Journal of Southeast University (English Edition) Vol. 20 No. 3 Sept. 2004 ISSN 1003—7985

TaChord: a Chord system using topology-aware routing and super peers

Chen Dongfeng Yang Shoubao Peng Xiaoyan

(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China)

Abstract: Due to minimum consideration of an actual network topology, the existing peer-to-peer (P2P) overlay networks, such as CAN, Chord, Pastry and Tapestry, will lead to high latency and low efficiency. In TaChord, a topology-aware routing approach in P2P overlays and an improved design in Chord are presented. TaChord and other algorithms are evaluated by physical hops, interdomain-adjusted latency, and aggregate bandwidth used per message. Experimental results demonstrate that TaChord has the drastic improvement in routing performance where average physical hop is half that of chord, and the impact of cache management strategies in the TaChord overlay cannot be neglected.

Key words: peer-to-peer network; topology-aware routing; super peer; Chord

Peer-to-peer (P2P) Internet applications have recently been popularized through file sharing applications such as Gnutella[1] and Freenet[2]. These systems have many interesting technical features such as decentralized control, self-organization, and adaptation. However, these systems, such as Gnutella, may have scaling problems. Meanwhile, several research groups have developed a new generation of scalable P2P systems that support distributed hash table (DHT) functionality; among them are Tapestry[3], Pastry[4], Chord[5], and CAN[6]. In these systems, files are associated with a key, and each node in the system is responsible for storing a certain range of keys.

Chord, in its original design, does not consider network proximity at all. As a result, messages may travel arbitrarily long distances over the Internet in each routing hop. And it assumes that nodes in the system are uniform in resources such as network bandwidth and storage. This results in its routing without taking actual network topology and differences between node resources into consideration. In topology-aware Chord, a secondary overlay layered on Chord maintains characteristics of Chord. Topology-aware routing algorithms will not ignore the latency of individual hop, and are prone to result in low latency paths. In fact, in Ref.[7], it shows that topology-aware Tapestry has greatly improved point-to-point routing distance and reduced network bandwidth usage.

In Ref. [8], a super-peer network has the potential to combine the efficiency of centralized search with the autonomy, load balancing and robustness against attacks provided by distributed search. In this case, we also adopt super peers in topology-aware Chord. These super peers maintain node ID lists of the autonomous system (AS), as well as routing cache.

In this paper, we present our P2P system based on the Chord system, and demonstrate its potential performance benefits by simulation.

1 Chord Routing and Key Location

Chord uses a one-dimensional circular key space. Messages are forwarded only in the clockwise direction in the circular ID space. The node is called the key' s successor, whose identifier most closely follows the key. When a lookup message begins, the node checks its finger list and selects a node closest to the key. In an N-node Chord, each node maintains information only about $O(\log N)$ other nodes. Routing efficiency depends on the finger list of $O(\log n)$ nodes spaced exponentially around the key space. All lookups need $O(\log n)$ messages to other nodes. When a machine joins or leaves, the routing information updates require $O(\log^2 n)$ messages.

1.1 Hash function

As described in Ref. [5], a hash function such as SHA-1 maps nodes and keys onto an m-bit circular identifier space. It defines a successor of a key k as the node, which equals or follows k in the identifier space. This node is called the successor node of key k, and it stores the key k. That is, each node with a hashed identifier hid is responsible for all hashed keys k such that

Received 2003-09-05.

Foundation item: The National Natural Science Foundation of China (No. 60273041).

Biographies: Chen Dongfeng(1979—), male, graduate; Yang Shoubao (corresponding author), male, professor, syang@ustc.edu.cn.

$k \in$ (predecessor (hid), hid]

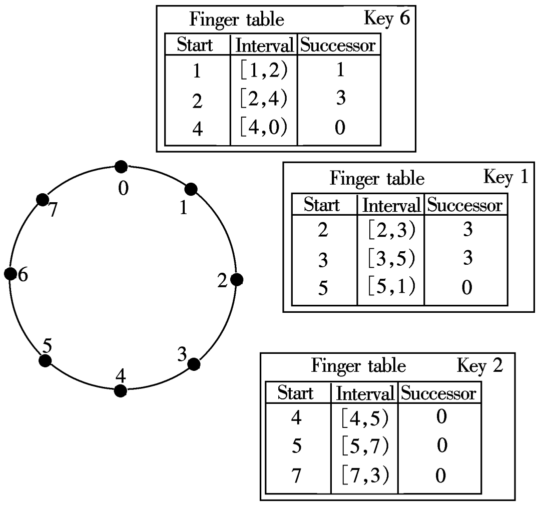
If the node successor (k) is found, it shows that the routing is successful.

If m is not taken properly, there is a chance of a collision where some nodes hash to the same identifier. However, in SHA-1 hash function, m is large enough to make it impossible. A unique suffix, such as the node's IP address and port, also can be appended to the identifier for each node to ensure unique identifiers.

1.2 Routing and key location

In the m-bit circular identifier space, each node maintains a finger table with m entries. The i-th entry in the table at node n contains the identity of the first node, s, that succeeds n by at least 2^{i-1} on the identifier circle, i.e., s = successor $((n+2^{i-1}) \bmod 2^m)$, where $1 \leqslant i \leqslant m$. The node n.successor is the immediate successor of n on the identifier circle, and n.predecessor is the immediate predecessor of n on the identifier circle.

Fig.1, as described in Ref.[5], shows an example of interval routing. There are three nodes whose identifiers are 0, 1, and 3. The finger table of node $n = 1$ stores the successors of identifiers $(1+2^0) \bmod 2^3 = 2$, $(1+2^1) \bmod 2^3 = 3$, and $(1+2^0) \bmod 2^3 = 5$, respectively. The successor of identifier 2 is node 3, as this is the first node that follows 2, the successor of identifier 3 is node 3, and the successor of 5 is node 0. There are three keys whose identifiers are 6, 1, and 2, which belong to node 0, 1, and 3, respectively.



Finger table Key 6

Start	Interval	Successor
1	[1,2)	1
2	[2,4)	3
4	[4,0)	0

Finger table Key 1

Start	Interval	Successor
2	[2,3)	3
3	[3,5)	3
5	[5,1)	0

Finger table Key 2

Start	Interval	Successor
4	[4,5)	0
5	[5,7)	0
7	[7,3)	0

Fig.1 Key and finger table associated to each node

Suppose node 1 wants to find the successor of identifier 5. Since 5 belongs to the circular interval [5, 1), it belongs to the second finger interval. Therefore, node 1 checks the successor, which is 0. Because node 0 is the successor of key 5, node 0 will return the result to node 1.

In Chord, nodes can join and leave at any time. It must preserve the ability to locate every key in the network, when nodes join and leave. More details are discussed in Ref. [5].

2 Topology-Aware Chord Routing

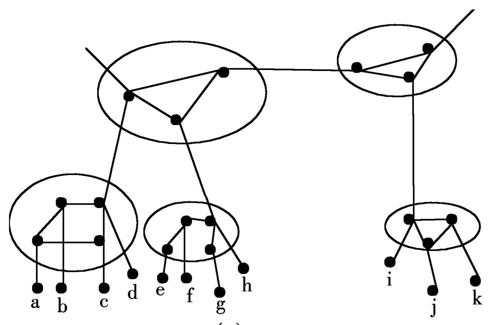
Here we present the overall design for the topology-aware Chord with super peers, and design algorithms for the topology-aware routing. In peer-to-peer systems, transfer of data between peers is no longer simple and uni-cast. It is associated with the real network, where there are a number of traditional servers, middleware forwarding and routing layers, and so on. And communication performance is very important in peer-to-peer systems. So we utilize the network topology in Chord. In the area hierarchy[9] of real networks, a set of routers are grouped into a level 0 area, a set of level 0 areas are grouped into a level 1 area, and so on. Level k routers keep routing information on all other level k areas within its own level $k+1$, and all level $k-1$ areas within itself. It means that routers become bottlenecks. However, we modify the technology of routing indices[10], which allow peers to forward queries to neighbors that are more likely to have answers.

In the topology-aware overlay, the routing idea is to choose routing table entries which refer to the topologically nearest among all nodes. At the same time, we use super peers to improve routing performance on the overlay. These super peers have high bandwidth and the ability to fast access to the wide-area network. A TaChord system, a topology-aware Chord system with super peers, provides a shortcut routing algorithm by looking up super peers' caches.

To describe how a TaChord functions, we will first give an overview, and then describe the TaChord routing in detail.

2.1 TaChord construction

A node in a TaChord system maintains the following data structures: ① Finger table, which is a successor list of nodes that immediately follow it in the key space, as described in Chord; ② Routing table with a few entries, which is constructed according to the physical network topology; ③ Local node ID list, which consists of all nodes in the same AS; ④ Key caches (only if it is a super peer), which is used in the shortcut routing. Here is an example of TaChord routing in Figs.2 and 3.



(a)

Cache (optional)	
ID	IP
hash_1	IP_1
hash_2	IP_2
hash_3	IP_3

Finger table	
D	IP
suc_1	IP_1
suc_2	IP_2
suc_3	IP_3

Pointer table	
ID	IP
file_1	IP_1
file_2	IP_2
file_3	IP_3

Routing table	
ID	IP
neighbor_1	IP_1
neighbor_2	IP_2
neighbor_3	IP_3

Local node table	
ID	IP
ID_1	IP_1
ID_2	IP_2
ID_3	IP_3

(b)

Fig.2 Structure of route table. (a) Network topology; (b) Route table

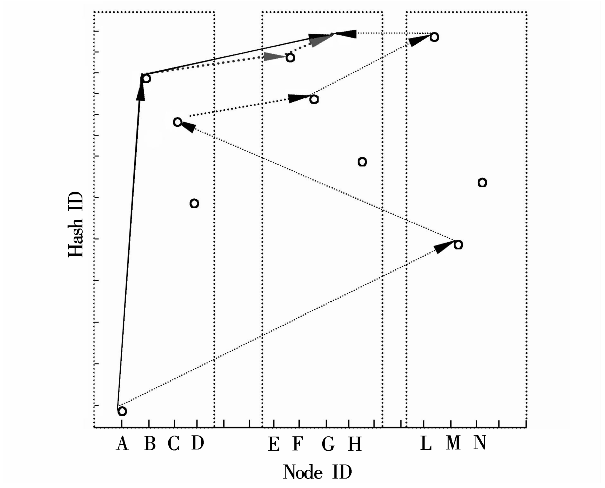


Fig.3 Routing path of TaChord and Chord

In this example, node b is a super peer. Suppose node a wants to find the successor of some key, node g. In Chord routing, node a checks its finger table and selects the next node, m. The node a forwards messages to node m, and so on. As a result, the routing path of Chord is as follows: a→m→c→f→l→g.

In the topology-aware Chord routing algorithm, node a makes out node g from the finger table as it does in Chord, then simultaneously picks up node b in its local node list. Since it is closer to the destination, node b is selected as the next node. If node b is a super peer without routing cache, it will do next as its former node a does. Node b forwards messages to the next node e. Finally, the successor of the key node g is found. We receive the routing path of topology-aware Chord as follows: a→b→e→g. If node b is a super peer with routing cache and the key is just in the cache entries, node b will directly send messages to the destination. As discussed above, the TaChord routing path is a→b→g.

We calculate the routing latency. Each inter-domain hop counts as 3 hop units of latency. The routing latency of TaChord, topology-aware Chord without cache, and Chord are 4, 5, 15, respectively. TaChord shows the drastic improvement in latency. We also present other performance improvement in section 3.

2.2 Routing algorithm

There are different communication properties, such as for content-based or policy-based routing[11]. We choose policy-based routing, which can be used to meet quality of services (QoS) requirements and enforcing message delivery order policies across peers.

Here we describe routing algorithms required for a topology-aware Chord. When it receives a routing message, a node n firstly checks its finger table, routing table and local node ID list. If the node where the key is stored is not found, it will be sent to the node's super peers. If the super peers' caches show that no other node has looked up the key before, the node n needs to determine the next node closest to the message destination from these data sets above. The following is the pseudo code of TaChord algorithm.

```
route(n, key)
  if(n.precedor < key && key < = n.hashid)
    return n;
  else
    if(lookup_Superpeers(n.key))
        return n;
    n_1 = lookup_in_fingertable(n, key);
    n_2 = lookup_in_routingtable(n, key);
    n_3 = lookup_in_localnodelist(n, key);
    next = min_distance(n_1, n_2, n_3, key);
    return route(next, key);
```

n is the node which wants to find the successor node of an identifier key. The algorithm firstly checks whether current node is the node responsible for the key. To reduce message traffic in an inter-domain network, super peers keep a cache to store remote nodes with which those nodes in the same domain have communicated. If the destination is found in the cache, it is returned to the sender. If not, n picks the node closest to the destination node. The process continues in a similar fashion to the next node next till the query is satisfied.

To initiate the cache of super peers, all nodes in the same domain will send the query result to its super peers as soon as the query is finished, and super peers will add the result into their cache. Many strategies are used in managing these cache entries. As we will present following, there are three popular policies: FIFO (first-in-first-out), LFU (least frequently used), and LRU (least recently used).

3 Evaluation and Results

In this section, we present some analysis and simulation results showing the performance improvement possible with the use of TaChord. We intend to obtain results on the following questions:

- What is the performance improvement with the use of TaChord compared with existing algorithms?
- What is the influence of the sum of super peers k in a domain on the efficiency of the TaChord construction?
- What is the difference of policies used in managing super-peer cache?

Before presenting our simulation results, we first describe a network simulator and network topology we use.

3.1 Network topology

We use BRITE[12], a topology generator, to implement our simulation. Recent empirical studies have shown that Internet topologies exhibit power laws of the form $y = x^{\alpha}$ for the following relationships: ① Outdegree of node versus rank; ② Number of nodes versus outdegree; ③ Number of node pairs within a neighborhood versus neighborhood size (in hops); ④ Eigenvalues of the adjacency matrix versus rank. Some generated topologies may not obey power laws ① and ②. In Ref. [9], BRITE is a more realistic topology generator, which generates truly representative Internet topologies.

Here we use a two-level hierarchical topology generated by BRITE. The hierarchical topology was generated by the bottom-up approach. The router-level topology used Waxman models, and router nodes were assigned to AS in the way of heavy-tailed form. We constructed Chord networks of size 4 096, and marked 100 as the size of AS.

3.2 Hop and latency

We measured the performance of hop and latency using three algorithms: original Chord, topology-aware Chord without super peer cache, and TaChord. The TaChord parameters are set to super-peer set size $k = 3$ and FIFO cache removal strategy. We measured the hops and latency F (probability density function) of different algorithms.

Fig. 4 shows a distribution of the hop per message. Both topology-aware Chord without super peer cache and TaChord algorithms improve upon original Chord point-to-point routing. This is because topology-aware Chord incorporates topology information to routing. As also shown in Fig. 4, TaChord offers maximal improvement. In TaChord, if the destination is found in its super peer cache, the node sends the message directly to the destination.

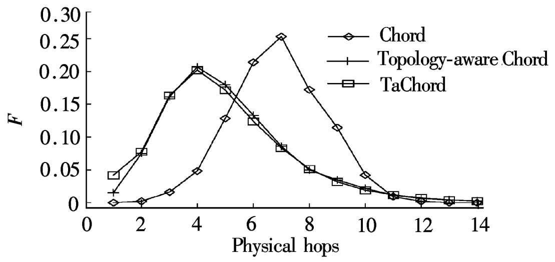


Fig.4 Distribution of physical hops

When it comes to the fact that inter-domain routes have higher latency, Fig.5 shows a PDF where each inter-domain hop counts as 3 hop units of latency. We use max as the metric, which is the value corresponding to maximum probability. It is easily made out that the value of max is 12 for TaChord, while the value is 21 for Chord. It shows that topology-aware Chord still presents a great improvement in the PDF of the latency.

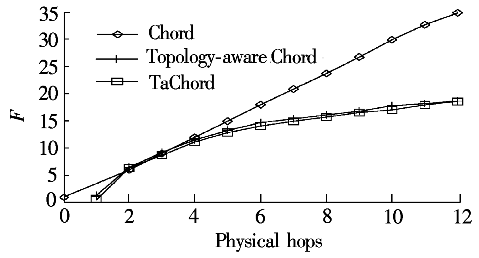


Fig.5 Distribution of latency

We finally measure the aggregate bandwidth taken per message delivery, using units of (average size of MSG × hops). As shown in Fig.6, when we do not take intra-domain bandwidth used into account, TaChord dramatically reduces inter-domain bandwidth

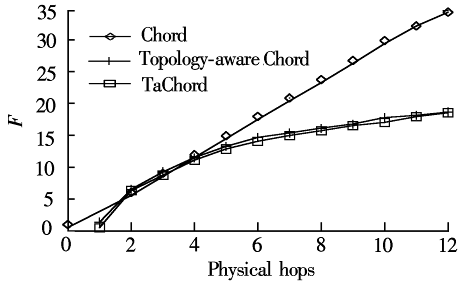


Fig.6 Aggregate bandwidth used per message

usage per message delivery. That is because TaChord, which uses cache and topology information, forwards messages directly to the destination, and reduces message forwarding on the inter-domain.

3.3 Super peer redundancy

A super peer may become a single point of failure for its domain. That is, when the super peer fails or leaves unexpectedly, the local node list may be destroyed and cache in the super peer will be not available any longer. In Ref.[8], super-peer redundancy is introduced into the design of the super-peer. All super peers in the same domain share with equal responsibilities. When a node receives a query message, it sends the query to each super peer in the same domain in a round-robin fashion. We only considered the cases where $k=1, 3, 5$, respectively, and we constructed Chord networks of size 1 024, and marked 50 as the size of AS.

As shown in Fig. 7, super-peer redundancy is good. In Fig. 7(a), the curve of the case where $k=5$ drops rapidly after 3 physical hops. We can see that both physical hops and latency improve greatly as k increases. This is expected, since a k-redundant super-peer has much greater availability and reliability, and has cache entries k times more than a single super-peer.

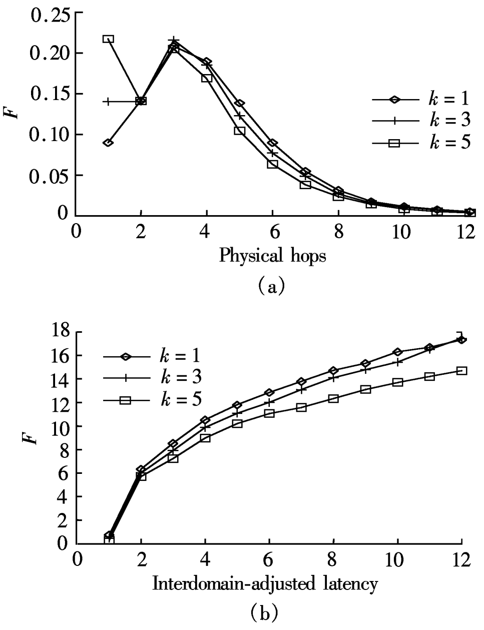


Fig.7 Performance of k-redundancy. (a) Distribution of physical hops; (b) Distribution of latency

However, super-peer redundancy costs a lot. When a node joins or leaves, each super peer must receive or update metadata. Each super peer maintains cache, and the aggregate cost of cache memory is k times greater than a single super-peer. Besides, when a node sends queries to each super peer in a round-robin fashion, it must increase intra-domain network traffic.

3.4 Cache removal policy

Here we compare the performance of some simple cache removal policies, such as FIFO, LRU, and LFU. To simplify the analysis, we consider the case where the keys are limited in one part of the one-dimensional circular key space. Fig.8 shows that LRU and LFU show a little improvement in hops and latency found in the simplistic simulation. We can see that different cache management strategies affect the performance improvement of TaChord system.

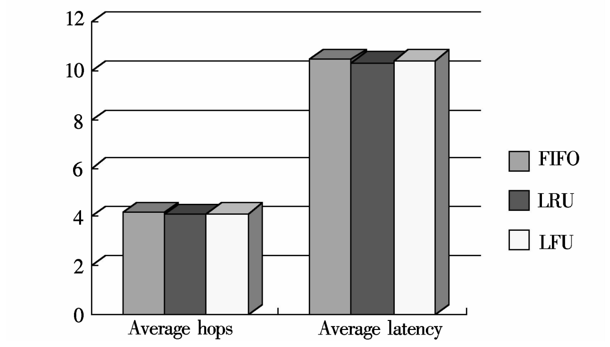


Fig.8 Average hops and latency of a 4 096 node network with 100 domains

4 Conclusion

This paper presents a study of topology-aware routing in Chord. We present improved routing in Chord, which significantly reduce the overhead of routing and bandwidth consumption in an inter-domain overlay. Simulations confirm that TaChord yields good performance at low overhead. Super-peer redundancy is good, and also increases the overhead of topology-aware overlay construction and maintenance. Cache management strategies can also affect the performance improvement in TaChord. The investigation of the overlay maintenance, super-peer size, and cache removal policies is ongoing.

References

[1] The Gnutella protocol specification v0.4[EB/OL].http://www.clip2.com. 2003-06.
[2] Clarke Ian, Sandberg Oskar, Wiley Brandon, et al. Freenet: a distributed anonymous information storage and retrieval system[EB/OL]. http:// freenet.sourceforge.net. 2003-06.
[3] Zhao Ben Y, Kubiatowicz John. Tapestry: an infrastructure for fault-tolerant wide-area location and routing[R]. Computer Science Division University of California, UCB/CSD-01-1141, 2001.

[4] Rowston Antony, Druschel Peter. Pastry: scalable, decentralized object location and routing for large scale peer-to-peer systems [R]. Cambridge: Microsoft Research Ltd, 2001.

[5] Stoica Ion, Morris Robert, Karger David, et al. Chord: a scalable peer-to-peer lookup service for internet applications[A]. In: *ACM Sigcomm*[C]. 2001. 2 − 10.

[6] Ratnasamy S, Francis P, Handley M, et al. A scalable content-addressable network [A]. In: *ACM Sigcomm* [C]. 2001. 1 − 3.

[7] Zhao Ben Y, Duan Yitao, Huang Ling, et al. Brocade: landmark routing on overlay networks[A]. In: *Electronic Proceedings for the* 1*st International Workshop on Peer-to-Peer Systems*(*IPTP*' 02)[C]. Cambridge, 2002. 2 − 5.

[8] Yang Beverly, Garcia-Molina Hector. Designing a super-peer network [A]. In: 19*th International Conference on Data Engineering*, *IEEE Computer Society*[C]. Bangalore, 2003. 3 − 6.

[9] Tsuchiya P F. The landmark hierarchy: a new hierarchy for routing in very large networks [J]. *Computer Communication Review*, **1988**, **18**(4): 35 − 42.

[10] Crespo Arturo, Molina Hector Garcia. Routing indices for peer-to-peer systems [A]. In: *Proceedings of the* 22*nd International Conference on Distributed Computing Systems* (*ICDCS*' 02)[C]. Vienna, Austria, 2002. 23 − 34.

[11] Lee Craig A, Coe Eric, Michel B Scott, et al. Using topology-aware communication services in grid environments [A]. In: *Proceedings of the* 3*rd IEEE/ACM International Symposium on Cluster Computing and the Grid* (*CCGRID*03)[C]. Tokyo, 2003. 1 − 6.

[12] BRITE, a network topology generator[EB/OL].http://www.cs.bu.edu/brite/. 2003-06-02.

TaChord: 利用拓扑相关路由算法和超级节点的 Chord 系统

陈东锋　杨寿保　彭小燕

(中国科学技术大学计算机科学技术系, 合肥 230027)

摘要: 由于未考虑实际网络拓扑结构, 当前结构化对等网络系统存在高延迟低效率的毛病. 本文提出一种与实际网络拓扑结构相关的路由方法, 并开发出一种基于 Chord 的 TaChord 系统. 在仿真系统中, 用物理跳数、域间延迟和每个消息占用的带宽来衡量 TaChord 和其他系统的路由性能. 仿真结果表明, TaChord 系统的路由性能有了极大的提高, 如平均物理跳数是 Chord 的一半, 并且不同路由缓存管理策略对 TaChord 的路由性能能产生重要的影响.

关键词: 对等网络; 拓扑相关路由; 超级节点; Chord

中图分类号: G237.5; H315.9