CHOP6: A DHT Routing Mechanism Considering Proximity

Shouta Morimoto Fumio Teraoka
Graduate School of Science and Technology, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama 223-8522, Japan
{monmon,tera}@tera.ics.keio.ac.jp

Abstract

This paper proposes a DHT routing mechanism called CHOP6. CHOP6 takes proximity into account by using the properties of the IPv6 address format. CHOP6 introduces the 64-bit node ID, in which the lower part is the IPv6 global routing prefix. By examining the node ID, it is possible to roughly estimate the proximity to the node. Similar to Chord, CHOP6 uses the finger table. In CHOP6, the finger table entry holds more than one candidate nodes. When sending a query, the source node estimates the proximity to the candidate nodes by examining the node IDs if the RTT to the nodes is unknown. CHOP6 was implemented in Java on FreeBSD. The measured results showed that the time to get data in CHOP6 is less than that in Chord and the number of the messages necessary to maintain routing information in CHOP6 is also less than that in Chord.

1 Introduction

In recent years, peer-to-peer (P2P) technologies are widely used in a variety of services. MSN Messenger and Skype are typical applications based on P2P technologies. On-line games using P2P technologies are also getting popular. Although these applications require realtime property, most of current DHT algorithms, which are one of the core technologies in P2P, do not take proximity of nodes into account. There are several proposals that take account of proximity. However, some of them require extra messages other than routing to learn proximity and some of them require extra messages when a node joins. Thus, it is difficult to meet the realtime requirements in existing DHT algorithms.

This paper proposes *CHOP6*, a DHT routing mechanism based on Chord which takes account of proximity of nodes and avoids extra routing messages. CHOP6 can make message routing time shorter by sending messages to a closer next hop node. It can also make the system more scalable by avoiding transmission of redundant control messages. CHOP6 introduces an ID format in which the network prefix of the IPv6 address is used and estimates proximity based on the prefix and RTT. CHOP6 was implemented in Java on FreeBSD. The measured results showed that the time to get data in CHOP6 is less than that in Chord and the number of the messages necessary to maintain routing information in CHOP6 is also less than that in Chord.

2 Related Work

Kademlia[2] is a DHT based on the XOR metric. It uses the XOR metric to learn the distance between nodes or keys and processes routing queries asynchronously. A node sends routing queries to α nodes in the most appropriate sub-tree, where α is a system-wide parameter. The source node processes only the reply message that reaches the source node first and ignores the other reply messages. It is expected that the source node can quickly receive a reply from the node which is close to the source node. However, reply messages other than the first one are discarded. This increases the network load and results in lack of scalability.

In GNP[3], the internet space is considered as a coordinate space and each node is given a coordinate. A node can know the distance to another node without direct communication. When a node joins GNP, it connects to the "Landmarks" to learn its relative position. However, since GNP needs several messages not required in routing, it is not suitable to a DHT where nodes frequently join and leave.

3 Design of CHOP6

CHOP6 (CHOrd considering Proximity on ipv6) is a DHT routing mechanism considering proximity on IPv6 and solves problems mentioned in Section 2. It is based on Chord [1] that does not consider proximity into account.

3.1 Architecture of CHOP6

The behavior of CHOP6 is basically the same as that of Chord except to use the properties of the IPv6 aggregatable

Proceedings of the 2007 International Symposium
on Applications and the Internet Workshops (SAINTW'07)
0-7695-2757-4/07 $20.00 © 2007 IEEE

global unicast address (IPv6 address, in short) format and the RTT to the candidate nodes to estimate the proximity to them. CHOP6 introduces a 64-bit ID. The ID format of CHOP6 is different from that of Chord. Similar to Chord, the data is stored in the successor node in CHOP6. The 64-bit key of the data is obtained by calculating the hash value of the keyword of the data, and then the successor is determined by the key.

In CHOP6, a finger table entry holds k ($k \geq 1$) candidate nodes while a finger table entry holds a single node in Chord. k is a system-wide parameter. When sending a query, the source node selects the closest node among the candidate nodes in the finger table entry for the next hop.

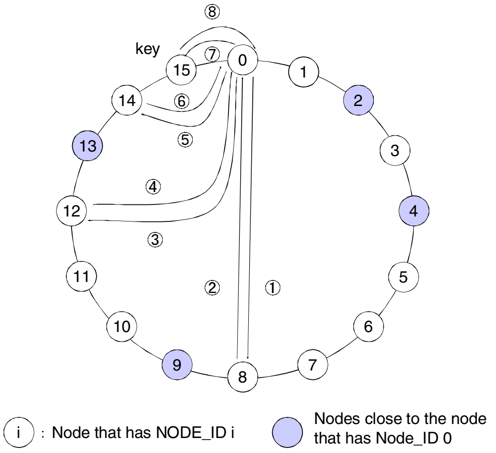
⑧
key
⑦
15
0
1
14
⑥
2
13
⑤
3
④
12
4
③
11
②
①
5
10
6
9
7
8
i : Node that has NODE_ID i
Nodes close to the node that has Node_ID 0

Figure 1. Behavior of Chord

The behavior of Chord is depicted in Figure 1 for the sake of comparison. For simplicity, 4-bit IDs are used in this example. In this example, *Node 0* sends a query to *Node 15* for *key 15*, where *Node 0* means that the node has *ID 0*. Each node in Chord has the finger table. In this example, *Node 0* has entries of the nodes whose IDs are apart from its ID by 1, 2, 4, and 8. *Node 0* selects the closest node to the target node in the ID space. Thus, it takes long time to send queries if the network distance between nodes is long.

Figure 2 shows the behavior of CHOP6, where 4-bit IDs are used and $k = 2$. Since $k = 2$, each finger table entry holds two candidate nodes. For example, the finger table of *Node 0* holds the information about *Node 1, 2, 3, 4, 5, 8*, and *9*. It is expected that the query transmission time in CHOP6 is shorter than that in Chord because the source node can select the closest node among the candidates in the finger table entry for the next hop.

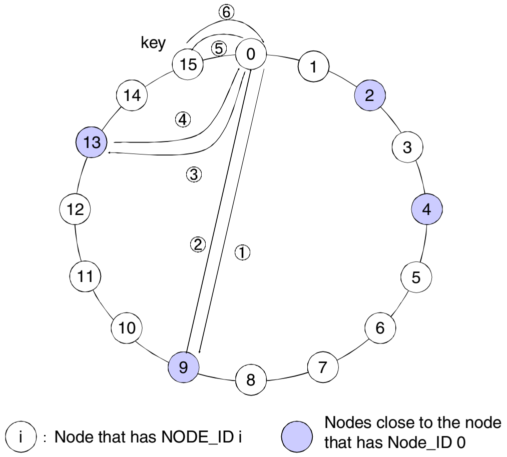
⑥
key
⑤
15
0
1
14
2
④
13
3
③
12
4
②
①
11
5
10
6
9
8
7
i : Node that has NODE_ID i
Nodes close to the node that has Node_ID 0

Figure 2. Example of CHOP6 operation

3.2 Features of the IPv6 address

The IPv6 global aggregatable unicast address consists of the 3-bit format prefix (001), the 45-bit global routing prefix, the 16-bit subnet ID, and the 64-bit interface ID. CHOP6 focuses on the global routing prefix of the IPv6 address. In this paper, the higher 64-bit of the IPv6 address is called *the network part*.

The IPv6 address blocks of /16 or /23 in size are assigned to RIRs (Regional Internet Registry) by IANA (Internet Assigned Numbers Authority). An RIR divides the assigned address blocks into small address blocks (/32 at minimum) and assigns them to NIRs (National Internet Registry). Thus, it is possible to estimate the country or the region to which the node is connected by examining the upper 32-bit of the IPv6 address of the node. In CHOP6, such address assignment information is maintained in each node as the *address assign table*, which is used to estimate proximity.

3.3 Node ID Format

CHOP6 makes use of the IPv6 address structure described in the previous section. Figure 3 shows the node ID format in CHOP6. The upper $(32 - \alpha)$-bit is the *Random ID* part and the lower $(32 + \alpha)$-bit is the *IPv6 prefix* part, where α is a system-wide parameter ($-32 < \alpha < 32$). Similar to Chord, the Random ID part is assigned the hash value of the IP address of the node and the port number.

$(32-\alpha)$ *bits*	$(32+\alpha)$ *bits*
Random ID	IPv6 prefix

Figure 3. Node ID format in CHOP6

3.4 Routing Mechanism

The basic routing mechanism of CHOP6 is the same as that of Chord. When a node (*Node A*) searches for the node which has some key, it selects the appropriate node (*Node B*) in its finger table and sends the FIND_SUCCESOR_REQUEST message to *Node B*. When *Node B* receives the FIND_SUCCESSOR_REQUEST message, it returns the FIND_SUCCESSOR_REPLY message including the ID of the more appropriate successor to *Node A*. Upon receiving the FIND_SUCESSOR_REPLY message, *Node A* sends the FIND_SUCESSOR_REQUEST message to the node selected by referencing the FIND_SUCESSOR_REPLY message received from *Node B*. *Node A* iterates the procedure described above until it finds the target node.

In CHOP6, proximity is estimated based on the IPv6 prefix part of the node ID and the RTT to the node. A node can know the node ID of the next hop node before communicating with it while the RTT to the next hop node cannot be learned before communicating with it. Therefore, there are three cases when a node selects the most appropriate next hop node among k candidate nodes in the finger table entry. The three cases are described bellow.

3.4.1 Case 1: All RTT to k candidate nodes is unknown

This is the initial state when a node joins CHOP6. The source node selects a node connected to the same area by examining the node IDs of the k candidate nodes in the finger table entry. If there is no node in the same area, the source node selects a node whose node ID matches to the node ID of the source node as long as possible.

3.4.2 Case 2: Some RTT is known

After communicating with some nodes, the source node learned the RTT to some nodes in its finger table. According to the procedure of Case 1, it is expected that the nodes whose RTT is known to the source node are appropriate for the next hop. However, since this is not always true, the RTT to other candidates must be measured.

In this case, the source node selects the node whose RTT is the smallest among the candidate nodes with probability p $(0 \leq p < 1)$. If there is a node whose RTT has not been measured, the source node selects such a node with probability $1 - p$. If there are more than one nodes whose RTT has not been measured, the source node selects the next hop node based on the node ID as described above.

3.4.3 Case 3: All RTT is known

In this case, the source node has already communicated with all k candidate nodes. The source node selects the node whose RTT is the smallest with probability p, selects the node whose RTT is the 2nd smallest with probability q, and so on. At last, the source node selects the node whose RTT is the largest with probability $(1 - p - q - ...)$, where

$$0 \leq ... < r < q < p < 1$$
$$p + q + r + ... \leq 1$$

The reason why a node whose RTT is not the smallest is selected with some probability is that the RTT changes time by time. The RTT to the candidate nodes is measured periodically and the *estmatedRTT* is calculated from the *currentRTT* and the *newRTT* as follows:

$$estimateRTT = w \times currentRTT + (1-w) \times newRTT$$

where w is a weight.

4 Evaluation

We implemented CHOP6 and Chord in JAVA on FreeBSD.

4.1 Evaluation environment

Figure 4 depicts the test network which consists of ten PCs: four PCs as routers and six PCs as host machines. In the figure, "fxp$n$" means a network interface. 100msec delay was added to packets forwarded by router R1 and 30msec delay was added to packets forwarded by routers R2, R3, and R4 by using dummynet. In this topology, R1 is regarded as the global Internet. R2, R3, and R4 are regarded as regions such as Asia-Pacific and North America. H1 to H6 are regarded as countries. As a result of such configuration, no delay was added to the communication within a country, 60msec delay was added to the communication within the same region, and 320msec delay was added to the communication between regions.

4.2 Evaluation method

Several CHOP6 nodes (processes) were running on each physical machine. Each node stored 10 kinds of data in the system. The time to get the all data was measured by changing the parameter k from 1 to 6 and changing the number of nodes on a machine N in 1, 2, 4, and 8. When $k = 1$, CHOP6 is the same as Chord.

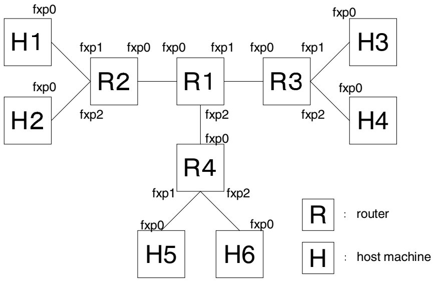

Figure 4. Test network topology

4.3 Evaluation result and inspection

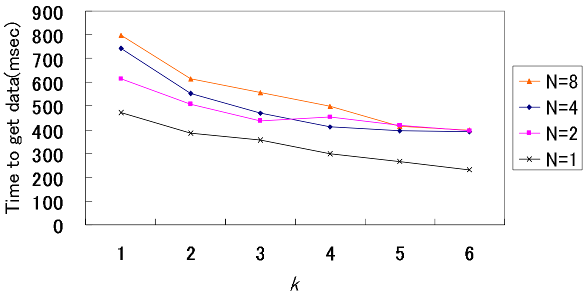

Figure 5. Evaluation result

Figure 5 shows measured results. As shown in the figure, the larger the parameter k becomes, the shorter the time to get data is required for all N. This is because CHOP6 can select a closer node as the next hop by holding more than one candidate nodes in the finger table entry.

When the parameter N changed in 1, 2, 4, and 8 with $k = 1$, the time to get data also increased. Since the hop count required in Chord is O($log\ N$), the required hop count increases by one if the number of nodes becomes twice. However, when $k = 6$, as N becomes large, the time to get data does not increase as much as $k = 1$. This is because the larger k and N are, the higher the probability to select a closer node becomes.

4.3.2 Discussion on the number of messages

This section discusses the number of messages when k becomes larger in Chord and CHOP6. CHOP6 does not introduce any new messages. The discussion focuses on the number of the messages necessary to maintain the successor list, the predecessor, and the finger table for routing, and also the number of the messages to get data.

It seems that the larger k becomes, the more STABILIZATION_REQUEST and FIX_FINGER_REQUEST messages are sent to stabilize or fix the finger table. However, there is no difference in the number of the messages necessary to stabilize or fix the finger table between CHOP6 and Chord. The reason is as follows. In CHOP6, the node sends the FIX_STABILIZATION message only to the first node in the successor list, not to all nodes in the successor list. When a node receives the reply of the STABILIZATION_REQUEST message, it updates its finger table by using the successor list included in the reply. In case of the FIX_FINGER_REQUEST message, the node sends the message to the first node in the finger table entry and updates the finger table by using the successor list included in the reply message. However, the size of the message is k times as large as that in Chord.

In CHOP6, the number of the messages to get data is less than that of Chord because the hop count required to get data in CHOP6 is less than that in Chord. Therefore, CHOP6 requires less messages than Chord.

5 Conclusion

This paper proposed a new DHT routing algorithm called CHOP6 considering proximity between nodes in the IPv6 Internet to meet the requirements of realtime applications. CHOP6 introduces the 64-bit node ID. The lower part of the ID includes a part of the global routing prefix part of the IPv6 address of the node. The finger table entry in CHOP6 holds k ($k \geq 1$) candidate nodes. The next hop node is selected among the k nodes based on proximity. Proximity to a node can be estimated by measured RTT. If the RTT is unknown, proximity is estimated by the global routing prefix part of the IPv6 address in the node ID. CHOP6 was implemented in JAVA on FreeBSD. The measured results showed that the time to get data in CHOP6 is shorter than that in Chord. In addition, the number of messages necessary to maintain the routing information in CHOP6 is also less than that in Chord.

References

[1] S. I., M. R., K. D., K. M. F., and B. H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160, August 2001.

[2] M. P. and M. D. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of the 1st IPTPS*, pages 53–65, March 2002.

[3] N. T. S. E. and Z. H. Predicting internet network distance with coordinates-based approaches. In *Proceedings of the IEEE INFOCOM 2002*, pages 170–179, 2002.