A Topology-Aware Improvement on Chord

Zhou Xiaofan[1], Yang Xudong[1], Wang Zhiqian[1]
[1] Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia,
Beijing University of Posts and Telecommunications(BUPT), Beijing 100876, China
zhouxiaofan@gmail.com

ABSTRACT: Routing efficiency is the critical issue when constructing peer-to-peer overlay. However, Chord has often been criticized on its careless of routing locality. In this paper, we propose a topology-aware improvement on the basis of Chord, which focuses on achieving better routing efficiency. It has lower latency by appending a topology-aware Chord circle to divide topology regions. The simulation shows that the improvement has achieved lower latency per message routing.

KEYWORDS: P2P[1]; Chord[2]; Topology-Aware[3]

I. INTRODUCTION

A P2P (peer-to-peer) network uses diverse connectivity between participants in a network and the cumulative bandwidth of network participants rather than conventional centralized resources where a relatively low number of servers provide the core value. P2P networks are typically used for connecting nodes via largely ad hoc connections. Such networks are useful for many purposes. Sharing content files containing audio, video, data or anything in digital format is very common, and real-time data, such as telephony traffic, is also passed using P2P technology.

By far the most common type of structured P2P network is the distributed hash table (DHT), in which a variant of consistent hashing is used to assign ownership of each file to a particular peer, in a way like a traditional hash table's assignment of each key to a particular array slot. Some well known structure network, such as Chord[10], Kademlia[7] and CAN[8], build on top of DHTs and afford a query complexity of O(log N).

Chord is one of the original distributed hash table protocols, which is being developed at MIT by Stoica, Morris et al. Attractive features of Chord include its simplicity, provable correctness, and provable performance even in the face of concurrent node arrivals and departures[3]. However the routing table of Chord is not based on topological feature, but based on hash id. The difference of delays between logical neighbors is ignored in Chord, but it is often so big to have a great impact on system[9].

To solve the problem, a number of Chord-like protocols has been proposed, such as PChord[1], EpiChord[4], AChord[5]. This paper describes a topology-aware improvement algorithm about building sub-circles on Chord. It chooses physical neighbor nodes, in which the latency is likely to be lower, to form a upper sub-circle based on the original Chord circle. By different levels of topology division, we can easily building a hierarchy overlay network. Like the Cache structure in computer architecture, we put low-latency nodes on higher levels and access them first to reduce the system latency of Chord query.

II. ALGORITHM MODEL AND IMPROVEMENT

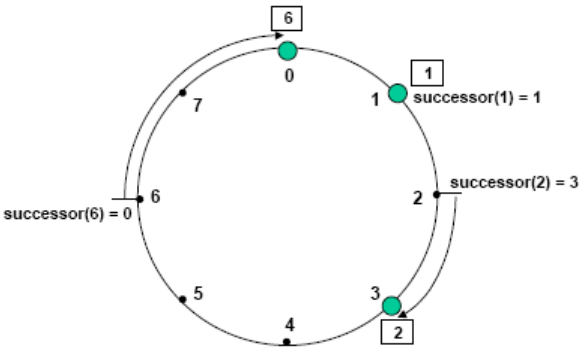

Fig.1 An identifier circle consisting of the three nodes 0, 1 and 3. In this example, key 1 is located at node 1, key 2 at node 3, and key 6 at node 0.

A. The Base Chord Protocol

Using the Chord[10] lookup protocol, node keys are arranged in a circle. The circle cannot have more than 2^m nodes. The ring can have ids/keys ranging from 0 to 2^{m-1}.

IDs and keys are assigned an m-bit identifier using what is known as consistent hashing. The SHA-1[2] algorithm is the base hashing function for consistent hashing. The consistent hashing is integral to the probability of the robustness and performance because both keys and IDs (IP addresses) are uniformly distributed and in the same identifier space.

Each node has a successor and a predecessor. The successor to a node or key is the next node in the identifier circle when you move clockwise. The predecessor of a node or key is the next node in the id circle when you move counter-clockwise. For example, the successor of node 2 is node 3, and the predecessor of node 1 is node 0.

For efficiently searching, Chord uses a finger table to partition the circle nodes into 1+logN segments. Each node has its own finger table, using O(log N) memory to maintain the Chord circle information.

B. Problem of Chord and Topology-Aware Improvement

Chord maps its nodes to an artificial one-dimensional space, lookups keyword by uniformly identifier. A node's identifier is chosen by hashing the node's IP address, while a key identifier is produced by hashing the key. So the logical distance between physical neighbor nodes is random.

978-0-7695-3600-2/09 $25.00 © 2009 IEEE
DOI 10.1109/IFITA.2009.375

Then there is a problem here, the hops of these routing algorithm are logical rather than physical. A logical hop usually contains a lot of physical hops. In a large P2P network, the difference of latency between two peers is significant. Most likely a logical hop happens in the two nodes with great delay, which probably cause long-distance transmission and block the network.

Using topological neighbor nodes can reduce DHT routing delay efficiently, which has been proved by Ratnasamy et al.[6,11] According to this point, this paper presents an improved algorithm: Partition nodes according to topology attribute and build a multi-circles structure, so that most of lookups will be completed in the upper circles with low latency. Then the system latency of query will be significantly reduced with initial hops.

In order to simplify the algorithm description, we only discuss two circles situation in this paper.

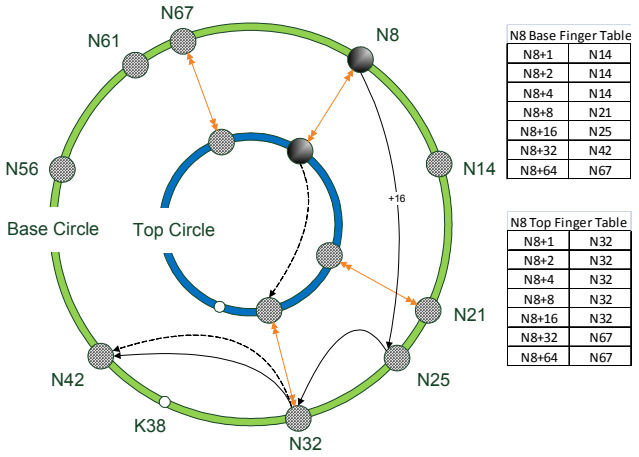

Fig.2 Node N8 launches a query on K38, the black lines for the original Chord lookup routing, and the dashed lines for improved lookup routing.

As shown in Figure 2, the outer green circle is called Base Circle(original Chord circle) and the inner blue one is called Top Circle(topology-aware). Each node on Top Circle has additional routing table called top finger table. The routing of query message has two steps: First, use the top finger table to approach the target node through physical neighbor nodes. Second, use the base finger table to locate it.

Based on the description above, we can express the improved algorithm as follows:

1. Node n launches a query on the Keyword k. In other words, lookup the successor of id. id=Hash(k).
2. Determine whether the successor(on Base Circle) of the current node is also a successor of Node id. If it is, return. The query ends.
3. Determine:
   a. If the current node is on Top Circle, find the highest predecessor of id from the top finger table, then set it as the current node, back to Step 2.
   b. If not, find the highest predecessor of id from the base finger table, set it as the current node, back to Step 2.

Pseudocode:

Definitions:

- finger[k]: first node that succeeds.
- successor: the next node from the node in question on the identifier ring.
- predecessor: the previous node from the node in question on the identifier ring.

```
// ask node n to find the successor of id
n.find_base_successor(id)
  if (id in (n, n.base_successor])
    return n.base_successor;
  else
    // forward the query around the circle
    n0 = n0.closest_preceding_node(id);
    return n0.find_successor(id);

// search for the closest predecessor of id
n.closest_preceding_node(id)
  n1 = n.top_closest_preceding_node(id);
  if ( n1 == n)
    return
      n.base_closest_preceding_node(id);
  else
    return n1;

// search the base finger table
n.base_closest_preceding_node(id)
  for i = m downto 1
    if (base_finger[i] in (n,id])
      return base_finger[i];
  return n;

// search the top finger table
n.top_closest_preceding_node(id)
  for i = m downto 1
    if (top_finger[i] in (n,id])
      return top_finger[i];
```

Fig.3 The pseudocode to find the successor node of an identifier id. Remote procedure calls and variable lookups are preceded by the remote node.

III. EVALUATION

We claim that the query complexity of the improvement is the same as Chord, but with lower latency. This is achieve by query physical neighbor nodes first, which probably has low latency, to reduce the average latency of hops. This sounds reasonable but needs experiments to prove.

We design and running the experiment using Peersim[6] simulator, which is a component-based simulation environment for P2P protocols. It is composed of two simulation engines: a simplified (cycle-based) one and event driven one. In this experiment, we use the event-based engine for more realistic.

A. Experiment Setup

The simulated network is initialized with 5000 nodes on Base Circle, including 1000 low-latency nodes on Top

Circle. The simulation runs in the churn environment where nodes join and leave randomly(total of nodes between 3000 and 7000). The experiments are querying the network every 100 unit times with the randomly chosen sender and target, and printing results every 200,000 unit times. After 1,000,000 unit times, the experiment ends.

The experiments sum the query latency and hops and calculate the averages the two algorithms used.

B. *Simulation Results*

Definitions:

Delays: the total latency a query travels through the overlay network to reach an object.

Hops: numbers of nodes that must be visited to solve a query.

Groups: experiments with different random seeds.

No: sequence numbers of results..

TABLE I. DELAYS AND HOPS

Groups	A (Random seed=1211175965593)				B (Random seed=1211177902658)			
No.	1	2	3	4	5	6	7	8
Average Delays								
original Chord	26707	26588	27100	26500	26367	27190	27036	26699
improved Chord	23693	23533	23986	23979	23556	23404	23704	23807
Max Delays								
original Chord	85653	75404	74232	71819	77763	78350	82314	75544
improved Chord	70967	65349	72255	70572	68836	75962	70653	73861
Hops(avg/max/min)								
original Chord	5/13/0	5/12/0	5/14/0	5/12/0	5/13/0	5/13/0	5/13/0	5/14/0
improved Chord	5/13/0	5/12/0	5/13/0	5/12/0	5/13/0	5/13/0	5/13/0	5/13/0

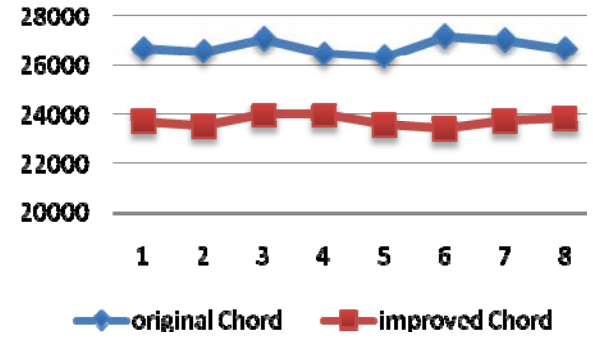

Fig.4 Delays and Hops

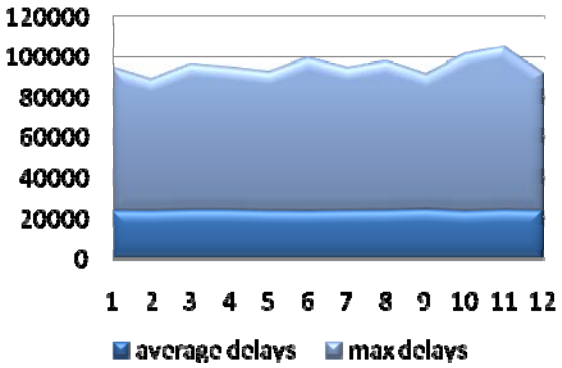

Fig.5 Delays of Improved Chord

C. *Comparative Analysis*

Table 1 gives a comparison of the original and the topology-aware improved Chord algorithms. The equality of hops in both algorithms shows the improvement has nothing to do with reducing hops.

Figure 4 shows that the improved Chord achieves a better performance with 11% lower delays. It indicates the topology-aware algorithm can be effective in reducing the routing latency for query. Because the delays in physical neighbor nodes are likely to be lower.

Figure 5 shows that the average delays in the improved Chord is relatively stable, compared to the max delays with randomly changes. Chord may contact at most O(log N) nodes to find a successor in an N-node network, with high probability. Based on Chord, the improvement only has an O(log N) additional cost in memory to maintain the sub-circle information, and limited impact on the complexity of algorithm. The improvement does not affect the system performance, but significantly increased the efficiency of query.

D. *Remaining Issues and Solutions*

How to choose a sub-circle: In the simulation above, we just choose low-latency nodes to compose the sub-circle. But there will be a problem in practice, peers must have a given way using the local information about the network to find physical neighbor nodes and determine whether themselves belong to a known sub-circle. Of course, peers have to pass authentication before joining a sub-circle.

For choosing sub-circle, We list feasible schemes as follow:

- Get the local DNS-suffix of peers.
- Measure IP addresses.
- Select manually by client.
- Choose the sub-circle with minimum delays in known circles.

Problems of Multi-Circles: In a large P2P network with numbers of sub-circles, peers perhaps belong to different circles at the same time. Each circle should have a rank for query routing. And there will be more problems to be solved when nodes join and leave.

We leave these issues to resolve in the future.

IV. CONCLUSION

In this paper, we propose a topology-aware improvement on the basis of Chord, which focuses on achieving better routing efficiency. It has low latency by choosing physical neighbor nodes to form a upper sub-circle based on the original Chord circle.

The improvement successfully reduces the latency in the query. This may be important in some kind of the applications, such as collaboration, files share and media stream transportation.

ACKNOWLEDGMENT

I would like to thank my best friend Rui Hu for teaching me a lot.

REFERENCES

[1] Feng Hong, Minglu Li, Jiadi Yu, Yi Wang. "PChord: Improved on Chord to Achieve Better Routing Efficiency by Exploiting Proximity", in Proceeding of the 25th IEEE International Conference on Distributed Computing System Workshops(ICDCSW'05), 2005.

[2] FIPS 180-2. "Secure Hash Standard", U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, 2002.

[3] HU Ying-song, GUO Shou-lie. "An Extended Algorithm for Hierarchical Low-Latency Chord Protocols", Computer Engineering & Science, Vol.99, No.4, pp.74-77, 2007.

[4] Leong Ben, Liskov Barbara, Demaine Erik D. "EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management", MIT Technical Report MIT-LCS-TR-963, Cambridge, MA, 2004.

[5] Le Hai Dao, JongWon Kim. "AChord: Topology-Aware Chord in Anycast-Enabled Networks Hybrid Information Technology", ICHITapos 06, Vol.2, Issue Nov. 2006 pp.334-341, 2006.

[6] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, Spyros Voulgaris. "Peersim", http://peersim.sourceforge.net/, 2007.

[7] Petar Maymounkov, David Mazières "Kademlia: A Peer to peer information system Based on the XOR Metric", in Proceedings of IPTPS02, Cambridge, USA, 2002.

[8] Ratnasamy S, Francis P, Handley M, et al. "A Scalable Content-Addressable Network", in Proceedings of ACM SIGCOMM 2001, San Deigo, CA, 2001.

[9] Ratnasamy S, Handley M, Karp R, et al. "Topologically-Aware Overlay Construction and Server Selection", INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol.3, pp. 1190-1199, 2002.

[10] Stoica, Morris et al. "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications", in Proceedings of ACM SIGCOMM 2001, San Deigo, CA, 2001.

[11] Yi Jiang, Jinyuan You. "A Low Latency Chord Routing Algorithm for DHT", 2006 1st International Symposium on Pervasive Computing and Applications, 2006.