repeated until document k is found or some predetermined depth is reached.

From the above algorithm description we can see that the "cluster" is expanded with the search process. If peer j retrieves document k from peer t, peer j retrieves the peers set which own document k from peer t and inserts the peer set into local state space. So the cluster to which peer j attaches is expanded. In addition, if peer j cannot contact peer t, it will just try another peer from peers which own document k. Through this way good peer failure tolerance is available in our scheme.

If a new peer i wants to join the "overlay network", it can only employ the same bootstrap mechanism as Yoid (2000) to find one peer which has been included in the overlay. We suppose the chosen peer is j. Then peer j starts its search process as described above. Peer i gets state information from peer j. Then peer j can join the overlay network.

SIMULATION RESULT

The test data in our scheme is based on a "Document/Term (short phrase)" model, which is similar to the "Peer/Document" model. We randomly retrieved 1000 documents from the Internet and employed TRS (2002), a Chinese Segmentation software, to retrieve the lexion from each document (a lexicon is a list of all the unique terms that appear in a document). Then we choose at random 50 terms from each document's lexicon (each document contributes 50 terms). Now we get the document/term model. Each document resembles a peer and the terms contained by the document resemble the documents contained by the peer. Our test environ-

ment is a centralized platform, which is a PC. Its hardware configuration is PIII 800M, 128M RAM and its software environment is RedHat 7.2.

The benchmark of our scheme is N-Gnutella, which is a variation of Gnutella. It also uses our centralized simulation environment. So there are 1000 peers and each peer contains 50 documents. Compared with Gnutella, N-Gnutella has two major differences:

1. It does not employ flooding, but just chooses a random neighbor to forward query when processing query.

2. It is modified to adapt to our centralized test environment.

There is only one major evaluation metric in our scheme: average request pathlength. Based on this metric, we evaluate the network convergence, peer failure tolerance and search efficiency in a stable network state.

Our first test is to evaluate the network convergence. Because in our schemes network overlay(cluster) is constructed along with the process of search, we test whether the average request pathlength will decrease with search process. We choose 200 terms from all 50000 terms at random and group them into ten groups. So each group has 20 terms. For each group, we start to query each term in the group from a randomly selected peer and get its request pathlength and then calculate the average request pathlength in the group. We do the same thing for every group. The test result is shown in Fig.1.

Fig.1 shows the evolution of the first, second, third, . . . , nineteenth, twentieth group of the request pathlength in the search process. We can see that the initially high pathlengths decrease rapidly over the search process. In the beginning, queries have long pathlengths, but as the network converges, the request pathlength drops below 10.

Our second test is to evaluate the network resilience to peer failure. After the first test, the network has evolved into a relatively stable state. Then we gradually remove some peers to test the query pathlength. Each time we remove 2% peers we then query ten terms at random. If some query can not be successful below 200 steps, we stop this query and assign 200 to its request pathlength. After ten term queries we calculate the average pathlength. Test result is shown as Fig.2.

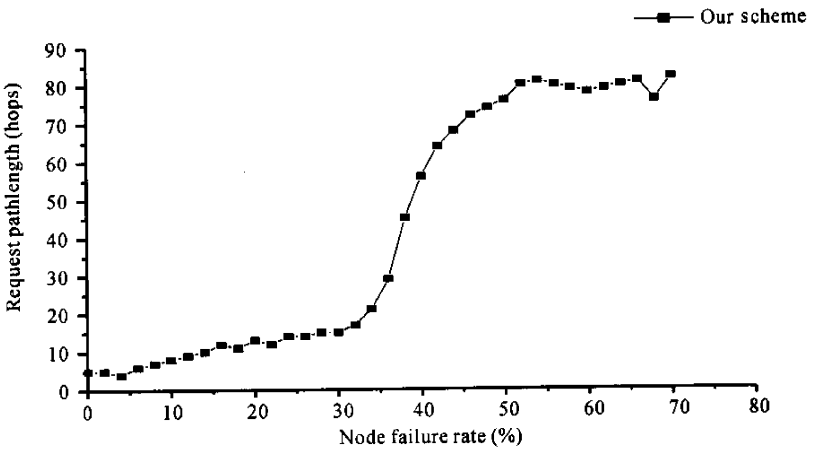

Fig.2 Request pathlength versus node failure rate

Fig.2 shows that our scheme has good resilience to peer failure. The network is surprisingly robust against quite large failures. The request pathlength remains below 30 even if peers failure reaches 30%.

Our third test compares the efficiency of our scheme with N-Gnutella. We compare the average request pathlengths of our scheme with N-Gnutella's when the network is in stable state. We make a query on 20 groups terms(each group has 10 terms). We calculate each group's average request pathlength. The test result is shown in Fig.3.

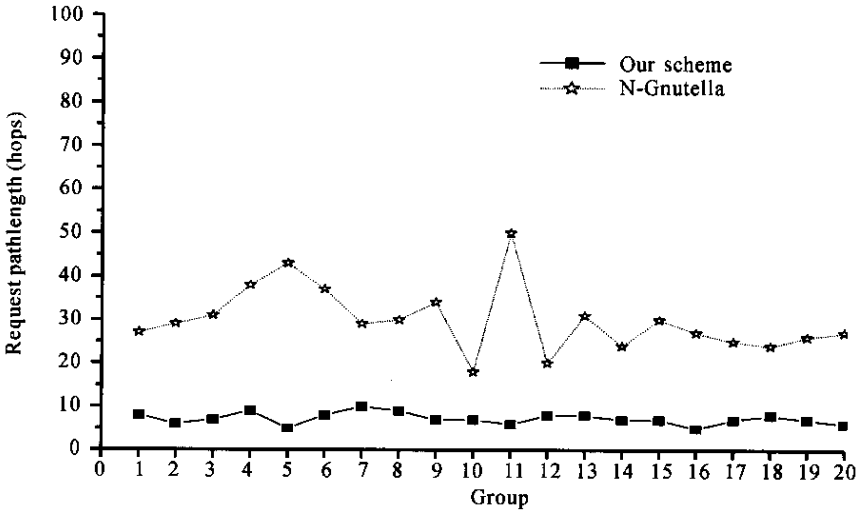

Fig.3 Request pathlength versus group

Fig.3 shows that the average request pathlength in our scheme is far below that in N-Gnutella. The ultimate cause is that N-Gnutella chooses random peer to construct overlay and forward query but in our scheme peer chooses neighbors which have similar interests and forward query to a peer which has greater probability to satisfy query. The overlay in our scheme is constructed through search process. So peers which have similar interests are clustered and have relatively close relation. The search efficiency is therefore greatly improved.

Our fourth test compares the search efficiency on rare terms in our scheme with that in N-Gnutella. At first on each term we employ a simple algorithm to find the number of documents which own the term. Then we choose 100 terms which have smallest document number and group them into 5 groups at random. Each group has 20 terms and the 20 terms are ordered by the number of documents which own the term. They are numbered from 1 to 20. After this we have the final 20 groups. Each group has 5 terms, each of which has the same serial number as that in the original group. We calculate the average number of documents in each group which owns each item. According to the ratio of average number of documents which own the term to the total number of documents, we have the following array:

0.12%, 0.16%, 0.22%, 0.24%, 0.28%, 0.32%, 0.36%, 0.38%, 0.42%, 0.44%, 0.48%, 0.50%, 0.52%, 0.54%, 0.58%, 0.60%, 0.62%, 0.64%, 0.66%, 0.68%

We conduct the query for each group in stable network state(our scheme and N-Gnutella) and get the average pathlength in each group. The test result is shown in Fig.4.

Fig.4 shows that compared with N-Gnutella our scheme improves the efficiency of search on rare terms and expands the search scope.

CONCLUSION

This paper presents a "cluster" based search scheme in peer-to-peer network. Its idea is based on the fact that data distribution in information society has structured feature. We designed an algorithm to cluster peers which have similar interests. When receiving a query request, peer preferentially forwards to peer which is in the same cluster and has more similarities. Through this scheme, search efficiency is remarkably improved and at the same time good resilience to peer failure is achieved. The following work still needs to be done in the future:

1. When choosing peer to forward query we should take consideration of peer's dynamic performance state.

2. Multi-keywords complex query should be supported.

3. When choosing peer to construct overlay we should take into account the physical network overlay.

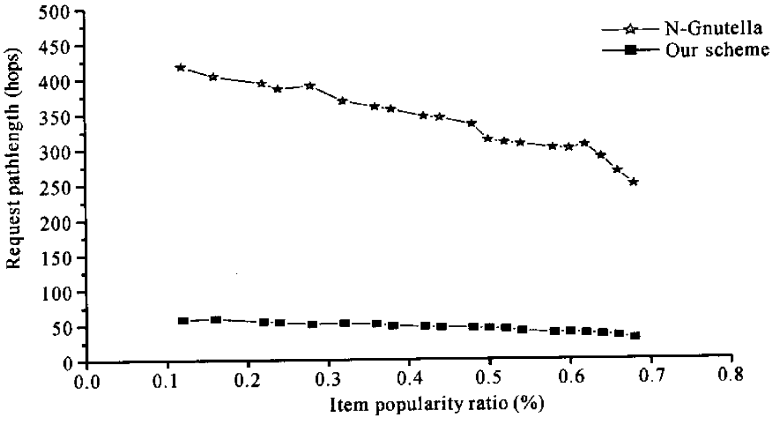

Fig.4 Request pathlength versus item popularity

ACKNOWLEDGMENTS

We thank the reviewers of this paper for their valuable comments and suggestions.

References

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

Druschel, P., and Rowstron, A. 2001. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer System. In Proceeings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware 2001)W (Nov. 2001).

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

Gnutella: http://gnutella.wego.com/, 2001.

Napster: http://www.napster.com, 2001.

Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S., 2001. A Scalable Content-Addressable Network. In Proc. ACM SIGCOMM 2001.

Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M. and Hari Balakrishnan, 2001. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *In*: Proc. ACM SIGCOMM 2001.

TRS: http://www.trs.com.cn/product/content/200203260001.jsp, 2002

Zhao, B. Y., Kubiatowicz, J. and Joseph, A., 2001. Tapstry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing. Tech. Rep. UCB/CSD-01-1141, Univeristy of California at Berkeley, Computer Science Department.