Building a Peer-to-Peer Overlay for Efficient Routing and Low Maintenance

Honghao Wang and Yiming Hu

University of Cincinnati, Cincinnati, OH, 45219, USA

Abstract. Most of current structured P2P systems exploit Distributed Hash Table (DHT) to archive an administration-free, fault tolerant overlay network and guarantee to deliver a message to its destination within $O(logN)$ hops. While elegant from a theoretical perspective, those systems face difficulties in a realistic environment. Instead of building P2P overlays from a theoretical perspective, this design tries to construct an overlay from the physical network. By combining different network topology aware techniques, a distinctive overlay structure closely matching the Internet topology is created. The P2P overlay based on this structure is not only highly efficient for routing, but also keeps maintenance overhead very low even under highly dynamic environment.

1 Introduction

Most of current structured P2P overlays exploit Distributed Hash Table (DHT) to archive an administration-free, fault tolerant overlay network and guarantee to deliver a message to its destination within $O(logN)$ hops.

While elegant from a theoretical perspective, DHT-based systems face difficulties in routing efficiency and high overhead in a realistic environment. Although DHT designs guarantee to solve a query within $O(logN)$ hops, the previous study [1] has shown that a significant fraction of nodes could be connected over high latency / low bandwidth links. The presence of even one such slow logical hop on a logarithmically long path is thus likely. This increases the overall cost of the lookup. Recent DHT designs have tried to optimize routing by exploiting physical network information, however, latest research [2] has pointed out that the latency of last few hops in a lookup still approximated 1.5 times the average round trip time under Proximity Neighbor Selection (PNS). The situation will be worse under dynamic environment. As mentioned by Rhea et al. in [3], for 1000 nodes system under modest churn rate, a Pastry system (FreePastry) failed to complete 70% requests. While almost all lookups in a Chord system were completed, the lookup latency increased more than 20 times.

In order to handle system churn, the node in the latest Pastry, Bamboo DHTChurn system is designed to periodically detect its neighbors and share the leaf set with them. The overhead of those operations is substantial. The bandwidth is about 1.8kps per node, and the traffic is more than1.5TBytes daily for a 100,000 system, which is 7.5 times larger than the Gnutella system [4].

In order to build an efficient structured P2P overlay in a realistic environment, this paper, from a different angle, proposes a new protocol. Opposite to current P2P designs,

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 766–775, 2005.

our approach focuses on building the system overlay closely matching the physical network. As a result, physical network characteristics, such as the power law of the Internet, network locality among nodes and asymmetric throughput of major network connections, are fully exploited. By a novel piggyback mechanism, the system subtly integrates maintenance work into a common operation. As a result, the overhead of the system maintenance is reduced to the minimum.

The rest of this paper is structured as follows. Section 2 presents the background of the Internet topology and related techniques. Section 3 describes in detail how to construct the system overlay closely approximating the Internet topology and build an highly efficient system on it. In section 4, we evaluate this approach using simulation. Section 5 compares it to related work. Section 6 concludes the paper.

2 Internet Structure and Related Techniques

The Internet is made up of many Autonomous Systems (ASes). An AS, normally an ISP or organization like companies or universities, is a network under a single administration authority using a common Interior Gateway Protocol (IGP) for packet routing. Machines within an AS are normally geographically close and connected by LAN/MAN techniques. Some border routers running Border Gateway Protocol (BGP) connect it with neighboring ones. There are many methods and techniques to acquire network aware information, such as BGP routing tables [5, 6], widely used landmark technique [7] and the network of physical springs [8].

BGP routing tables can provide router-level Internet topology. However, they are not only hard to obtained, but also too complex to use. For AS-level information, lots of public services, such as the CIDR Report [6] and WHOIS service, are available. All information, such as IP address span, AS number and connectivity, are announced. Some service even updates daily and provides thorough analysis. By measuring RTTs between particular node and other nodes or pre-selected hosts (landmarks), landmark techniques and the Vivaldi can provide some kind of coordinates to reflect node's relative position in the Internet. Since many unpredictable factors, such as changes of routing, network bandwidth and traffic, can affect RTTs, those techniques are not that accurate and stable, and do not directly reflect network topology.

Above techniques will be used together in our design to ensure the overlay structure close matching the physical network, which will be discussed later.

3 Overlay Design

In this section, we will firstly discuss how to construct the overlay closely approximating the physical network, and then to build an efficient P2P overlay based on it.

3.1 Building an Overlay Based on Internet Structure

As Figure 1 shows, based on published AS-level Internet topology, all nodes are divided into *groups* by their AS locus. The group is the basic unit for routing and organizing nodes in the system. However, to partition nodes only by the AS may be too coarse,

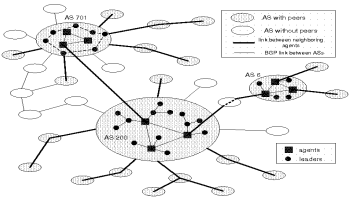

Fig. 1. Building the system overlay closely matching Internet topology

especially for large ASes with lots of nodes. Thus, we propose to use landmark technique to further divide nodes into *teams*. The default routers of previously joined nodes would be good candidates for landmarks. As Ratnasamy et al. mentioned in [7], 6 to 8 random selected landmarks were enough to divide a network with thousands of hosts. Hosts will be grouped by their latency ordering of landmarks, which is called *landmark vectors*.

Several nodes with high bandwidth and availability will be elected within each group as *agents* to perform routing for the group. A leader will as be selected to route for the team. Although each peer arbitrarily joins and leaves the system, previous researches [1, 4] have showed that their behaviors were highly skewed. Most of nodes have very short uptime, however, there are 18% and 10% nodes with 90% uptime in Napster and Gnutella, respectively. Also, those 10% hosts also contribute 90% of the total traffic. Thus, those 10% nodes are appropriate candidates for agents and leaders. The more detail will be addressed in later sections.

3.2 Peer-to-Peer Overlay Design

In order to support ID lookup operation like structure P2P overlays, the similar ID mechanism is employed. Each object in our overlay has a 128bits ID, which can be generated by a basic hash function such as SHA-1. Instead of mapping a small range of objects to each node, a two levels mapping mechanism is used in our overlay. The first level is among groups. The second level is among teams. Since valid Internet AS number is from 1 to 64511 and only about 17,000 AS is active currently, a 32bits ID for groups is considered enough. Also, each team will be assigned a 32bits team ID. Each group and its teams will be assigned the unique ID randomly when they first appear. Each group will store the objects with the first 32bits between its group ID and the next one, which is similar to Chord [9]. Within a group, the object will be further mapped into a team by its second 32bits. In other words, given a 128bits object ID, the first 32bits is used to locate its responsible group, and the second 32bits is used to find the related team.

The team is the basic unit to store objects. Two copies of objects will be kept within a team for improved reliability and availability. One copy will be kept in the leader to reply queries for other peers. The other one will be striped into blocks by erasure code technique and store among teammates. Previous researche [2] has pointed out that erasure code technique can significantly improve dependability, and reduce bandwidth for updating objects. The only drawback is the read latency. However, our design successfully solves this problem. Since more than 50% peers are connected with asymmetric

network connections, such as DSL and cable modems, to retrieval information from several nearby peers will be evidently faster than from one. Also, this characteristic is helpful to quickly recover failure leaders and agents. The number of nodes within a team will be varied to facilitate clustering physical nearby nodes and keep team stable under extremely churn. Our experiments show that the suitable team size can be from 10 to 50 nodes.

Routing. The routing mechanism is actually simple compared with DHT designs. As mentioned early, since a two levels mapping mechanism is used in the overlay, the routing table is made up of two tables. One records each group's ID and information of agents within the overlay, called *routing table*. This table is maintained in each agent and leader to route messages for normal nodes. The other one is only kept in each agent for its group. It records each team's ID and leader's information within the group, called *delivering table*. Given an object ID, the responsible group and agents can be located by the routing table. Instead of hop by hop approaching the destination, messages will go directly to the agent in the target group. When the message reaches the agent, it simply forward that message to the response team leader by the delivering table. Finally, the leader replies the message. Normally, 3 agents can provide enough availability and performance without loading the host machines. A detailed analysis is given in section 3.3.

Node Joining and Leaving. The procedure of nodes' joining and leaving is simple. When a node N joins the system, its AS number can be determined by its IP by published services. The bootstrap protocol is straightforward. By any node within the system, the joining request will be forwarded to an agent of the node N's AS. After measuring N's landmark vector, the node will join one team according to its network locality. The overhead is minimal, since only the leader has to update some book-keeping information. If a team is too populous, it will split into two based on network locality. If the joining node is the first one in the AS, that request will be sent to a physical nearby group. Instead of forming a new group, the node will become a teammate within that group, called *mother group*. When nodes within that AS are enough for three teams, agents will be selected and an individual group is born.

For the normal node, its leaving or failure is automatically tolerated by the team. When an agent or a leader is leaving the system, a new one will be selected. As mentioned early, datum from different nodes can rebuild the routing tables and objects for the new one in seconds. Its information will quickly spread out by a piggyback mechanism, which will be discussed in the following section.

3.3 Maintenance

In P2P environment, not only each peer's leaving and failure is unpredictable, but also the whole system is highly dynamic. Thus, to build a system of efficiency and low maintenance under such dynamic environment is really a challenge for all P2P overlays. Although current DHT designs can tolerant nodes' failure, latest research [3] has showed that those systems degenerated quickly under dynamic environment.

Although agents and leaders are considered to have higher availability than normal nodes, they are not well-maintained servers. The leaving of agents and leaders will not

Table 1. All traffic generated by different roles and system functions. * is the ratio to DSL or calble modem connection with 3Mb downlink and 384kb uplink.

System Operations	Agents (Bps)		Leaders (Bps)		Nodes (Bps)	
	Downlink	Uplink	Downlink	Uplink	Downlink	Uplink
Lookups	666.67	666.67	200	200	10	10
Ring Protocol	20	20	26.67	20	0	0.33
Update Others	44	264	0	0	0	0
Piggyback	400	0	0	120	0	0
Total	1131 (0.38%*)	951 (2.48%*)	271 (0.09%*)	560 (1.46%*)	10	10.33

impact the system much, since their datum can be rebuilt quickly. Although node's crash is not considered a common case, it should be quickly detected and recovered. A modified ring protocol [10] is used to monitor agents and leaders. As the AS 701 showed in Figure 1, all leaders will form a ring as their ID relationship. Every second, each node will send a keep-alive message to its successor and predecessor. By combining reports from other leaders, the agent can accurately find the crash leader. The same method is also used between agents. Normally, the leader can monitor teammates' changing through their query messages.

Since the leaders cache the routing table from group agents, some method is needed to keep them consistence. For small groups, like the AS6 in Figure 1, leaders can directly exchange information with a nearby agent. However, this simple schedule is not suitable for large AS. Since nodes are organized as the AS, a same network administration, administratively Scoped IP Multicast can be exploited. For the AS without multicast, *dissemination trees* with agents as the root will be used to multicast the information to all leaders, as the structure of AS200 in Figure 1 shows. For even larger AS, it will be partitioned into two or more groups.

To keep those routing tables up to date is critically important for lookup correctness. Also the overhead to maintain them should be low. Otherwise, both system performance and scalability will be affected. Thus, our overlay is designed to integrate this work into the common operation, *lookup*. When the leaders send out or answer messages, the information of changed agents, which is about 4 bytes for one agent, will be appended to the messages and reach an agent within the destination group, and then spread to all leaders. After that, the information will be further sent to more groups with outing messages. Actually, this gossip-style piggyback mechanism is highly efficient and robust. It is well known that with high probability all groups will be informed within $O(logN)$ turns, in which N is the number of groups. Assuming one turn costs about 3 seconds, which is the time for an update to reach and spread in a group, a system with 5000 groups for one million nodes could be updated within 4 turns, 12 seconds. Our experiments show that in a system with 10,000 nodes distributed in 100 ASes, assuming a median session time of 15 minutes for each node, the success rate of first attempt with the piggyback technique is 99.53%.

Scalability Analysis. Currently, the hosts of real world systems are distributed in 4 to 5.5 thousands ASes. The total number of nodes is from 200,000 to 1,000,000, and the number per AS varies from several to thousands [4]. Thus, the size of routing table with

5000 groups and their agents' information is about 60KB, and size of delivering table with 300 teams is about 3KB.

Previous researche [3] has pointed out nodes' joining and leaving in the P2P environment can be modeled by a Poisson process. Thus, an event rate λ corresponds to a median inter-event period of $\ln 2\lambda$. Therefore a churn rate of λ corresponds of a median node session time of N nodes within a network is the following.

$$t_{med} = N \ln 2/\lambda \tag{1}$$

Considering an overlay with one million nodes, distributed in 5000 ASes with 3 agents each group, the median session time of 15 minutes for agents, the churn rate for 15,000 agents is 11 per second by Formula 1. The total bandwidth to deliver the information within a group by router-based multicast protocol is 44Bps for downstream. As for the dissemination trees, the bandwidth is 220Bps for 5 fanouts per agents and middle leaders. By assuming average 20 nodes a team, average query rate of 0.5 per node/second, piggybacking three agents information each time, and 20 bytes per message, Table 1 shows the break down of bandwidth of every role and operation. As Table 1 shows, the extra overhead for agents and leaders is trivial. Even the load of agents or leaders are highly skewed, the overhead is not a burden for most of peers with DSL or cable modem connections.

4 Simulation and Experimental Results

In this section, we compare our design to Chord by simulation with real Internet AS-level topology and latency.

The Internet latency datum used in the simulation is from the King method [11], which has measured the network latencies of more than 1700 DNS servers. The AS-level topology graph is from the CIDR Report [6]. By mapping each DNS server to its AS, a 1701 nodes (ASes) graph with their network latency and topology has been formed. The average round trip delay between nodes pairs is 168ms, and the average AS path length is 2.97 hops. The p2psim [12] has been used to simulate the Chord protocol with Proximity Neighbor Selection (PNS). All experiments were performed in a Dell Dimension PC, which has one 2.8GHz Pentium IV processor and 1.2GB RAM, running Linux.

In order to make the experiments close to the deployment environment, the density and distribution of nodes of real P2P overlay are needed. The previous research [4] has showed that the average nodes for KaZaA and Gnutella were about 200 and 60 per AS, respectively. Also, all nodes distributed in about 5000 ASes. The host density, connectivity and traffic volume of P2P systems are highly skewed and exhibited heavy tails, which can be approximated by Zipf's distribution. As a result, our experiments are modeled to have 10,000 nodes over 100 ASes under the Zipf distribution. The 100 ASes is carefully chosen from the 1701 ones to keep almost equal latency and AS path length with the original one. For the Zipf distribution, the largest AS has 1,000 nodes and the smallest one has 36 ones. Also, the uniform distribution has been simulated for comparison.

We use metrics of lookup latency stretch and failure rate for overlay performance, and bandwidth per node and link stress for overhead. In the simulation, each node alternately crashed and re-joined the network; the interval between successive events for

each is exponentially distributed with a mean time from 15 minutes to 1 hour. Every node issues lookups for random keys at intervals exponentially distributed with a mean of 10 seconds. The p2psim is configured with successors number of 16, base of 16 and refreshing intervals of 1 minute for both successors and fingers. Actually, this is a normal and modest configuration without favoring either request successful rate or consumed bandwidth, according to the research of Li et al. [12]. Our overlay is configured to have 3 agents for a group. The maximum nodes number of a team is 19, each leader is directly connected to an agent, and each message can piggyback up to 3 agents' IP addresses.

4.1 Experimental Results

The picture on the left of Figure 2 shows the result of the comparison of lookup latency stretch of the two overlays. The stretch is the ratio between the latency of the overlay routing to the ideal one. All protocols time out individual messages after an interval of three times the round-trip time, and the message is considered failure. The stretches include both successful and failure ones. As the figure shows, compared with the Chord protocol, our overlay is much more efficient even highly dynamic environment, the latency stretch is nearly the ideal one. Actually, the stretch under uniform distribution is a little better than skewed one for slightly higher successful rate. However, the skewed node distribution favors the Chord protocol with PNS supporting, since an AS with lots of nodes can facilitate a message to quickly approach the destination without traipse. Although the Chord protocol with PNS gets the benefit from skewed nodes distribution, its stretch increases quickly with the extent of system churn.

The picture in the middle of Figure 2 shows the request failure rate of different protocols. The system churn significantly impacts all P2P overlays. As the figure shows, the request failure rates of all protocols increase quickly with churn. Although the Chord with PNS can make a little benefit from skewed node distribution, its failure rate rapidly raises to more than 5% under 15 minutes mean node life time. Opposite to the Chord protocol, the failure rate of our protocol under uniform distribution is evidently better than skewed one. This is because that skewed distribution may exacerbate the churn in our overlay, since it will be harder to find more stable nodes for agents and leaders in some very small groups. Although the failure rate of our protocol is also increased fast, it will be no more than 1%. Also, our design is much better than Chord under all

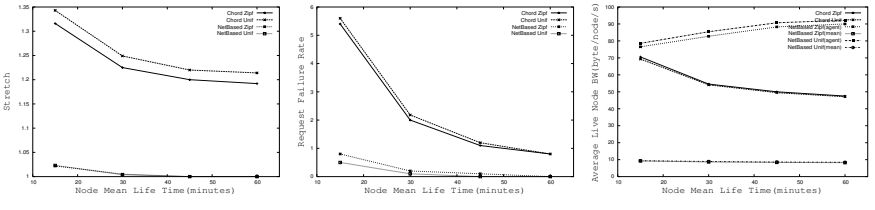

Fig. 2. Comparison of stretch, request failure rate and bandwidth of different protocols and roles, under different system churn rates and node distributions

configurations. We argue that to involve unstable nodes into routing may not be a good choice, though the DHT can tolerate them.

The picture on the right of Figure 2 shows the bandwidth consumed by different protocols and roles. Since our protocol is asymmetric one, the bandwidth of both agents and the whole overlay are illustrated. Although the bandwidth of agents of our overlay is significantly larger than Chord, it is not a burden for most peers with high speed network connections as analyzed early, and they are no more than 3% of the overlay. For the mean bandwidth of the overlay, our protocol is highly efficient compared with Chord, and it is not sensitive to system churn.

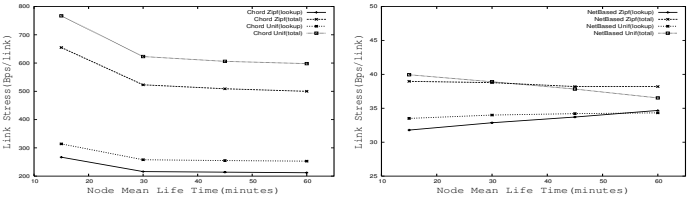

Fig. 3. Link stresses put by different protocols to the Internet back bone, under different system churn rates and node distributions. The left one is for Chord protocol, the right one is for network based one.

While the bandwidth reflects the overhead of each peer, it can not indicate the traffic of the whole overlay. By mapping each node to its exact AS number and finding out the paths between them, we evaluate the traffic impact to the Internet back bones. By recording each link used by every message, the link stress has been computed. Figure 3 shows the link stress of different protocols. Both the total overlay link stress and the lookup (useful) one are presented. As the figure for Chord illustrates, system churn will significantly increase the P2P traffic in the Internet. Compared with uniform node distribution, skewed one generates less traffic due to more communication within one AS. However, Chord protocol is far from efficient. Not only is the average traffic value 12 to 20 times larger than ours, but also the useful (lookup) one is just about 40% of all. In another words, the useless (maintenance) overhead consumes most resource of the whole overlay. The link stress of our protocol is much less than the Chord, since most of maintenance work is accomplished with the same network. Moreover, the useful (lookup) part is more than 80%.

5 Related Work

In order to improve routing efficiency, physical network information is exploited in many DHT designs. Current works can be classified into three main categories: proximity routing, topology-based node ID assignment and proximity neighbor selection [13].

Proximity routing is employed in Chord [9] and their improvements. While physical network information is not taken into account when building system overlay. heuristic algorithms are used to choose many hops with small latency instead of large latency

ones. Topology-based node ID assignment is employed in CAN. When a new node joins the overlay, it joins a node that is close to it in IP distance. Proximity neighbor selection is employed in Pastry [14]. Routing table entries are selected according to the proximity metric among all peers that satisfy the constraints of logical overlay.

In Brocade [15], a secondary overlay network of supernodes based on AS-level topology is used to improve routing performance. Nodes in the default network establish direct connection to a supernode nearby. At the same time, supernodes collect the information of connected nodes and advertise their information in the second overlay. Although their benefits are limited by logical overlay restrictions, those systems produce significant improvements compared to original designs.

Some constant time lookup schemes have been proposed for P2P environment. Gupta et al. [16] proposed to maintain accurate routing table with complete membership in every node through quickly information dissemination. Although their design could reach the optimal stretch, the overhead to maintain the routing tables in each node was real expensive. Kelips [17] and HiScamp [18] were proposed to use gossip-style protocol and keep more states within each node to solve a key within $O(1)$ hops. However, their convergence time for an event, such as node's joining and leaving, was too long.

6 Conclusions

Aware of the difficulties faced by current structure P2P designs, from a different angle, this paper proposes a new approach to build P2P overlay. Our approach focuses on the physical network, and builds the overlay closely matching it. Thus physical network characteristics, such as the power law of the Internet, network locality and asymmetric network connections, are naturally and fully exploited. The whole system not only keeps routing highly efficient, but also adapts extremely system churn by low maintenance overhead.

References

1. S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, (San Jose, CA), January 2002.
2. F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris, "Designing a DHT for low latency and high throughput," in *USENIX First Symposium on Nerworked Systems Design and Implementation(NSDI'04)*, Mar. 2004.
3. S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," in *Proceedings of the USENIX Annual Technical Conference*, 2004.
4. S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks," in *In Proc. ACM SIGCOMM Internet Measurement Workshop, Marseille, France, Nov. 2002.*, 2002.
5. Routeviews.org, "Route Views Archive." http://www.routeviews.org.
6. CIDR-Report, "The CIDR Report." http://www.cidr-report.org.
7. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, 6 2002.
8. R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," in *Proceedings of the 2004 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2004.

9. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, (San Diego, CA), pp. 149–160, 2001.
10. L. M., A. S., and F. A., "A Efficient Algorithms to Implement Unreliable Failure Detectors in Parially Synchronous Systems," in *Proceedings of the 13th Symposium on Distributed Computing (DISC'99)*, (Bratislava, Slovaquia), 1999.
11. K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts," in *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002)*, (Marseille, France), November 2002.
12. J. Li, J. Stribling, R. Morris, M. Kaashoek, and T. Gil, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn," in *Proceedings of 24th IEEE INFOCOM*, March 2005.
13. M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in Peer-to-Peer overlay networks," in *International Workshp on Future Directions in Distributed Computing (FuDiCo)*, 2002.
14. A. Rowstron and P. Druschel, "Pastry: Scalable, decentraized object location and routing for large-scale peer-to-peer systems," in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, (Heidelberg, Germany), Nov. 2001.
15. B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz, "Brocade: Landmark routing on overlay networks," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, (Cambridge, MA), 2002.
16. A. Gupta, B. Liskov, and R. Rodrigues, "Efficient Routing for Peer-to-Peer Overlays," in *USENIX First Symposium on Nerworked Systems Design and Implementation(NSDI'04)*, Mar. 2004.
17. I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, (Berkeley, CA), 2003.
18. A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie, "HiScamp: self-organizing hierarchical membership protocol," in *Proceedings of the 10th European ACM SIGOPS workshop*, Sept. 2002.