# A Low Latency Chord Routing Algorithm for DHT

Yi Jiang, Jinyuan You
*Dept. of Computer Science and Engineering, Shanghai Jiao Tong University*
*Shanghai, P.R.China, 200030*
*jiangyi@cs.sjtu.edu.cn, you-jy@cs.sjtu.edu.cn*

## Abstract

*In this paper, we propose a low latency routing algorithm for structured peer-to-peer network. The LLCHORD(A Low Latency Chord Routing Algorithm for Distributed Hash Tables) protocol is used in the routing query message to the destination. Previous protocol, such as Chord, already provides a scalable query latency of O(log(N)), where N is the number of nodes in peer-to-peer network. But the latency of the each hop is omitted in the analysis of the routing performance. In LLCHORD protocol, while retain the O(log(N)) routing latency, the routing is aimed at reduce the cost of each hop. At this way, the LLCHORD routing performance of LLCHORD protocol is better than Chord. In LLCHORD protocol, the cost of the each step to the destination node is reduced. Unlike the general Chord protocol, the next hop of the query message can be in the clockwise and counter clockwise side of the destination. In this way, the closing up process to the destination key is in dual directions.*

**Keywords:** peer-to-peer, dual direction finger table, low latency, routing, distributed hash table.

## 1. Introduction

Peer-to-peer means scalable and self-organizing. In peer-to-peer network, each node hash similar functionalities and plays the role of a server and a client at the same time. Peer-to-peer system can be categorized into structured and unstructured. The first generation of peer-to-peer network is hybrid peer-to-peer network; it requires a centralized directory service. Query and searching in those systems is simply and high efficient. The unstructured peer-to-peer network used to use flooding to query the network. This always leads to network congestions. The structured peer-to-peer network, such as Chord[1], Pastry[2], Tapetry[3], build on top of distributed hash tables solved the problem and afford a lookup complexity of *O(log(N))*. The structured peer-to-peer network is idea way to build peer-to-peer application system on top of it. In this paper, we refer peer-to-peer network as structured peer-to-peer network.

In this paper, we address two problems facing the LLCHORD protocol in distributed peer-to-peer network.

- How to reduce the average hops needed by query in Chord?
- How to reduce the query latency in Chord?

To solve the first problem, we use a simple latency-sampling algorithm to trace the latency of the each hop and use the trace results to coordinate the routing of query message. Every hop there are two choices, the routing algorithm choose the one that is closest to local node in network distance to reduce the routing latency.

To solve the second problem, we adapt the finger table in Chord to a dual directional finger table in which the fingers in clockwise side and in counter clockwise side of the local node are stored.

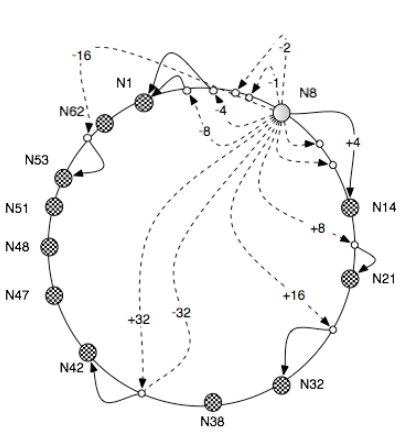
## 2. Related works

The matching of the overlay topology and underlying physical network cause serious problem in peer-to-peer computing. The mismatching exists not only in structured peer-to-peer network but also unstructured peer-to-peer network. To overcome the bad effects of the mismatch, many methods are used to probe the physical network topology. IDMap[7] use tracers to measure the latency between the clients and advertise the measurements to the clients.

GNP[8] (Global Network Positioning) models the Internet as a geometric space and distributed computes geometric coordinates to characterize the positions of hosts in the Internet. Landmark nodes measure the RTT(round-trip time) among themselves and use this information to compute a coordinates in a Cartesian space for each of them. These coordinates are then distributed to clients, which measure RTTs to landmark nodes and compute a coordinates in the Cartesian space for itself, based on the RTTs and the coordinates of landmark nodes.

Those works mainly focus on how to measure the network distance, and those approaches with partially client server feature, that is they need centralized node or a third-party node to perform the measurement. Also, some of the methods need to prorogate the routing information frequently. The traffic generated is not welcome in the structured peer-to-peer system.

The other system, such as Topologically-Aware CAN[9], builds the topology constrained by underlying network topology. This type of the topology may cause the uneven of the distribution of the key in key space and thus increase the chances of the load unbalancing.




N8' Figer Table

| | |
|---|---|
| N8-32 | N38 |
| N8-16 | N53 |
| N8-8 | N1 |
| N8-4 | N1 |
| N8-2 | N1 |
| N8-1 | N1 |
| N8+1 | N14 |
| N8+2 | N14 |
| N8+4 | N14 |
| N8+8 | N21 |
| N8+16 | N32 |
| N8+32 | N42 |

N8' Neighbor Table

| | |
|---|---|
| N1 | 1ms |
| N14 | 2ms |
| N21 | 2.2ms |
| N32 | 2.32ms |
| N38 | 3ms |
| N42 | 3.5ms |
| N53 | 3.9ms |
| ... | ... |

Figure 1. The finger table and neighbor table in LLCHORD protocol. The finger table is the same as the finger table in Chord in structure, but the finger in LLCHORD pints to two directions, the clockwise direction and the counter clockwise direction. In LLCHORD, the finger table is used as the routing table. The neighbor table is the table that indicates the distance between the local node and the fingered node. The routing of the query message is separated into two steps in LLCHORD.

In our system, there are no needs for centralized nodes or third-party node; every node knows only the approximate distances between the local node and the finger node. This avoids the messages that are used to probe the network to form a clear view of the topology.

## 3. Backgrounds

### 3.1. Measurement of network distance

According to [12], although the overall distance estimating accuracy is high, the relative errors are larger than 0.25 for more than 20% of the estimations. The network distance is dynamic and changing, the distance is coarse. So, we choose not to build the overlay topology from the physical topology. But adapts the existing topology to the latency matrix of the nodes, when the distance has changed, the routing is changed too. On the other hand, in the LLCHORD protocol, only the distance information of the relative static set of nodes is need, so the sampled distance is enough.

In our system, we also think to define the distance as the hops in IP level when routing a packet from one host to another. But this is too simple to reflect the distance between hosts. So the distance between two network hosts is defined as the round-trip time of the two hosts. This round-trip time is the sum of the delays on the path at IP level.

In LLCHORD, the distance between two nodes can be obtained from network coordinates system, such as GNP, or can be measured by piggyback QoS data into each query message.

### 3.2. The Chord DHT routing algorithm

Chord is the most important structured peer-to-peer routing algorithm until now. The keys of the DHT are *l*-bit identifiers, i.e., integers in the range *[0, $2^l$-1]*. They form a one dimensional identifier circle modulo $2^l$ wrapping around from $2^l - 1$ to *0*.

Every node in Chord is assigned a *l*-bit node ID, the node ID is the hash value of the IP address of the node. The data record *(K, V)* in the Chord has a unique key P, this P is a *l*-bit too, P is calculated by *hash(K)*. The P is used to identify the data record *(K, V)*.

The overlay network of Chord is a virtual cycle contains $2^l$ nodes. The node is mapping into the virtual cycle by its node ID. For each node ID, the first physical machine on its clockwise side is called its successor node, or *succ(nodeID)*. The data record *(K, V)* has an identifier *P = hash(K)*, the P is designate the position of data in the virtual cycle. The data record *(K, V)* is stored in the first physical machine clockwise from *P*. This machine is called the *successor node* of *P*, or *succ(P)*.

In the Chord implementation, for the efficiency of searching, Chord use finger table to partition the virtual cycle into *1+logN* segments. This finger table let each machine use *O(log(N))* memory to maintain the topology information.

The Chord protocol is mainly a query protocol, it supports only one operation: given a key, it maps the key onto a node. When node searches a resource, it first calculates the hash value of the resource name (the key) and then determines the position of the resource in the virtual cycle of node. In Chord implementation, there are two modes of query, iterative and recursive. In iterative mode, the machine proposes the query communicate with all the nodes in the finger table. In the recursive mode, the machine proposed the query only communicate with its successor, and the query is forward by the successor node until it is reach the destination.

## 4. LLCHORD Routing Protocol

In the LLCHORD routing protocol, the dual directions finger table and the neighbors table is used to reduce the query latency. In the worst case, the Chord algorithm has a routing complexity of *O(log(N))*, when the node numbers increasing and eventually exceeds a limit, such as $2^{10}$, the routing to specific target may have a undesirable latency. More badly, the effects of the mismatch between the overlay network and the physical world can cause the routing latency raise to a high level.

While the hash used in Chord is also used in LLCHORD, the key space of the LLCHORD protocol

```
n.find_successor(id)
  if id in fingers then
     return fingers[id]
  else
     n = next_hop(id)
     return n.find_successor(id)

n.next_hop(id)
  d1 = |id - local|
  if d1 ≤ N/2 then direction = cw
  else direction = ccw

  if direction == cw then
     for i = m downto 1 do
        if (finger[i] ∈ [n,id]) then
           ff1 = finger[i]
           if i<m then ff2 = i+1
           else ff2 = null
        end if
     end for
  else if direction = ccw then
     for i = -m to -1 do
        if (finger[i] ∈ [n;id]) then
           ff1 = finger[i]
           if i>-m then ff2 = i-1
           else ff2 = null
        end if
     end for
  end if
  if d(ff1,n) > d(ff2,n) return ff2
  else return ff1
```

Figure 2. The pesudocode for find_successor using finger table and neighbor table.

is the same as the Chord, the dual directional finger table and the neighbor table is the biggest difference between the two protocols.

### 4.1. Routing table in LLCHORD protocol

The routing table in LLCHORD is composed of the links to the neighbor nodes and the finger nodes. The fingers are used to reduce the average hops when routing the query. The neighbors are used to reduce the each hop latency when routing message. The mix using of the two types of the routing tables in LLCHORD protocol produce a low latency Chord protocol. The routing table is shown in Fig.1.

### 4.2. The dual directional finger table

The fingers in the LLCHORD finger table points to the clockwise direction and the counter clockwise direction. The dual directional finger table is designed to reduce the routing overhead when we change the generic chord protocol. In LLCHORD protocol, the query message may traverse the intermediate nodes in order way. For example, the query message traverses the way {N8, N10, N11, N21} to its destination. In this path, the N8 is the source node, which initializes the query, N21 is the destination. The N10 and N11 are the mid-way nodes. And the N8<N10<N11<N21 in the key space. This is how chord route the message. But in LLCHORD, the path the message traversed doesn't obey the rule that the next hop node is in the clockwise direction of the source node. In LLCHORD, the only thing guaranteed is through each hop, the $D = |N_{nexthop} - N_{destination}|$ is reduced. Considering the facts that the D is reduced every hop, so the message is forwarded to the destination finally in limited hops.

The design of the dual directional finger table is try to reduce the message overhead when we broken the rule that every next hop is in the clockwise side of the source node. We adapt the routing protocol to use the dual directional finger table to achieve the aim.

### 4.3. The neighbor table

The neighbor table is the table that records the distances between the fingers and the local node. The core of the LLCHORD protocol is to reduce the routing latency in Chord. The neighbor table can be used to assistant the choice of the next hop. The definition of the routing table is shown in Table 1.

The routing in LLCHORD is divided into two main steps. If the destination key is in the neighbor table, then the query is routed to the node directly. This is the best condition; the one hop routing has lower latency then multi-step with high probability. Considering the fact that the neighbor table contains the node that is near in the Internet, the routing can has secondary choice. If the destination key is not in the neighbor table, then lookup the finger table, and get the successor of the key. But the successor can be the clockwise successor or the counter clockwise successor. So the next hop's node's key may be bigger or smaller then the destination key. The next hop's selection is based on the following term:

$$\min\left\{D(N_{nexthop}, N_{local}) \mid \left|K_{nexthop} - K_{local}\right| < \left|K_{nexthop} - K_{destination}\right|\right\}$$

In the above term, the notation $D(N_{nexthop}, N_{local})$ means the distance between the local node and the next step. This is the physical distance that is represented by the ping-pong time between the two nodes. In other words, $D(N_{nexthop}, N_{local})$ is the value in the neighbors' table.

The notation $\left|K_{nexthop} - K_{local}\right| < \left|K_{nexthop} - K_{destination}\right|$ means the next hop must more closely to the destination node in the key space than local node. This ensures the message is forward towards the destination.

Then, choose a key in the neighbor table that is close to the key of successor in numeric. This key is the next hop to which the query message is routed.

At this time, we choose the nearest node in the finger table as the next hop, this way can route the message to its destination. But unfortunately, we need more hops than Chord. To provide a complexity of *O(log(N))* in routing, the queries in LLCHORD must be forward at

| Notation | Definition |
|---|---|
| Finger[k] | First node on circle that succeeds $(n \pm 2^{k-1}) \bmod 2^m$, $-m \le k \le m$ |
| Neighbor[k] | The distance between the finger and the local node, $-m \le k \le m$ |

Table 1, Definition of routing tables. Notice that the fingers point to not only clockwise direction but also counter clockwise direction.

least half the remaining distance around the ring. So the next hop is chosen using the following formula:

$$\min\left\{D(N_{nexthop}, N_{local}) \mid \frac{1}{2}\left|K_{local} - K_{destination}\right| \le \left|K_{nexthop} - K_{local}\right| < \left|K_{nexthop} - K_{destination}\right|\right\}$$

Fig.2 shows the pesudocode of the *find_successor* operation. This *find_successor* operation uses the neighbor table to assistant the finding of successor.

### 4.4. The definition of the successor

As we can see in the previous section, the finger table contains the fingers that points to the node succeed the local node in clockwise side and the counter clockwise side. So the definition of the successor in LLCHORD is slightly different from the Chord.

In Chord, the successor is defined as the first node whose identifier is equal to or follows the key in the identifier space. This successor is in the counter clockwise side of the key. In LLCHORD, the key hash a pair of successors, which are defined as the node closest to the key in the key identifier space. So the key has two successors, the one is $successor_{cw}$, the other one is $successor_{ccw}$. This is the definition of the successor in LLCHORD, but the selection of the next hop will consider not only the direct successors but also the latency.

The next hop of the query is selected among the $successor_{cw}$ and $successor_{ccw}$. The one which is closest from the local node is the next hop.

### 4.5. Node joining and leaving

Node joining and leaving in LLCHORD is handled like the Chord does. But the difference is that the LLCHORD has a neighbor table. When a node joins in the LLCHORD network, it becomes one of the nodes in the ring. At this moment, the neighbor table is empty. So the node will construct the neighbor table after joined the network.

Once the node joined the ring. It will probe the node its finger pointed to and get the ping-pong time from the QoS of the message. In this way, the neighbors' table will be constructed within seconds. Before the neighbors' table is constructed, the LLCHORD routing is as the same as what Chord protocol does. When node n starts, it calls *n.join(n')*, where n' is any known Chord node to the local node.

Before a node leaves the network, it notifies the predecessor and the successor nodes about its leaving.

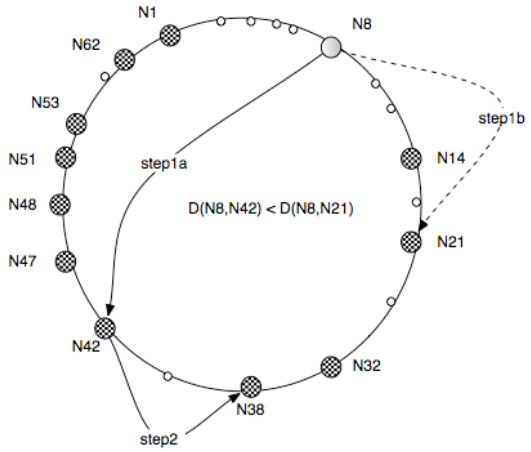



Figure 3. The choosing of the next hop in LLCHORD. N42 is chosen as the next hop instead of N21 for short network distance between N8 and N42.

Then those nodes update its finger table and invalidate the corresponding entry in the neighbor table.

### 4.6. Routing

The LLCHORD routing message is passed through many intermediate nodes before it arrived at the destination. How the intermediate nodes are selected is the key of the LLCHORD protocol. Every node received message and judge the type of the message and retrieve the message header. The destination and the message type are stored in the message header. If the message's destination is the node then it is consumed by the message listener resides in the local node. If the destination node is not the local node but the local node is the direct successor of the destination then the message is consumed too. The other message is forwarded to the other node according the routing table.

The next hop is selected according the finger table and the neighbors' table. In LLCHORD, there is a dual directional finger table. This is because that the structure of Chord is a ring. Then, there is a side that is short in key distance between the source and destination and another side that is long. Before we routing the message to its destination, we choose the short one. In this way, we reduce the hops of the query.

There is another trick in the routing. Once the direction of the routing is determined, the message can be routed to its destination in only one *side* of the ring. But the message can be routed in two directions around the next hop node in this side of the ring. Suppose the query is routed along the clockwise side of the initiator. As Fig.3 shows, there is query from N8 toward N38. In Chord, the first step will be N21. But in LLCHORD, the first hop is N42 (we assume that *D(N8,N42) < D(N8,N21)* in this figure).

The message is routed to a node that is close to the local node in Internet and is the successor of the desti-

nation key at the same time. The pseudo code is illustrated in the Fig.2.

The procedure *n.next_hop(id)* is used to get the node that not only close to the local node in internet but also close to the destination node in key space. This is a compromise between the average hops and the total latencies. Because the next hop is selected among the finger table, and each hop makes the query more close to the destination in the ring of node then previous hop. So we can claim that the LLCHORD protocol has the same query complexity level as Chord protocol. That is *O(log(N)-1) in LLCHORD*. Where *N* is the number of node in the network. Because each hop consumes less time then Chord protocol, so the total latency is assumed to be lower then the Chord protocol.

### 4.7. Update the routing table

The nodes in the LLCHORD network is changing dynamically, so the finger table and the neighbor table need to be updated frequently. The updating of the finger table is the same as Chord.

The *stabilize*, *notify*, *fix_fingers* and *check_predecessor* operations are used to stabilizing the network. Each node will accumulate the turn round time of each message and remote call. The sender of the message calculates the relative distance between two nodes from the time of a message consumes.

But the message sending and receiving is frequently, so the accumulates the information in 10 seconds and update the finger table.

Besides the periodic updating of the finger table and neighbors' table, the QoS is used in the message transporting. Each node will get enough information about the latency and bandwidth between the remote node and local node. Many other systems uses ICMP ping to measure the distance between hosts. The physical network topology is more complex. Nodes may reside in a local area network protected by the firewall. So the ICMP ping cannot reach those hosts. So we use the QoS to measure the distance between the nodes.

## 5. Experiments and Results

The performance is compared with the Chord like protocol. We claim that the query complexity is the same as the Chord, but with low latency. This is achieve by reduce the latency of each hop when query the distributed hash table. This sounds reasonable but needs experiments to prove.

We design and running the experiment using J-Sim[10] simulator. J-Sim is a component-based, compositional simulation environment. It has been built upon the notion of the autonomous component-programming model. The basic entity in J-Sim is components, but unlike the other component-based software packages/standards, components in J-Sim are autonomous and are realization of software ICs. The benefits of J-Sim's component model are we can use concentrate on the protocol itself and pay less attention to the simulation. J-Sim has an INET framework, which simulates the packet switch network. The application layer protocol can be build on top of the INET framework and simulated in J-Sim.

### 5.1. Experiment setup

In the experiments, we didn't simulate the node joining and leaving. The topology of the nodes we used in simulation is generated using BRITE[11], which is a general purpose topology generator. We use BRITE generate 1024 nodes in flat router model with exponential bandwidth distribution. The max bandwidth between nodes is 10240kbps, and the minimal bandwidth is 10kbps. The topology model used in BRITE is Barab´asi-Albert[6] model

The experiments are querying the network 10000 times with the randomly chosen key. The key is generated by hash the random number generated evenly between 1 and 10000. The experiments sum the query latency and hops, and calculate the average hops and average latency the two protocol used. In this way, we compare the Chord protocol and the LLCHORD protocol.

### 5.2. Results

The result shows that query in the Chord protocol needs 5.62 queries to find the key in the distributed hash table and consumes 85.2 ms per query in average. The LLCHORD protocol needs 6.94 queries to find the key in the distributed hash table and consumes 79.87 ms per query in average.

The experiments shows that the average hops of query in LLCHORD protocol are comparable to the Chord protocol. But the average latency in the LLCHORD protocol is lower then the Chord protocol. This is because that the latency of each hop during the query is reduced while the hops need to find the destination key is maintains in the same level.

### 5.3. Analysis

The experiments show that LLCHORD algorithm reduces the routing latency for distributed hash table. The reducing of the latency is improved for two reasons. First, choosing the low latency fingers as the next hop can reduce the latency of the query. Second, the usage of the dual directional finger table reduces the maximal distance between keys to ½N. The reducing in the maximal key distance can reduce the average hops used by query.

## 6. Conclusion

In this paper, we describe a refined chord protocol to control the average latency of the query. The proposed

protocol successfully reduces the latency in the query. This may be important in some kind of the applications, such as collaboration and media stream transportation. And the performance of our protocol is discussed. Briefly, the LLCHORD has two contributions. First, using the local information about the network to reduce the routing latency. Second, using the dual directional finger table to reduce maximal key distance in the Chord ring.

In the future, we will give more performance analysis and experiments and research how to get controllable transport latency in distributed hash table based system.

## Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No.60503045.

## References

[1] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. pages 149–160.

[2] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *18th IFIP/ACM Int. Conference on Distributed System Platforms*, pages 329–350, Nov,2001.

[3] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide area location and routing. pages Tech. Report, UCB/VSD–01–1141, U.C. Berkeley,

[4] J. Kleinberg. The small-world phenomenon: an algorithmic perspective. Cornell computer science technical report 99-1776. 2000.

[5] Albert, R., Jeong, H., A.-L. Barabasi, A. -L.: The diameter of the World Wide Web. Nature 401, 130 (1999).

[6] A.L. Bar´abasi and R. Albert. Emergence of Scaling in Random Networks. Science, pages 509–512, October 1999.

[7] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz and Y. Jin. An Architecture for a Global Internet Host Distance Estimation Service. IEEE INFOCOM 1999, pp. 210-217, New York

[8] T. S. Eugene, Ng, H. Zhang. Towards global network positioning. ACM SIGCOMM Internet Measurement Workshop 2001.

[9] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. IEEE Infocom'2002.

[10] J-Sim, http://www.j-sim.org/, 2006

[11] Brite: Boston university Representative Internet Topology gEnerator, http://www.cs.bu.edu/brite/, 2006

[12] T.E.Ng and H.Zhang. Predicating internet network distance with coordinates-based approaches. In Proc. IEEE INFOCOM, New York, NY, June2002.