Global Journal of Computer Science and Technology
Volume 11 Issue 11 Version 1.0 July 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Proposal of a Flooding-Based Flexible Search Method for Chord Networks

By Keiichi Endo, Kensuke Kanetada, Dai Okano, Kaname Amano

Ehime University, Bunkyo-cho 3, Matsuyama, Japan

Abstracts - Recently, peer-to-peer (P2P) network models have been attracting considerable attention. P2P models can be classified into structured and unstructured P2P models. Representative search methods for structured and unstructured P2P networks are Chord and Flooding, respectively. It is difficult to realize flexible search in Chord networks. On the other hand, a large number of query transmissions are required for Flooding. In this study, we propose a Flooding-based search method for Chord networks. Our method works separately from the traditional search method for Chord networks. It suppresses the transmission of redundant queries by considering the topological properties of the structured networks and enables flexible search in Chord networks. Through simulation experiments, we evaluate the performance of the proposed search method and show its effectiveness.

Keywords : P2P, overlay networks, distributed hash tables, Chord, Flooding.

GJCST Classification : I.2.1

Strictly as per the compliance and regulations of:

Proposal of a Flooding-Based Flexible Search Method for Chord Networks

Keiichi Endo[α], Kensuke Kanetada[Ω], Dai Okano[β], Kaname Amano[Ψ]

Abstract - Recently, peer-to-peer (P2P) network models have been attracting considerable attention. P2P models can be classified into structured and unstructured P2P models. Representative search methods for structured and unstructured P2P networks are Chord and Flooding, respectively. It is difficult to realize flexible search in Chord networks. On the other hand, a large number of query transmissions are required for Flooding. In this study, we propose a Flooding-based search method for Chord networks. Our method works separately from the traditional search method for Chord networks. It suppresses the transmission of redundant queries by considering the topological properties of the structured networks and enables flexible search in Chord networks. Through simulation experiments, we evaluate the performance of the proposed search method and show its effectiveness.

Keywords : P2P, overlay networks, distributed hash tables, Chord, Flooding.

I. Introduction

In recent years, various types of services are provided on the Internet. In most of the services, the client-server model, which is illustrated in *Figure 1*, is adopted. However, providing a satisfactory service in the client-server model is getting more and more difficult because of the increase of Internet users and the large amount of transferred data. Therefore, many researchers are exploring the ways to provide various types of services in the peer-to-peer (P2P) model, which is illustrated in *Figure 2*. In this model, a node (user's computer) shares some files with other nodes.

Flooding (*Figure 3*) is the most basic search method for P2P networks. The requester transmits a search query to all the adjacent nodes, and the query is continuously forwarded to adjacent nodes until the TTL (Time To Live) is reduced to 0. However, in this search method, many messages are transmitted, which include redundant queries sent to the nodes that have already received the same queries.

The Distributed Hash Table (DHT) enables file search in a P2P network with much smaller number of messages. In the DHT, shared files are associated with nodes by using a hash function. Since the network is constructed in certain rules, a requested file can be found with high probability and without redundant queries. However, realizing flexible (e.g., partial match, fuzzy, or full-text) search is a challenging problem.

In this paper, we propose a Flooding-based flexible search method for a structured network organized by a DHT algorithm. Our method suppresses the transmission of redundant queries by considering the structure of the organized network.

The rest of this paper is organized as follows. Section II describes structured and unstructured P2P models. In Section III, we explain the proposed search method. We evaluate the performance of the proposed method in Section IV. We conclude this paper and discuss future work in Section V.

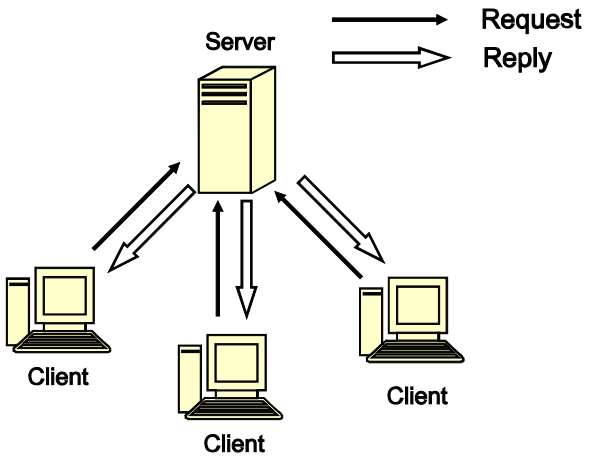

Figure 1 : Client-server model.

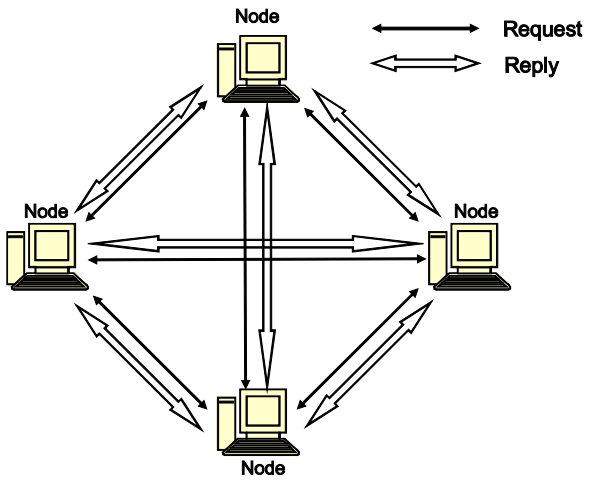

Figure 2 : P2P model.

Author [α Ω β Ψ] : Department of Electrical and Electronic Engineering and Computer Science, Graduate School of Science and Engineering, Ehime University, Bunkyo-cho 3, Matsuyama 790-8577, Japan.

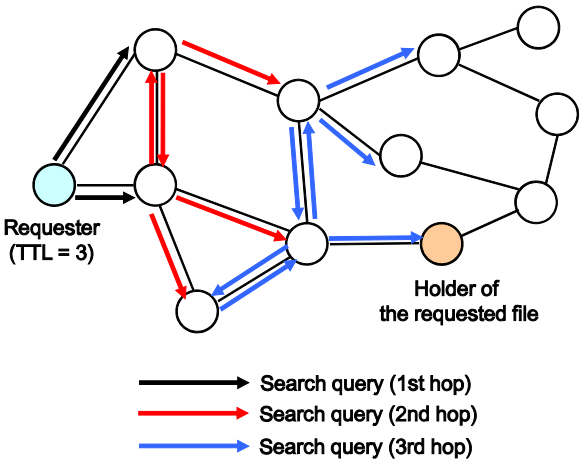

Figure 3 : Flooding search.

II. Structured and Unstructured P2P

P2P models can be classified into structured and unstructured P2P models. In this section, we explain the properties of each model and show some examples of existing search methods.

a) Structured P2P

Structured P2P networks are constructed in certain rules, and index information (describing the holders of shared files) or files themselves are distributed among the nodes. The DHT is a class of algorithms for constructing structured P2P networks and searching in those networks. Examples of DHT algorithms are CAN [2], Chord [3], Pastry [4], Tapestry [5], and Kademlia [6]. Since we propose a search method for Chord networks, we explain Chord in more detail below.

In Chord, the node ID that is calculated from node's IP address using a hash function is assigned to each node. The names of shared files are also hashed by using the same hash function, and each node manages the index information of the files whose hash values are less than or equal to the node ID and greater than the smaller adjacent node ID. For example, if the nodes in a Chord network are assigned node IDs as shown in *Figure 4*, the index information of shared files is placed according to *Table 1*. It should be noted that the nodes in a Chord network are considered to be circularly arranged. Therefore, the node ID 0 is often considered to be greater than the node ID 15.

We consider m-bit hash space, which means that node IDs are nonnegative integers less than 2^m. We define node S's i-th neighbor ($i = 0, 1, \ldots, m-1$) as the node assigned the ID next $((S + 2^i) \bmod 2^m)$, where next(X) denotes the smallest node ID greater than or equal to X. In the following discussion, we omit the remainder operation "mod 2^m."As an example, we show the neighbors of node 0 in *Figure 5*. We refer to node $(S + 2^i)$ and node next$(S + 2^i)$ as calculated neighbor and actual neighbor, respectively.

When a node searches for a file, a query is transmitted and forwarded to the neighbor whose ID is the closest to (and except for the last hop, not greater than) the hash value calculated from the name of the requested file until it reaches the node that should manage the index information of the requested file. For example, a query is transmitted and forwarded as shown in *Figure 6* when node 0 searches for a file whose hash value is 10. Since node 10 does not exist, node 11 replies to node 0 (with index information of the requested file if available).

In a Chord network, a requested file is found within $\log_2 N$ hops (N denotes the number of nodes) if the file exists in the network. In addition, the communication cost is low because queries are not replicated. However, it is difficult to realize flexible search because the destination node ID of a query is calculated basically from the name of the requested file, which is a mutual problem of the DHT.

A DHT algorithm that uses a locality-preserving hash function is proposed in [7]. Skip Graphs [8], SkipNet [9], and Skip Tree Graphs [10] do not use hash functions to construct structured P2P networks or to search in the networks. These methods enable range queries; however, it is still difficult to realize as flexible search as search methods for unstructured P2P networks such as Flooding. Moreover, load balancing is difficult if we adopt these methods.

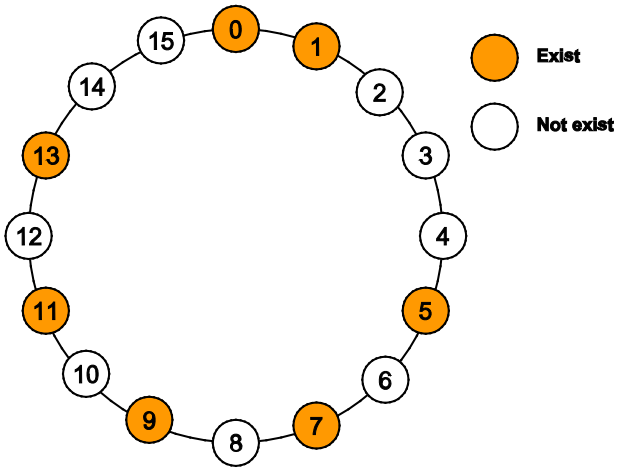

Figure 4 : Example of a Chord network.

Table 1: Index information allocation.

Node	Hash values of files
Node 0	14, 15, 0
Node 1	1
Node 5	2, 3, 4, 5
Node 7	6, 7
Node 9	8, 9
Node 11	10, 11
Node 13	12, 13

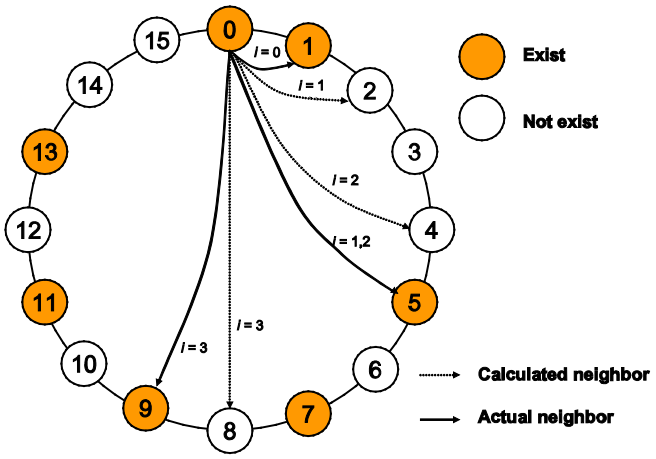

Figure 5 : Neighbors of node 0.

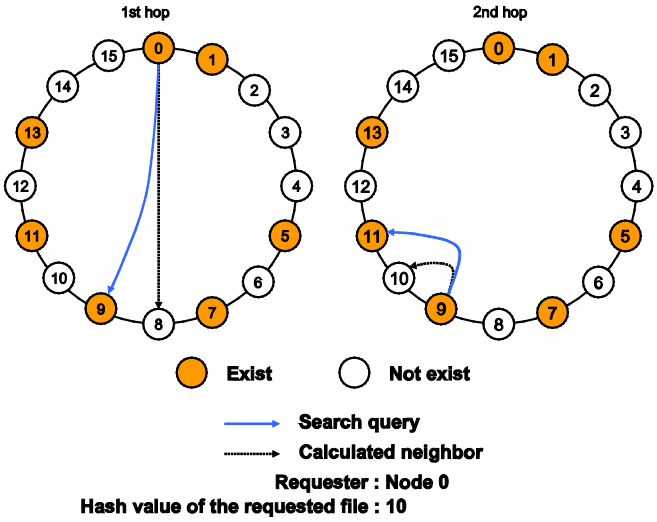

Figure 6 : Traditional Chord search.

b) Unstructured P2P

Unstructured P2P networks are constructed without strict restrictions. Examples of search methods for unstructured P2P networks are Flooding [11], *k*-Walker Random Walk [12], and the hybrid search scheme incorporating Flooding and Random Walk [13].

In Flooding (and many other search methods for unstructured P2P networks), TTL is used to control the propagation of queries. By using Flooding, a node searches for a file in the following procedure:

1. The TTL of a query is initially set to a certain positive integer.
2. The requester transmits the query to all the adjacent nodes. The TTL of the transmitted query is decremented.
3. If the receiver of the query possesses the requested file, the receiver replies to the requester. Otherwise, the receiver forwards the query to all the adjacent nodes except for the sender of the query unless the receiver has received the same query before. In the same manner as step 2, the TTL of the forwarded query is decremented.
4. Step 3 is repeated until the TTL is reduced to 0.

Since the paths of queries do not depend on the names (or the keywords) of requested files, various types of queries can be supported in unstructured P2P networks. However, the communication cost is high mainly because of redundant queries. Additionally, a node often fails to find a requested file that exists in the network because the spread of queries is restricted by TTL.

III. Proposed Method

In this section, we propose a flexible search method for Chord networks. Although our method is based on Flooding, it suppresses the transmission of redundant queries by considering the topological properties of the structured networks. Our method works separately from the traditional search method for Chord networks. Therefore, the traditional search method is still valid for exact match search when our method is adopted.

a) Basic Algorithm

In the proposed method, LTS (Limit To Send) is used instead of TTL in order to control the propagation of queries. A node searches for a file in the following procedure:

1. The LTS of a query is initially set to m.
2. The requester transmits the query to the requester's i-th neighbors such that $i <$ LTS. The LTS of the query that is transmitted to the i-th neighbor is changed to i.
3. If the receiver of the query possesses the requested file or the index information of the requested file, the receiver replies to the requester. Otherwise, the receiver forwards the query to the receiver's i-th neighbors such that $i <$ LTS unless the receiver has received the same query before. In the same manner as step 2, the LTS of the query that is forwarded to the i-th neighbor is changed to i.
4. Step 3 is repeated until the LTS is reduced to 0.

In this method, search terminates within m hops because the LTS is decreased by at least one every time the query is forwarded. In addition, as shown in *Figure 7*, all the nodes can receive the query even if the number of nodes is 2^m, which is the maximum number supported by the m-bit hash space. Furthermore, no redundant queries are transmitted in this case.

However, in practice, the i-th actual neighbors for different i's can be the same node because some node IDs are absent in a Chord network. For instance, in *Figure 8*, node 0's 0th, 1st, and 2nd actual neighbors are node 4. In this case, if the LTS of the query transmitted by node 0 to node 4 is changed to 0 (the smallest of 0, 1, and 2), node 6 cannot receive any queries as shown in *Figure 8*. Therefore, the LTS should

be changed to 2 (the largest of 0, 1, and 2). Although all the nodes can receive the query in this policy, some redundant queries are transmitted as shown in *Figure 9*. In the case of *Figure 10*, the number of transmitted messages is doubled by redundant queries.

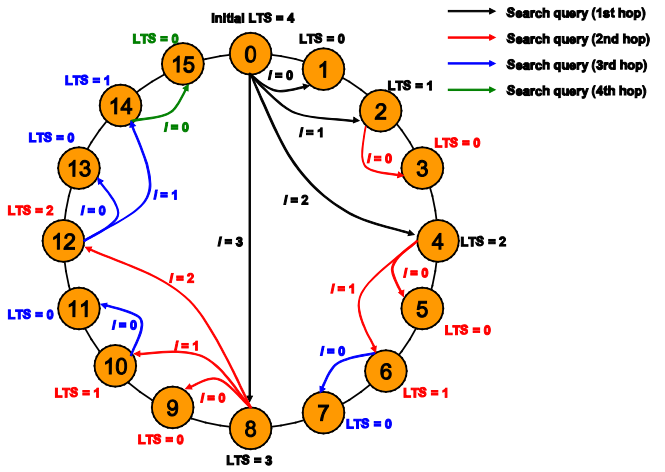

Figure 7 : Proposed search method.

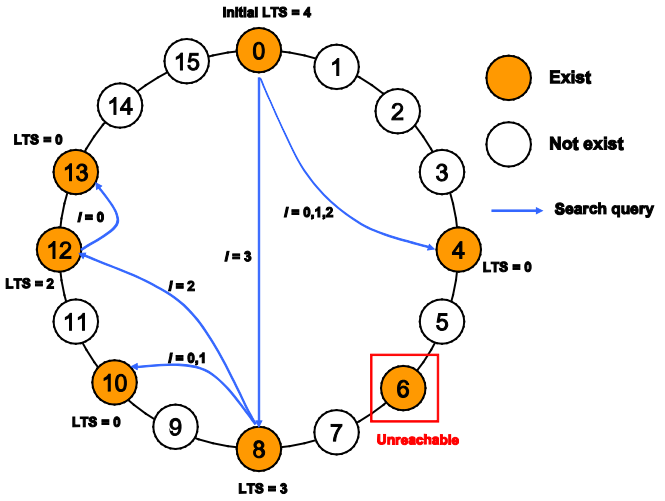

Figure 8 : Adopting the smallest *i* for LTS.

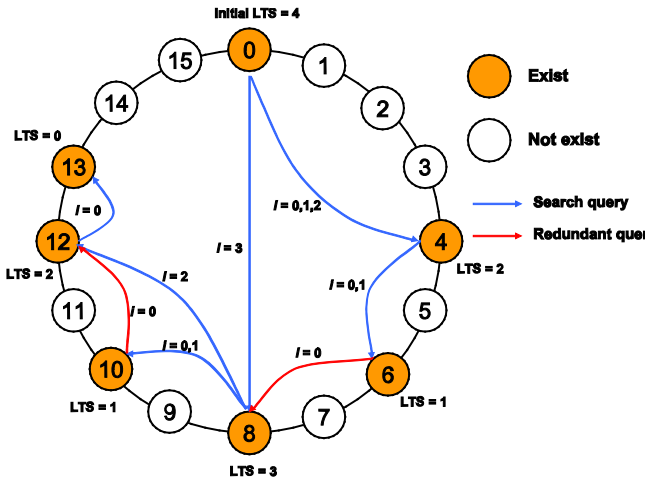

Figure 9 : Adopting the largest *i* for LTS.

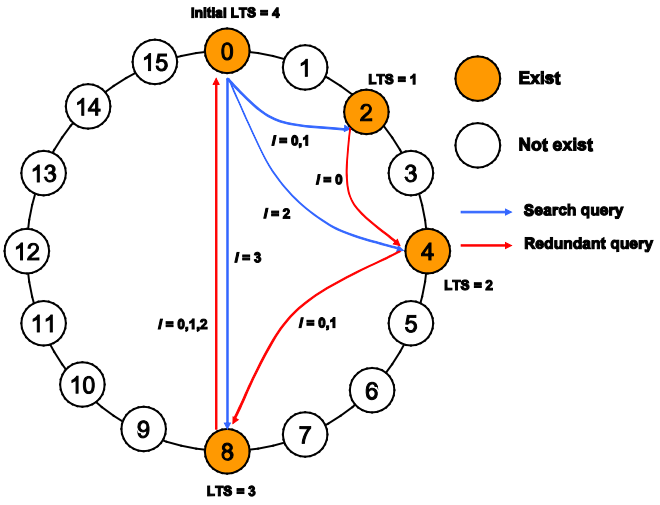

Figure 10 : Example of redundant queries.

b) Redundant Query Suppression

We suppress the transmission of redundant queries by including *StopID* in the query. The query is not forwarded to any node whose ID is greater than or equal to *StopID*. After this modification is applied, no redundant queries are transmitted. In the case of *Figure 10*, *StopID* of the query transmitted by node 0 to node 4 (the 2nd neighbor) is set to 8 (the 3rd neighbor) because node 0 transmits the same query to node 8 at the same time. In this manner, the transmission of redundant queries can be completely suppressed.

c) Index Information Replication

In this subsection, we consider how to decrease the number of transmitted messages. If we change the initial LTS from m to m - 1, the number of messages for search is approximately halved. However, since only half of the nodes in the network can receive the query, search success ratio becomes lower unless index information is replicated. If replicated index information is placed in the diagonal node (2^{m-1} larger node ID), search success ratio is not deteriorated. However, the number of messages for index information placement is doubled if the replicated index information is transmitted independently from the original. We propose a more efficient method to place replicated index information.

1. When a new file is shared, the hash value of the file, *HashA*, is calculated.
2. The diagonal node ID, *HashB* = *HashA* + 2^{m-1}, is calculated.
3. We let *HashNear* be the closer node ID of *HashA* and *HashB* from the holder of the newly shared file. We also let *HashFar* be the farther node ID of the two.
4. A message for index information placement is transmitted to the neighbor whose ID is the closest to *HashNear*.
5. The receiver of the message forwards it to the neighbor where the index information should be

placed (i.e., the neighbor whose ID is the smallest and not less than *HashNear*). If it is not possible, the message is forwarded to the neighbor whose ID is the closest to *HashNear*.

6. In step 5, if *HashNear* lies on or between the calculated neighbor ID and the actual neighbor ID, another message for index information placement is transmitted to *HashFar*.

Figure 11 illustrates this procedure. In this figure, the numbers in parenthesis correspond to the steps described above. This method enables efficient placement of index information because diagonal links are utilized.

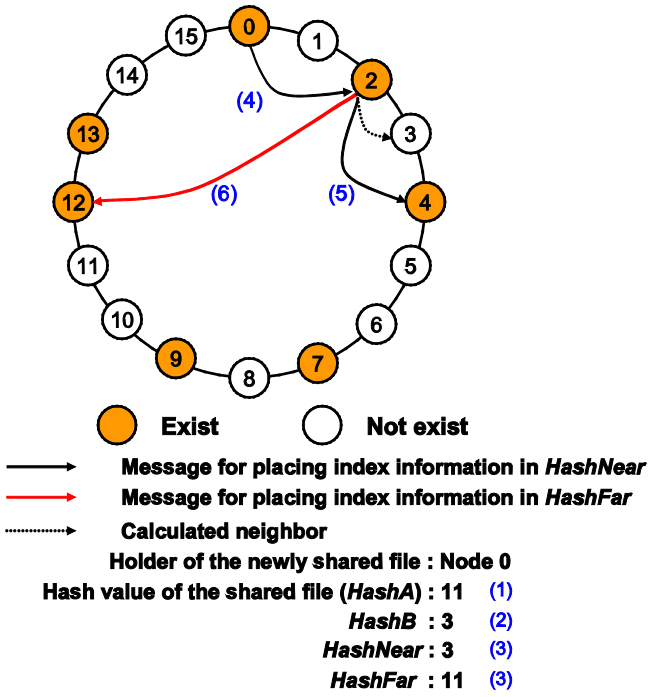

Figure 11 : Index information placement.

IV. Simulations

Through simulation experiments, we evaluate the performance of the proposed method and the effect of the modifications described in the previous section. We compare the following four methods:

- Chord0: traditional Flooding described in Section II-B. (Flooding is usually used in an unstructured network; however, in this section, we use Flooding in a structured Chord network.)
- ChordA: the proposed method described in Section III-A.
- ChordB: the proposed method with redundant query suppression described in Section III-B.
- ChordC: the proposed method with redundant query suppression and index information replication described in Section III-C.

Table 2 shows the common parameters used in the simulations. The initial TTL or LTS for each method is shown in *Table 3*. All the results shown in this section are based on the average of five experimental results in the same condition.

Figure 12 shows the ratio of redundant messages to all the messages transmitted for search. The ratio in ChordA is much lower than the ratio in Chord0, which implies that the basic algorithm of the proposed method achieves much more efficient search than traditional Flooding in Chord networks. No redundant messages are transmitted in ChordB because the redundant query suppression takes effect. We omit showing the result of ChordC because the search algorithm of ChordC is the same as that of ChordB.

Figure 13 shows the number of messages transmitted for search. Compared with Chord0 and ChordA, ChordB achieves much lower communication cost. The number of messages for search in ChordC is half the number of messages in ChordB because only half of the nodes in the network receive the query. In *Figure 14*, we show the number of messages transmitted for index information placement. The results of ChordA and ChordB are the same as that of Chord0. This figure shows that the number of messages for index information placement increases if index information is replicated; however, the difference is independent of the number of nodes in the network.

Figure 15 shows the maximum number of hops that queries travel in a search. (For example, in the case of *Figure 9*, the maximum number of hops is three because the search terminates after the queries travel at most three hops.) The maximum number of hops in Chord0 is the smallest of the four methods in most cases. This is because the query is forwarded to all the adjacent nodes in Chord0. However, Chord0 suffers from high communication cost as we have shown in *Figure 13*. The maximum number of hops in ChordC is one hop smaller than that of ChordA or ChordB. This is because the search area is half of the network in ChordC.

Table 2 : Common parameters.

Size of hash space	30 bits
Number of nodes	2^i (i = 5, 6, ..., 15)
Number of unique files	10000
Number of searches	1000

Table 3 : Parameters for determining search area.

Method	Initial value	Search area
Chord0	TTL = 30	all
ChordA	LTS = 30	all
ChordB	LTS = 30	all
ChordC	LTS = 29	1/2

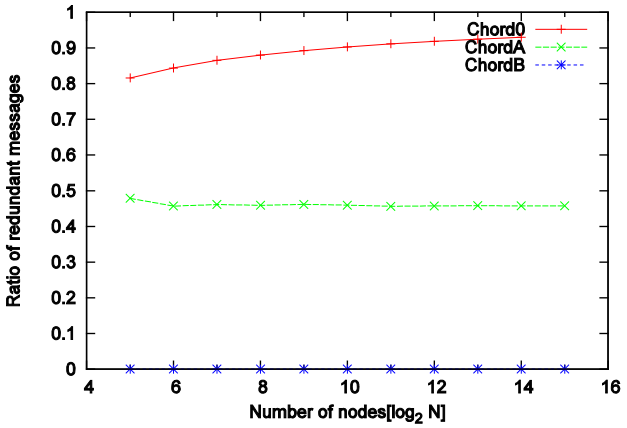

Figure 12 : Ratio of redundant messages.

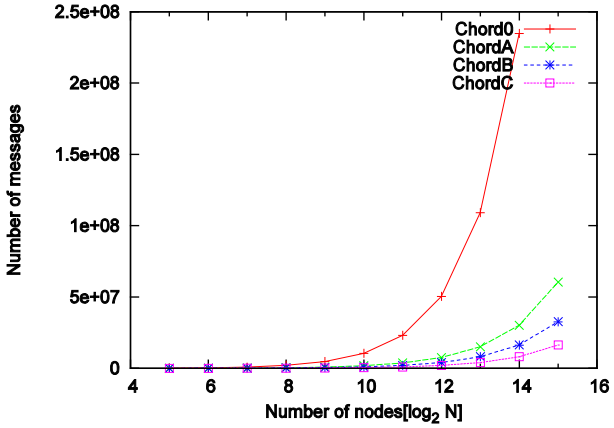

Figure 13 : Number of messages for search.

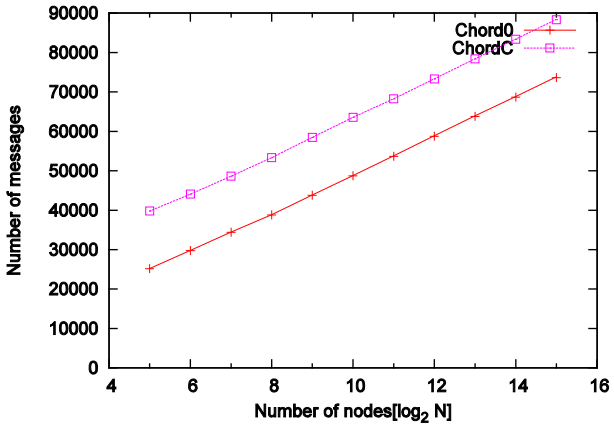

Figure 14 : Number of messages for index information placement.

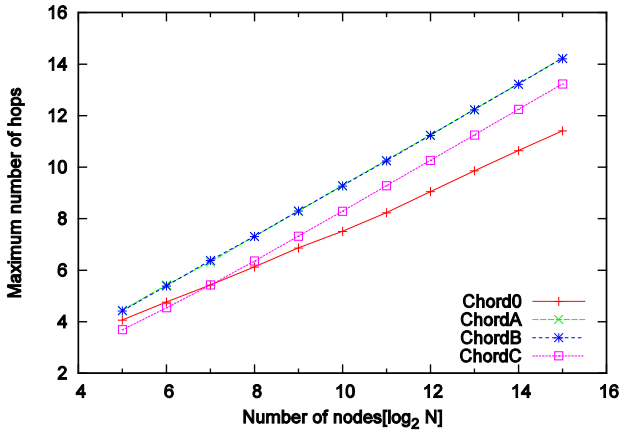

Figure 15 : Maximum number of hops.

V. Conclusion

In this paper, we have proposed a Flooding-based flexible search method for Chord networks. We have also described two modifications (redundant query suppression and index information replication) for improving the performance of the proposed method. The results of simulation experiments show that the proposed method enables efficient search.

In the future, we plan to investigate the method for reducing the communication cost with the search success ratio kept as high as possible. Since two parameters (LTS and *StopID*) are used for query forwarding in our method, the propagation of queries can be controlled more flexibly than other simpler methods such as [14]. We also plan to consider Flooding-based flexible and efficient search methods for various structured P2P networks constructed by other DHT algorithms such as Pastry [4], or non-DHT algorithms such as Skip Graphs [8] and BATON [15].

References Références Referencias

1. Ripeanu, M. and Iamnitchi, A. (2002). Mapping the Gnutella Network. *IEEE Internet Computing*, Vol. 6, pp. 50-57.
2. Ratnasamy, S., Francis, P., Handley, M., and Karp, R. (2001). A Scalable Content-Addressable Network. *Proc. SIGCOMM '01*, pp. 161-171.
3. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., and Balakrishnan, H. (2001). Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Proc. SIGCOMM '01*, pp. 329-350.
4. Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. *Proc. Middleware 2001*, pp. 329-350.
5. Zhao, B.Y., Kubiatowicz, J.D., and Joseph, A.D. (2001). Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. *U. C. Berkeley Technical Report UCB/CSD-01-1141*.

6. Maymounkov, P. and Mazières, D. (2002). Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. *Lecture Notes in Computer Science*, Vol. 2429, pp. 53-65.
7. Pitoura, T., Ntarmos, N., and Triantafillou, P. (2006). Replication, Load Balancing and Efficient Range Query Processing in DHTs. *Lecture Notes in Computer Science*, Vol. 3896, pp. 131-148.
8. Aspnes, J. and Shah, G. (2007). Skip Graphs. *ACM Transactions on Algorithms*, Vol. 3, No. 4, Article No. 37.
9. Harvey, N.J.A., Jones, M.B., Saroiu, S., Theimer, M., and Wolman, A. (2003). SkipNet: A Scalable Overlay Network with Practical Locality Properties. *Proc. the 4th Conference on USENIX Symposium on Internet Technologies and Systems*.
10. González-Beltrán, A., Milligan, P., and Sage, P. (2008). Range Queries over Skip Tree Graphs. *Computer Communications*, Vol. 31, No. 2, pp. 358-374.
11. Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. (2002). Search and Replication in Unstructured Peer-to-Peer Networks. *Proc. ACM ICS '02*.
12. Bisnik, N. and Abouzeid, A. (2005). Modeling and Analysis of Random Walk Search Algorithms in P2P Networks. *Proc. HOT-P2P 2005*.
13. Gkantsidis, C., Mihail, M., and Saberi, A. (2005). Hybrid Search Schemes for Unstructured Peer-to-Peer Networks. *Proc. IEEE Infocom 2005*.
14. El-Ansary, S., Alima, L.O., Brand, P., and Haridi, S. (2003). Efficient Broadcast in Structured P2P Networks. *Lecture Notes in Computer Science*, Vol. 2735, pp. 304-314.
15. Jagadish, H.V., Ooi, B.C., and Vu, Q.H. (2005). BATON: a Balanced Tree Structure for Peer-to-Peer Networks. *Proc. VLDB 2005*, pp. 661-672.

Global Journal of Computer Science and Technology Volume XI Issue XI Version I July 2011

This page is intentionally left blank