Twins: 2-hop Structured Overlay with High Scalability*

Jinfeng Hu, Haitao Dong, Weimin Zheng, Dongsheng Wang, and Ming Li

Computer Science and Technology Department, Tsinghua University, Beijing, China
{hujinfeng00, dht02, lim01}@mails.tsinghua.edu.cn,
{zwm-dcs, wds}@tsinghua.edu.cn

Abstract. How to build an efficient P2P overlay network on a large-scale system is still in the air. Pastry-like p2p overlays have low maintenance costs because of their log(N)-sized routing tables. However their lookup efficiency is quite low. One-hop overlays, although having high routing efficiency, can not scale to large systems because of its high maintenance cost. In this paper, we present a novel structured overlay network, Twins. Routing in Twins can be accomplished in 2 hops in very high probability. With a report-based multicast maintenance algorithm, the overlay network consumes very low maintenance cost in presence of large-scale and highly dynamic network environments. The experimental results indicate that, when the system running over a network of 5,000,000 peers, each peer consumes only 6 messages per second for maintenance, and the routing latency is only 2 hops in a very high probability of 0.99.

1 Introduction

With the introduction of Napster in 1999, the peer-to-peer system became "the fastest growing Internet application ever". Currently, the number of concurrent users of Kazaa has rapidly increased to about 3,000,000[3], and seemingly this trend has no sign to cease in the near future. The increasing system scale has become one of the main challenges for the design of structured P2P overlay network.

Structured overlays can be classified into two categories: Pastry-like overlays and one-hop overlays. The size of Pastry-like overlays' routing table is O(logN) (such as Pastry[6], Tapestry[9], and Chord[8]). With a relatively small-scale routing table, Pastry-like protocols need little amount of maintenance cost to deal with the system membership changes, which means that these protocols can achieve high scalability. But the price of this approach is that on average lookup operation requires O(logN) steps to converge. For example, a typical 16-based(b=4) Pastry network needs about $\log_{16}3{,}000{,}000 \approx 5.38$ hops for message forwarding, which can not be tolerated by many particular applications. The overlay route latency is becoming another challenge for P2P protocol.

The other approach of structured overlays is one-hop overlay [1]. Every overlay node keeps complete membership information of all the other nodes in the system. Its

* This work is granted by The National High Technology Research and Development Program of China (G2001AA111010), Chinese National Basic Research Priority Program (G1999032702), and a joint research grant from National Science Foundation of China (project No.60131160743) and Hong Kong Research Grant Council.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 174–183, 2004.

lookup operation accomplishes in one hop. But the one-hop overlay's routing table is too large and consumes too much bandwidth for updating. Nowadays, the average lifetime of peers is about one hour [7], that is to say, given a P2P overlay network of 3,000,000 nodes, every node must receive information of 3,000,000*2 =6,000,000 member-changing events per hour. Typically, the data structure of a membership-changing event is about 200 bits long, include corresponding node's nodeId, IP address and port. Thus the bandwidth cost is no less than 33.3 kbps, which is too heavy a burden to most modem-linked peers. So the scalability of existing one-hop overlay is very poor.

It is the two challenges mentioned above that motivates us to design a new scalable structured overlay, Twins, which simultaneously obtains high routing efficiency (2-hop routing in 99% probability) and low bandwidth overhead (6 messages per second in a 5,000,000-node network).

Twins' routing table consists of two parts, one containing all nodes sharing a h-bits prefix and the other containing all nodes having a b-bits common suffix(*h* and *b* are systematic parameters), with a simple routing algorithm it can route messages to their destinations in just 2 hops, in a very high probability of 0.99. Twins adopts a report-based routing table maintenance algorithm. When a membership change event occurs, the overlay will multicast this event in a report-based mechanism, which helps the overlay consumes low bandwidth to deal with node's join and crash. Our experimental result shows that when running over a P2P network of 5,000,000 peers, each Twins node consumes only 6 messages per second for maintenance. This cost, as well as routing table size, varies as a $O\left(\sqrt{N}\right)$ function to the overlay scale *N*, so Twins can also run well in an even larger environment. Moreover, Twins introduces probabilistic routing into structured overlay. How many hops a message passes before reaching its destination is not strictly determined. This makes room for scalability design: we can raise the expectation of hops to keep a low overhead by adjusting system arguments.

The rest of this paper is organized as follows. Section 2 presents the design of Twins protocol. In section 3 we give a formalized analysis of the routing performance and the maintenance cost. Experimental results are presented in section 4. Final conclusion is given in section 5.

2 The Twins Protocol

This section describes the Twins protocol. The Twins protocol specifies how to locate keys, how to construct the routing table, and when new node join or existing nodes fail how to maintain the structure of the system.

Like Pastry and Chord, Twins assigns every node and key an identifier using SHA-1, typically 128-bit long. They are ordered in an identifier ring modulo 2^{128}, and we assume that all these identifiers are uniformly distributed in the identifier space. For a node, say node M, we call the first p bits of M's node ID the node's prefix, and the last s bits the node's suffix, we assume that p(s)<128, and the scale of the overlay network is N($N << 2^{128}$).

2.1 Routing Table

Each Twins node has a routing table consisting of two parts. The first one contains all the nodes having the same prefix as M, called ***prefix set***. While the second one contains all the nodes having the same suffix as M, called ***suffix set***. In all, a Twins node's routing table is the union of its prefix set and suffix set.

Obviously if two nodes have the same prefix, their prefix set must also be the same. In this way, the set of all the peers can be partitioned into 2^h groups according to their different prefix. We call these groups *prefix groups*. Nodes within the same prefix group are fully interconnected with each other. Since node IDs are distributed evenly in the ID space, averagely each prefix-group includes about $N/2^h$ nodes. The *suffix group* is defined similarly, i.e., all the nodes are partitioned into 2^b different suffix groups which do not intersect with each other, and the expected value of the size of every suffix-group is $N/2^b$.

Given a 128 bits id *N*, we define prefix-group$_h$(N)={M | M is a 128 bit identifier, the first h bits of M is the same as N's first h bits}, suffix-group$_b$(N)={M | M is a 128 bit identifier, the last b bits of M is the same as N's last b bits}, h and b are system's parameters.

Figure 1 shows the routing table of a hypothetical node with a 12-bit node ID 011100101110.

2.2 Routing

Every message has a destination ID that is also 128-bit long. The first h bits of the message's destination ID is also called the message's h-bit prefix, and the unique prefix-group corresponding to this prefix is called its prefix-group. Similarly, we define the message's suffix-group as the group consisting of all the nodes whose ID's b-bit suffix is the same as the message's b-bit suffix.

Slightly different with Pastry, the destination node of a message is defined as following:

The destination node of a message M is the node in the M's prefix-group whose ID is the numerically closest to the destination ID of M.

Notice that all the nodes in same prefix-group are fully interconnected with each other, and the basic task of message routing is forwarding messages to the corresponding prefix-group. After that the message will directly reach its destination node in one hop.

When routing a message with destination ID *M*, node *N* first checks whether M and N have the same *prefix* (that is to say if M ∈ prefix-group$_h$(N)). If so, N can directly forwards it to the destination node, which must be in N's prefix set; otherwise N inspects its suffix set, tries to seek out a node E who has M's prefix(the node E, such that M ∈ prefix-group$_h$(E) and E ∈ suffix-group$_b$(N)) and forwards the message to E. For example, when the node shown in Fig.1, named N, will route a message M. If M's destination ID is 011100000000, according to our routing algorithm, M ∈ prefix-grouph(N), then N forward M to 011100100101, which is numerically closest to M in N's prefix set. And if M' destination ID is 001000000000, node N will choose node 001011001110 and forward M to it, because this node has the same 4-bit prefix as M.

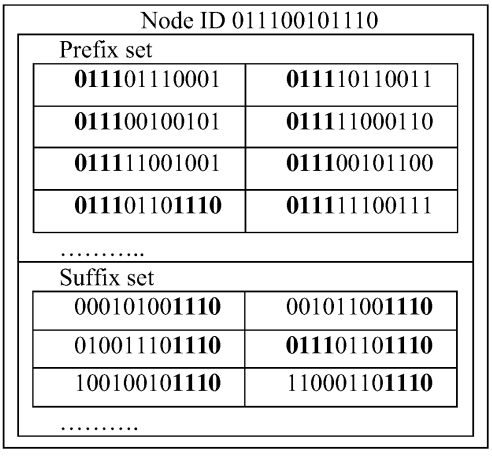

Fig. 1. State of Twins's Routing Table. Node's ID is 12 bit long, and we assume h=4, b=4. Node M's ID is 011100101110. M's Routing Table consists of Prefix set, which contains all the nodes whose ID has the same 4-bit prefix as M, and Suffix set, which contains all the nodes whose ID has the same 4-bit suffix as M. For all the nodes, their prefix and the suffix are shown in boldface. Because of space, we can not list M's Routing Table completely. And it is easily to discover that the prefix set may intersect with suffix set, in this case they have the common node ID **0111**01101**1110**.

If there is no node in N's suffix set which has the same prefix as M, N forwards the message to a node randomly chosen from its prefix set whose suffix is different from M.

2.3 Maintenance

Like one-hop overlay, Twins introduces a report-based multicast mechanism to maintain routing table. Note that prefix-groups are independent to one another, and so do suffix-groups. It allows us to simply consider how to maintain nodes' prefix set/suffix set within a prefix/suffix-group.

```
route(M)
  if  M ∈ prefix-group_h(N)
      then  Forward M to its destination node;
            Return;
  else  for every node E in N's suffix set do
        if  M ∈ prefix-group_h(E)
            then    Forward M to E;
                    Return;
  Random select a node E from N's prefix set;
  Forward M to E;
```

Fig. 2. Pseudo code of Twins' routing algorithm

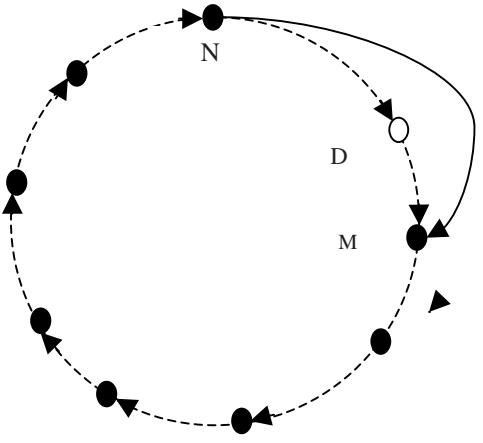

Fig. 3. This is a hypothetical prefix-group. All the nodes in this group are organized as a ring, in which every node must send periodical heartbeat message. We can see if there is a node, say M, which fail to receive heartbeat message from its frontal node, say D, for several heartbeat period, M will take node D for death, initiate the event of D's death and then multicast this event message in the prefix-group. The multicast algorithm is shown in fig.4. When node N receives this event message, it resets its next node as M and begins to send heartbeat message to M.

When a new node X joins Twins system, firstly it should contact an existing Twins node B, which is named as X's bootstrap node. On receiving X's join request, B initiates a lookup of X's ID to find two nodes, say P and S, in X's prefix group and suffix group respectively, and then X requests prefix set from P, and suffix set from S. After the receiving of X's prefix set and suffix set, X can establish its routing table to complete its joining process. And node P and node S multicast the event of X's joining to all the nodes in X's prefix-group and suffix-group respectively.

All the nodes in a group are ordered in an identifier ring. Every node sends periodical heartbeats to the first node next to it in the identifier ring. If a node D has not send heartbeats for several periods, then D is considered as dead, and the node next to D in the ring will multicast the event of D's death to all the nodes in this group. The message used to inform other peers that there occurs a member-changing event is called an event message. Fig.3 shows the maintenance mechanism of a hypothetical prefix/suffix-group.

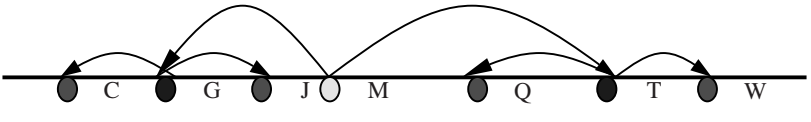

Fig. 4. An instance of tree-based multicast

Notice that all the nodes within a group are fully interconnected, there are various algorithms to carry out the multicast process. Here we adopt a simple tree-based multicast, illustrated in Fig. 4. When a node M initiates a multicast process , it sends the event to a node before it (say G) and another behind it (say T). Then alike, G and T each sends the event to two nodes, one ahead and the other behind. This procedure

continues. At each step, every node should ensure that once it sends the event to another node, there is no other node between them who has already received this event.

If more efficient multicast is desired, the multicast tree can be modified to be 2^b-based. And if more reliable multicast is desired, response-redirect mechanism can be deployed, which will double the maintenance cost.

In this manner, except for the changing member's node ID, IP address and port, an event message (e.g. message from T to Q in Fig.4) should additionally include node ID of the first node before the receiver who has already received the message (that is M), as well as node ID of the first such node behind it (that is T). Plus UDP head (64 bits) and IP head (160 bits), an event message will not exceed 500 bits.

Assuming that the average lifetime of nodes is 1 hour, which is the statistic result from [7], all the items in the routing table have to be refreshed in a period of 1 hour. It means that for a Twins system, which consists of 5,000,000 nodes, and h and b are both set as 10, on average every prefix/suffix-group contains $5{,}000{,}000/2^{10}$=4883 nodes. Then on average, every node receives (4883+4883)*2=19532 event messages per hour, 5.43 messages per second in other words. Plus heartbeats and their responses, the total message count per second will not exceed 6, i.e., the bandwidth cost is lower than 6message/second*500bit=3kbps.

3 Performance Evaluation

In this section, we give a formalized evaluation of Twins protocol, and describe how to determine the length of prefix and suffix in a given system environment, and then estimate routing table size and maintenance cost.

Assuming that in a Twins overlay network of N nodes, every node has a prefix with h bits and a suffix with b bits. Then there are 2^h prefix-groups and 2^b suffix-groups totally. On average, each prefix-group contains $H=N/2^h$ nodes and each suffix-group contains $B=N/2^b$ nodes.

3.1 Routing Performance

For a certain node M, it has a suffix set with 2^b routing table entries. M wishes that these entries would distribute over all the prefix-groups. But unfortunately it is not the reality: there must be some prefix-groups which there are no routing table entries in M's suffix set belonging to. We note the number of such prefix-group for the given node M as S. The expected value of S can be defined as follows:

$$E(S)=\sum_{k=0}^{H-1} k\times\frac{\binom{2^h}{k}\binom{B-1}{2^h-k-1}}{\binom{2^h+B-1}{2^h-1}}$$

For a given message destination ID D, We call the probability of there are no less than one routing table entry in M's suffix set which has the same prefix as D as *hit*

ratio. Obviously, the hit ratio can be defined as $hr = 1 - \frac{E(S)}{H}$. Consequently, hr is related to the ratio of size of suffix set (B) to the number of prefix-groups(2^h), we define $R = S / 2^h$, and then we can see that when 2^h is larger than 50, choosing R as 4.7 will ensure a hit ratio more than 0.99. So in common cases we demand that $R > R_0 = 4.7$. That is to say, $N/(2^b \cdot 2^h) > R_0$, namely

$$b + h \le \log_2(N/R) \tag{1}$$

Next we estimate maintenance cost. Heartbeat cost is a fixed small value, so for simplicity we can ignore it. The substantial cost is for maintaining prefix set and suffix set, with size of $H+B$. Assuming that nodes' average lifetime is L seconds, each node triggers two events in a period of L seconds on average. So every node receives $2 \cdot (H+B)$ events during every L seconds. If redundancy of the multicast algorithm we adopt is f, then the number of messages a node receives per second is

$$m = (H+B) \times 2 \times f / L = \left(\frac{N}{2^h} + \frac{N}{2^b} \right) \times 2f \Big/ L$$

When $h = b$, m reaches its minimum value:

$$m_0 = \frac{2N}{2^{\frac{1}{2}(h+b)}} \times 2f = \frac{4N \cdot f}{2^b \cdot L} \tag{2}$$

$$b = h = \left\lfloor \frac{\log_2(N/R)}{2} \right\rfloor$$

Considering both (1) and (2), we should set to ensure a 2-hop routing hit ratio larger than 0.99 using a minimal maintenance cost. Then we can get the following results:

a) Routing table size R_{size} satisfies

$$2\sqrt{N \cdot R} \le Rsize < 4\sqrt{N \cdot R} \tag{3}$$

b) Maintenance cost m_0 satisfies

$$\frac{\sqrt{N \cdot R} \times 4f}{L} \le m_0 < \frac{\sqrt{N \cdot R} \times 8f}{L} \tag{4}$$

Figure 5, 6 and 7 show the variation of b, R_{size} and m_0 as functions of N using f=1 and L=3600.

3.2 Scalability

Inequation (3) and (4) show that a Twins node with a routing table with $O(\sqrt{N})$ entries consumes $O(\sqrt{N})$ bandwidth to maintain it. This means a good scalability property of Twins overlay network. In addition, when the maintenance cost is not acceptable by peers, Twins also can trade hops for bandwidth consumption like other

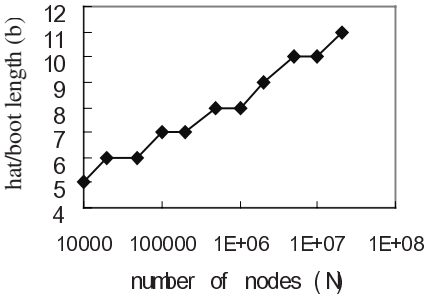

Fig. 5. Hat length vs. *N*

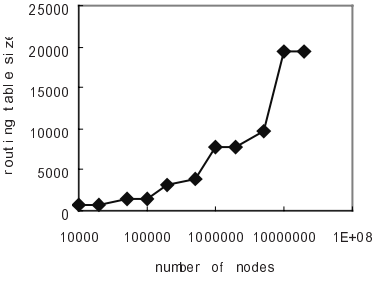

Fig. 6. Routing table size vs. *N*

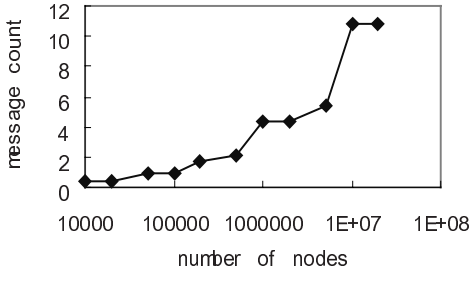

Fig. 7. Message count vs. *N*

overlay protocols. To illustrate it, we put Twins into a stricter environment where *N*=10,000,000, *f*=2 and *L*=2,400. Using inequation (4) we know that if keeping 2-hop routing, a node should receive at least 22 messages per second. Twins can reduce this cost by decreasing *R*, which will raise *b* and *h*, decrease *B*, and then reduce *hr*. Expected value of hop counts can be calculated as:

$$E(hops) = \sum_{i=2}^{\infty} i \cdot (1-hr)^{i-2} hr = 1 + \frac{1}{hr}$$

Even when *hr* drops to 0.25, average hop counts only rise to 5.

4 Simulation and Experimental Result

In this section, we present experimental results obtained with a simulator for Twins protocol. The simulator was implemented on ONSP, an overlay networks simulation platform, which can parallel simulating the function of most off-the-shelf peer-to-peer protocols. By implementing the event logic according to the protocol's definition, the user can easily simulate various protocols. The detailed description of ONSP exceeds the scope of this paper, we will present it in another paper.

Our experiments were performed on a 32 processors cluster (Pentium IV CPU and 2G memory), running Linux Redhat 7.0.

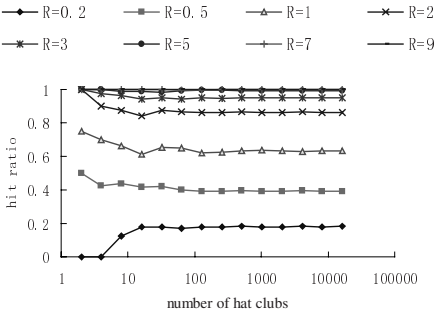

Fig. 8. Hit ratio vs. number of prefix-groups. *R* is the ratio of suffix set size (*B*) to number of prefix-groups (2^h). It shows that when 2^h exceeds 50, hit ratio is almost a constant number.

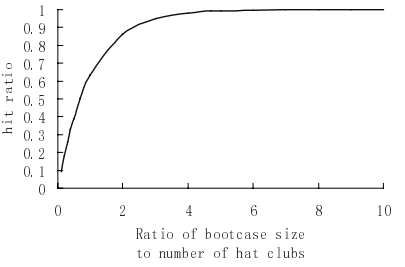

Fig. 9. Hit ratio vs. *R*, ratio of suffix set size to number of prefix-groups. Here number of prefix-groups is fixed at 1024.

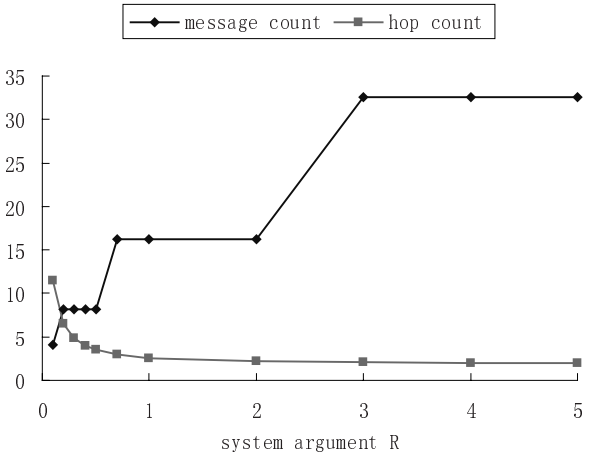

Fig. 10. Message cost and hop count in relation to the argument *R*, where *N*=10,000,000, *f*=2 and *L*=2,400

To simulate the real network topology and latency, we implemented a network topology generator based on GT-ITM[10]. The generator produces a transit-stub network topology model. We also obtained a trace of node joins and crashes from a measurement study of Gnutella[7]. The average living time of a node over the trace was about 2.3 hours. All the result is showed in the figure8, 9, and 10.

5 Conclusion

In fact, all the structured overlay networks are the compromise of size of routing table(degree of overlay node) and routing efficiency(diameter of overlay graph). We believe that the diameter of overlay graph is the most important object of peer-to-peer system design, because it impacts on the routing performance of overlay networks while degree of overlay node only has effect on maintenance cost.

The paper considers the trade-off between the size of routing table and the routing efficiency. We present a new structured overlay network that can route message in 2 hops in very high probability and scale to large membership changes. The main feature of Twins is the design of its routing table, the two parts of routing table insure each node keep a small-world manner routing state, which help our system to get high routing efficiency.

Our future works will be focused on exploring great heterogeneity of nodes in real P2P systems, to make Twins protocol more efficiency and scalable.

References

1. Anjali Gupta, Barbara Liskov, Rodrigo Rodrigues. One Hop Lookups for Peer-to-Peer Overlays. HOTOS IX. May 2003.
2. Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, Robbert van Renesse. Kelips: building an efficient and stable P2P DHT through increased memory and background overhead. IPTPS '03. February 2003.
3. Kazaa. `http://www.kazaa.com`. November 2003
4. Petar Maymounkov and David Mazieres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. IPTPS '02. March 2002.
5. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. SIGCOMM 2001. August 2001.
6. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. Middleware 2001. November 2001.
7. Saroiu, S., Gummadi, P. K., and Gribble, S. D. A Measurement Study of Peer-to-Peer File Sharing Systems. MMCN '02. January 2002.
8. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. SIGCOMM 2001. August 2001.