Computing and Informatics, Vol. 24, 2005, 249–261

TWINS: SCALABLE 2-HOP STRUCTURED OVERLAY NETWORK

Jinfeng Hu, Huanan Zhang, Weimin Zheng

Department of Computer Science and Technology
Tsinghua University
Beijing, P. R. China
e-mail: {hujinfeng00, zhanghn03}@mails.tsinghua.edu.cn
zwm-dcs@tsinghua.edu.cn

Manuscript received 2 September 2004; revised 6 April 2005
Communicated by Jiří Šafařík

Abstract. In this paper we propose a new structured overlay network, which is more efficient and scalable than previous ones. We call it *Twins*, because its routing table consists of two parts, one containing nodes with common prefix and the other containing nodes with common suffix. Twins routes messages to their destinations in just 2 hops even in a very large scale and the overhead is very low. When deployed in a peer-to-peer system with 5 000 000 nodes, each node receives only 6 messages per second for routing table maintenance. This cost, as well as routing table size, varies as a $O\left(\sqrt{N}\right)$ function to the overlay scale, so Twins can also run well in an even larger environment.

Keywords: Peer-to-peer systems, overlay networks, routing protocol, scalability, maintenance cost

1 INTRODUCTION

Peer-to-Peer software attracts more and more Internet users in recent years. In a Peer-to-Peer system, each node is both a server and a client, i.e., it provides resources (file content, disk space, Internet bandwidth, etc.) to others, and also obtains desired resources from others. Because of its spirit of equity and autonomy, more and more users use it as a common tool in their daily life. For example, the number of concurrent users of Kazaa increases very rapidly and has exceeded

3 000 000 by far [1]. Furthermore, there is no signal for this trend to cease in the foreseeable future.

Along with the evolution of Peer-to-Peer systems, more and more algorithms are proposed to solve the problems in this special environment, which is larger, more dynamic, and more asynchronous than traditional distributed systems. Structured overlay is a well-known and important one of them.

Structured overlay is an application-level layer which ensures the communication of any two nodes in a Peer-to-Peer system. Existing structured overlays can be classified into two categories: multi-hop overlays and one-hop overlays. The size of multi-hop overlays' routing tables is $O(\log N)$ (such as Pastry [2], Tapestry [3], and Chord [4]), where N is the total number of the nodes. With relatively small routing tables, multi-hop overlays' maintenance cost to handle the member-change events is very low. Therefore, they can achieve high scalability. But the price of this approach is their inefficient message routing that requires $O(\log N)$ hops for each message on average. For example, a typical 16-based Pastry needs about $\log_{16} 3\,000\,000 \approx 5.38$ hops for each message routing.

Being very different with multi-hop overlays, one-hop overlay [5] lets every node keep complete membership information of all the other nodes in the system. It can accomplish most lookup operations in one hop. However, the one-hop overlay's routing table is too large and consumes too much bandwidth for maintenance. Nowadays, the average lifetime of a node is about one hour [6], that is to say, given a system of 3 000 000 nodes, every node must receive information of $3\,000\,000 \times 26\,000\,000$ member-changing events per hour. The data structure of a member-changing event is at least 200 bits, including corresponding node's nodeId, IP address and port. Thus the bandwidth cost must be more than 33.3 kbps, which is too heavy a burden for most modem-linked nodes. So the scalability of the one-hop protocol is very poor.

It is the shortcoming of these two types of existing overlays that motivates us to design a new scalable structured overlay Twins, which is both scalable and efficient. Twins can obtain high routing efficiency (2-hop routing) in a very large scale at a very low cost (6 messages per second within a 5 000 000-node system).

Routing table of a Twins node consists of two parts, one containing pointers to all the nodes sharing a p-bit common prefix with the local nodeId and the other containing pointers to those sharing an s-bit common suffix. Under a simple routing algorithm it can route a message via just 2 hops with a very high probability. Twins adopts a report-based routing table maintenance algorithm. When a node joins or leaves, its state will be multicast to all the nodes that need to know this event. This method saves the maintenance cost greatly. Experimental results show that when deployed in a 5 000 000-node system, each Twins node consumes only 6 messages per second for routing table maintenance. This cost, as well as the routing table size, varies as a $O\left(\sqrt{N}\right)$ function to the system scale N, which indicates that Twins can also work well in an even larger environment.

Moreover, Twins introduces probabilistic message routing into structured overlays. How many hops a message passes before reaching its destination is not strictly determined. This makes sufficient room for scalability design: we can raise the expectation of hops to keep a low overhead by adjusting the system arguments.

The rest of this paper is organized as follows. Section 2 presents the design of Twins protocol. Section 3 gives a formalized analysis of the routing performance and the maintenance cost. Experimental results are reported in Section 4. Section 5 makes final conclusions.

2 THE TWINS PROTOCOL

Like Pastry and Chord, Twins assigns every node an identifier, *nodeId*, using SHA-1 upon its IP address and port, which is typically 128-bit long. The nodeIds are ordered and evenly distributed in the 2^{128}-modulo nodeId circle. For a node, say node M, we call the first p bits of M's nodeId the node's prefix, and the last s bits the suffix.

2.1 Routing Table

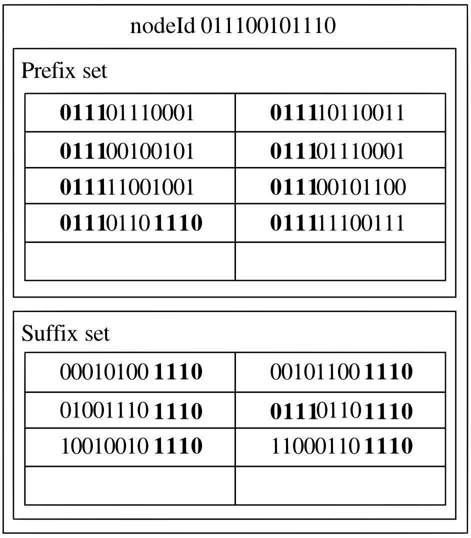
Each Twins node has a routing table consisting of two parts: The first one contains pointers to all the nodes having the same prefix with the local node, called *prefix set*, while the second one contains pointers to those having the same suffix, called *suffix set*.

Obviously if two nodes have the same prefix, their prefix set must be also the same. Thus all the nodes can be divided into 2^p groups according to their different prefixes. We call these groups *prefix groups*. Nodes in a given prefix group are fully interconnected[1] through their prefix sets. Since nodeIds are distributed evenly in the nodeId space, each prefix-group will contain about $N/2^p$ nodes (N is the total number of the nodes). The *suffix group* is defined similarly, i.e., all the nodes are divided into 2^s different suffix groups that do not intersect with one another. There are about $N/2^s$ nodes in each suffix group.

Given a 128-bit nodeId L, we define prefix-group$_p(L) = \{M \,|\, M$ is a 128-bit identifier whose first p bits are the same with L's$\}$, and suffix-group$_s(L) = \{M \,|\, M$ is a 128-bit identifier whose last s bits are the same with L's$\}$, p and s are system parameters.

Figure 1 shows the routing table of a hypothetical node with a 12-bit nodeId 011100101110. In the figure we assume $p = 4$, $s = 4$. It can be seen that the routing table consists of prefix set, which contains pointers to all the nodes whose nodeId has a common 4-bit prefix with the local node, and suffix set, which contains pointers to those whose nodeId has a common 4-bit suffix. For all the pointers, we only show the corresponding nodes' nodeId, omitting their IP addresses and ports. Prefixes and suffixes are all shown in boldface. Notice that the prefix and suffix set may intersect, pointers of node **0111**0110**1110** being an example.

[1] Full connection means that every node keeps pointers to all the others in a node set.



nodeId 011100101110

Prefix set

011101110001	**0111**10110011
011100100101	**0111**01110001
011111001001	**0111**00101100
01110110**1110**	**0111**11100111

Suffix set

00010100**1110**	00101100**1110**
01001110**1110**	**0111**0110**1110**
10010010**1110**	11000110**1110**

Fig. 1. An example of routing table, the local node has a nodeId of 011100101110

2.2 Routing

Every message has a destination key that is also 128-bit long. The first p bits of the key are also called the message's p-bit prefix, and all the nodes that have this prefix are called the message's prefix-group. A message's suffix-group is defined similarly.

With a slight difference with Pastry, the destination node of a message is as follows: The destination node of a message msg is the node in the msg's prefix-group whose nodeId is the closest to msg's key, numerically.

Notice that all the nodes in a prefix-group are fully interconnected, so 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 key k, a node M (say, its nodeId is M) first checks whether k and M have the same prefix (i.e., $M \in prefix - group_p(k)$). If so, M can directly forward it to the destination node, which must be in M's prefix set; otherwise M inspects its suffix set, tries to seek out a node E which has k's prefix and forwards the message to E. For example, when the node shown in Figure 1, namely M, would route a message with key $k = 011100000000$, according to our routing algorithm, $M \in prefix - group_p(k)$, M should forward the message to node 011100100101, which is the closest to k in M's prefix set. In another case where the message key (k) is 001000000000, M will choose node 001011001110 and forward the message to it, because it has the same 4-bit prefix with k.

```
route($msg, k$)          // when a node $N$ routes a message $msg$ with key $k$
  if $N \in \text{prefix} - \text{group}_p(k)$ {
    forward $msg$ to its destination node;
  } else {
    for every node $E$ in $N$'s suffix set do {
      if $E \in \text{prefix} - \text{group}_p(k)$ {
        forward $msg$ to $E$;
        return;
      }
    }
    select a random node $R$ from $N$'s prefix set;
    forward $msg$ to $E$;
  }
```

Fig. 2. Pseudo code of Twins' routing algorithm

If there is no node in M's suffix set that has the same prefix with k, M will forward the message to a random node selected from its prefix set whose nodeId has a different suffix. Pseudo code of the routing algorithm is shown in Figure 2.

2.3 Maintenance

Like the 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 is allowed to just consider the maintenance of the prefix sets of the nodes within one prefix-group.

2.3.1 Node Joining

When a new node X joins the system, it firstly contacts an existing node B, which is called X's bootstrap node. After receiving X's join request, B selects two nodes in its routing table, say P and S, which are in X's prefix group and suffix group, respectively. Then X can establish its routing table by getting its prefix set from P, and suffix set from S. After that X multicasts its joining event around all the nodes in its prefix-group and suffix-group.

2.3.2 Node Leaving

All the nodes in a group can be seen as a circle in the nodeId order. It is demanded that every node probes its right neighbor in the circle periodically. If a node does not respond for several times, it will be considered as a dead one, and its leaving event will be multicast around the group. Figure 3 illustrates the maintenance mechanism within a hypothetical prefix/suffix-group.

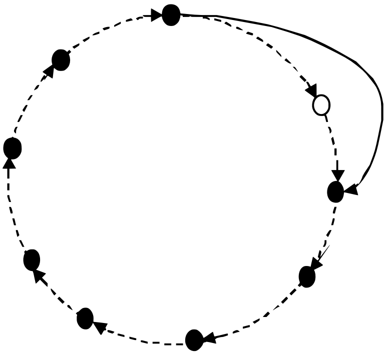


Fig. 3. This is a hypothetical prefix-group. All the nodes in this group are organized as a circle in the nodeId order. Every node probes its right neighbor periodically. We can see that if there is a node (seeing the hollow node in this figure) that does not respond the probing for several times, it will be considered as a dead one. After a node finds its right neighbor's death, it will broadcast the event around the group and turn its probe to the next right neighbor in the circle.

2.3.3 Multicast

Note that all the nodes within a group are fully connected. The multicast protocol can be designed in many ways. Here we adopt a simple tree-based multicast, illustrated in Figure 4, as Twins' basic design.

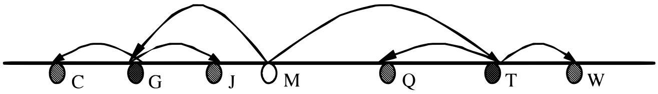


Fig. 4. An instance of the tree-based multicast

When a node M initiates a multicast process, it sends the message to a node before it (say G) and another behind it (say T). Then, similarly, G and T each send the message to two nodes, one ahead and the other behind. This procedure continues. At each step, every node should ensure that once it sends the message to another node, there is no other node between them which has already received the message.

If more efficient multicast is desired, the multicast tree can be modified to be l-based, where l can be an arbitrary value larger than 2. If more reliable multicast is desired, response-redirect mechanism can be deployed, which will increase the maintenance cost.

In this multicast approach, except the changing node's nodeId, IP address and port, an event message (e.g., the message from T to Q in Figure 4) must also include the nodeId of the first node before the receiver that has already received the message (namely M), as well as the nodeId of the first such node behind it (namely T). Adding the UDP header (64 bits) and the IP header (160 bits), an event message will not exceed 500 bits.

Assuming that the average lifetime of the nodes is 1 hour, which is accordant with the measurement result from [6], all the items in the routing table have to be refreshed in a period of 1 hour. It means that for a Twins system that comprises 5 000 000 nodes, if p and s are both set 10, every prefix/suffix-group will contain about $5\,000\,000/2^{10} = 4\,883$ nodes and every node will receive about $(4\,883 + 4\,883) \cdot 2 = 19\,532$ event messages per hour (namely 5.43 messages per second). Plus probes and responses, a node will send no more than 6 messages per second totally, i.e., the bandwidth cost is lower than $6 \cdot 500 = 3000$ bps. So Twins is a very lightweight protocol.

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 p bits and a suffix with s bits, then there are 2^p prefix-groups and 2^s suffix-groups totally. On average, each prefix-group contains $P = N/2^p$ nodes and each suffix-group contains $S = N/2^s$ nodes.

3.1 Routing Performance

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

$$E(L) = \sum_{k=0}^{P-1} k \times \frac{\begin{pmatrix} 2^k \\ k \end{pmatrix} \begin{pmatrix} S-1 \\ 2^k - k - 1 \end{pmatrix}}{\begin{pmatrix} 2^k + S - 1 \\ 2^k - 1 \end{pmatrix}}.$$

For a given message key, we call the probability that there is at least one routing table entry in M's suffix set which has the same prefix with the key as *hit ratio*. Obviously, the hit ratio can be defined as $hr = 1 - \frac{E(L)}{P}$. Consequently, hr is related to the ratio of the size of suffix set (S) to the number of prefix-groups(2^p); we define $R = S/2^p$, and then we can see that when 2^p is larger than 50, choosing R as 4.7 will ensure a hit ratio more than 0.99. Therefore, in common cases we demand that $R > R_0 = 4.7$. That is to say, $N/(2^s \cdot 2^p) > R_0$, namely

$$s + p \leq \log_2 (N/R)\,. \tag{1}$$

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

$$m = (P + S) \times 2 \times f/LF = \left(\frac{N}{2^p} + \frac{N}{2^s}\right) \times 2f \Big/ LF.$$

When $p = s$, m reaches its minimum value:

$$m_0 = \frac{2N}{2^{\frac{1}{2}(p+s)}} \times 2f = \frac{4N \cdot f}{2^s \cdot LF}. \tag{2}$$

Considering both (1) and (2), we should set $s = p = \left\lfloor \frac{\log_2 (N/R)}{2} \right\rfloor$ to ensure a 2-hop routing with a probability larger than 0.99 at a minimal maintenance cost. We can get the following results under this setting:

a) Routing table size R_{size} satisfies

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

b) Maintenance cost m_0 satisfies

$$\frac{\sqrt{N \cdot R} \times 4f}{LF} \leq m_0 < \frac{\sqrt{N \cdot R} \times 8f}{LF}. \tag{4}$$

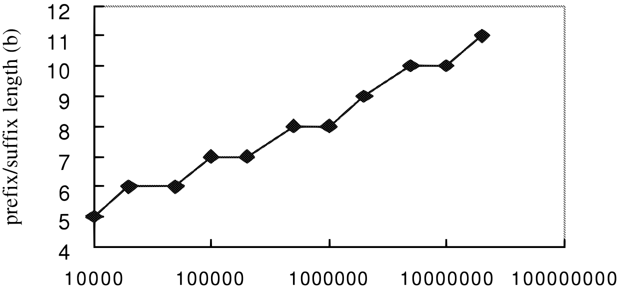


Fig. 5. Prefix/suffix length vs. N

Figures 5, 6 and 7 show the variation of s, R_{size} and m_0 as functions of N, using $f = 1$ and $LF = 3\,600$.

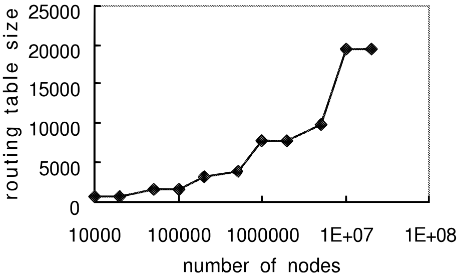


Fig. 6. Routing table size vs. N

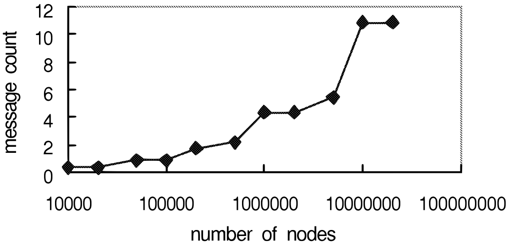


Fig. 7. Message count vs. N

3.2 Scalability

Inequations (3) and (4) show that a Twins node has a routing table containing $O\left(\sqrt{N}\right)$ entries and consumes $O\left(\sqrt{N}\right)$ bandwidth for maintenance. This means a good scalability property of Twins overlay network. In addition, when the maintenance cost is not acceptable by the nodes, Twins can tradeoff routing hops for bandwidth consumption like other overlay protocols. To illustrate it, we put Twins into a stricter environment where $N = 10\,000\,000$, $f = 2$ and $LF = 2\,400$. From 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 s and p, decrease S, and then reduce hr. Expected value of hop counts can be calculated as:

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

When hr drops to 0.25, average hop counts only rise to 5.

4 SIMULATION AND EXPERIMENTAL RESULT

In this section, we present experiment results obtained with a simulation of Twins protocol. The simulator was implemented on ONSP [7], an overlay network simulation platform based on parallel discrete event-driven mechanism which integrates the Transit-Stub model of the Internet generated by the GT-ITM [8] tool. To make Peer-to-Peer protocol simulation on ONSP, a user may only focus on the protocol details, without consideration of the Internet simulation and the parallel task management.

Our experiments are performed on a 16-server cluster that is connected by 2 Gbps Myrinet. Each server has four 700 MHz Xeon CPUs and 1 GB memories, running an operating system of Linux Redhat 7.3.

Both the distributions of node bandwidth and node lifetime are accordant with the measurement result of Gnutella [6]. The average lifetime of a node over the trace was about 2.3 hours.

All the results are shown in Figures 8, 9, and 10.

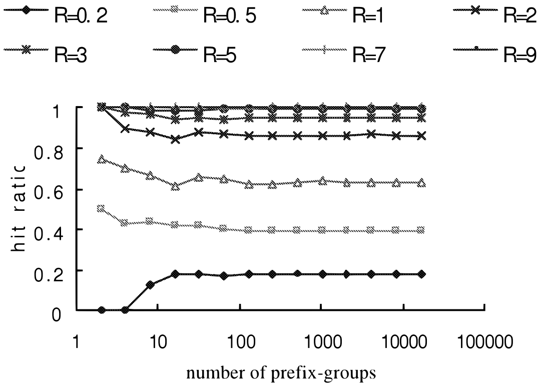


Fig. 8. Hit ratio vs. number of prefix-groups. R is the ratio of the suffix set size (S) to the number of prefix-groups ($2p$). It shows that when $2p$ exceeds 50, hit ratio is almost a constant.

Figure 8 depicts relationship between the hit ratio and the number of prefix-groups. We can see that when the number of prefix-groups exceeds 100, the hit ratio is almost unrelated to it and is only determined by R, that is, the ratio of the suffix set size to the number of prefix-groups. To make it more illustrative, we fix the number of prefixes at 1 024 and depict the relationship of the hit ratio and R in Figure 9. When R is larger than 4.7, the hit ratio is very near to 1.0.

Figure 10 shows the routing efficiency (hop count) and the bandwidth cost (message count) of Twins. We can see that when the system scale is very large (comprising 10 million nodes), Twins can reduce R to save the bandwidth cost, with a price of more hops for message routing. When R drops to 0.5, the hop TRIAL RESTRICTION count only rises to 3.53 while the message count drops to 8.14 from 32.6 where R is larger than 3.0.

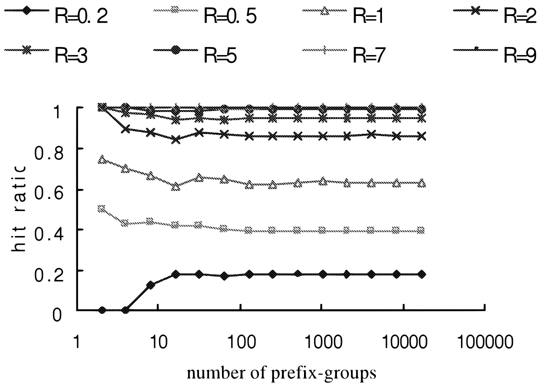


Fig. 9. Hit ratio vs. R. R is the ratio of the suffix set size to the number of prefix-groups. Here the number of prefix-groups is fixed as 1 024.

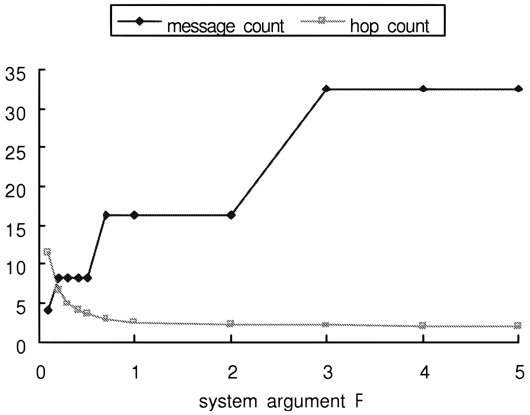


Fig. 10. Message count and hop count in relation to the argument R, where N = 10 000 000, $f = 2$ and $LF = 2\,400$

5 CONCLUSION

In fact, all the structured overlays compromise tradeoff between routing table size and routing hops. We believe that the latter is the most important factor for a peer-to-peer system, because it directly impacts the routing efficiency. Therefore, as long as the maintenance cost can be afforded, we should improve the routing efficiency at the best effort. Twins is a new structured overlay that can route message in 2 hops in very high probability and can extend to a very large scale even in a system with high member-changing frequency. The main feature of Twins is the design of its routing table, which consists of two symmetric parts that ensure 2-hop routing at a very low cost. Our future works will focus on the great heterogeneity of nodes in the real peer-to-peer systems, i.e., upgrading Twins protocol to a heterogeneous one.

REFERENCES

[1] Kazaa. `http://www.kazaa.com`. November 2003.

[2] Rowstron, A.—Druschel, P.: Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. Middleware 2001, November 2001.

[3] Zhao, B.—Kubiatowicz, J.—Joseph, A.: Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing. Technical Report UCB/CSD-01-1141, U. C. Berkeley, April 2001.

[4] Stoica, I.—Morris, R.—Karger, D.—Kaashoek, M. F.—Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. SIGCOMM 2001, August 2001.

[5] Gupta, A.—Liskov, B.—Rodrigues, R.: One Hop Lookups for Peer-to-Peer Overlays. HOTOS IX., May 2003.

[6] Saroiu, S.—Gummadi, P. K.—Gribble, S. D.: A Measurement Study of Peer-to-Peer File Sharing Systems. MMCN '02, January 2002.

[7] Wu, Y.—Li, M.—Zheng, W.: ONSP: Parallel Overlay Network Simulation Platform. The 2004 International Conference on Parallel and Distributed Processing Techniques and Applications. PDPTA '04, June 2004.

[8] Calvert, K. L.—Doar, M. B.—Zegura, E. W.: Modeling Internet Topology. In IEEE Communications, 1997.

Jinfeng Hu received his Ph. D. and B. Sc. degrees at the Department of Computing Science and Technology, Tsinghua University, P. R. China in 2005 and 2000, respectively. His research interests include large-scale distributed computing, global storage system, peer-to-peer computing, etc.

Huanan Zhang is a graduate student at the Department of Computing Science and Technology, Tsinghua University, P. R. China, where he received his B. Sc. degree in 2003. His research interests include peer-to-peer computing, overlay networks, information plane, etc.

Weimin Zheng is a professor at the Department of Computing Science and Technology, Tsinghua University, P. R. China where he received his M. Sc. degree in 1982, and his BS degree at the Department of Automation, Tsinghua University in 1970. His research interests include distributed computing, cluster system, Grid computing, peer-to-peer computing, parallel compiling, parallel program debugging, etc.