An Efficient and Secure Peer-to-Peer Overlay Network

Honghao Wang, Yingwu Zhu and Yiming Hu
Department of Electrical & Computer Engineering and Computer Science
University of Cincinnati
{wanghong, zhuy, yhu}@ececs.uc.edu

Abstract

Most of structured P2P overlays exploit Distributed Hash Table (DHT) to achieve an administration-free, fault tolerant overlay network and guarantee to deliver a message to the destination within $O(logN)$ hops. While elegant from a theoretical perspective, those systems face difficulties under an open environment. Not only frequently joining and leaving of nodes generate enormous maintenance overhead, but also various behaviors and resources of peers compromise overlay performance and security.

Instead of building P2P overlay from the theoretical view, this paper builds the overlay network from the perspective of the physical network. By combining different network topology-aware techniques, a distinctive overlay network that closely matches the Internet topology is constructed. The P2P system based on this structure is not only highly efficient for routing, but also keeps maintenance overhead very low even under highly dynamic environment. Moreover, the system provides a new aspect to solve many security issues.

1. Introduction

Structured peer-to-peer (P2P) overlays, such as Chord [17], CAN [8], Tapestry [20] and Pastry [14], are based on Distributed Hash Tables (DHT) to organize all nodes in the system into node ID ring to achieve an administration-free, fault-tolerant overly network and guarantee to deliver a message to the destination within $O(logN)$ hops, where N is the number of peers in the system. While elegant from a theoretical perspective, DHT-based systems face difficulties in an open Internet deployment.

Due to various behaviors and resources of peers in the real world, the basic operation, the key lookup, of current P2P overlay is far from efficient and hard to be improved. Although DHT designs guarantee to solve a query within $O(logN)$ hops, a single hop may across two nodes connected via a high speed LAN, or two separated by a low-bandwidth, long-latency link across half the world, which will cause significant latency. Proximity Neighbor Selection (PNS) were widely used to optimize overlay routing. However, latest research [4] has pointed out that the latency of last few hops in a PNS based routing still approximated 1.5 times the average round trip time in the underlying network. In order to handle system churn, frequently joining and leaving of nodes, overlays should repeatedly fresh their routing tables, which generate enormous maintenance traffic. Based on the default maintenance periods for Bamboo [13, 12], a latest designed DHT system, the total maintenance traffic is about 7.5 times larger than the same Gnutella system [15].

Moreover, P2P overlays face serious security attackers from malicious nodes, though they have mechanisms to tolerate fault nodes. Many researches [1, 5, 16] have been done on different aspects of P2P overlay networks, such as secure routing and object storage/retrieval. However, due to the complexity of the P2P environment, some solutions are not satisfied, and others are still open.

As Castro et al. mentioned in [1], the secure routing of a P2P overlay network requires the solution of three problems: securely assigning node IDs to nodes, securely maintaining the routing tables, and securely forwarding messages. Although certified node IDs can prevent nodes to compromise the integrity of the overlay by choosing IDs by themselves, an attacker may swap certificates among nodes it controls to increase the fraction of attacker's nodes in target nodes' routing tables. Also, an attacker may obtain a large number of legitimate node IDs, which is called a *Sybil attack* [5]. For securely maintaining routing table entries, while imposing strong constrains on node IDs in the routing table can limit the effect of hostile nodes, it trades off the performance of overlays. Since fault nodes may drop the message passing by, routing the message to the wrong place, or pretending to be the root of the message, failure test and redundant routing are normally used. However, those methods are either hard to be implemented or involve significant

Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)
0-7695-2421-4/05 $20.00 © 2005 IEEE

overhead.

This paper, from a different angle, proposes a new P2P overlay structure. Unlike current P2P designs, which focus on construct an elegant system overlays, such as node ID ring, our approach focuses on the physical network, and builds system overlay based on it. Thus, the physical network characteristics, such as network locality among nodes, network administration among Internet Autonomous Systems and asymmetric throughput of major network connections, can be fully exploited. Not only is the overlay highly efficient in message routing and maintenance, but also it provide novel methods to solve security problems.

The rest of this paper is structured as follows. Section 2 describes briefly how to construct a overlay network closely matches the Internet topology and how to build an high efficiency and low maintenance system on it, the details of the overlay network can be found in another paper [18]. Based on this novel overlay network, Section 3 discusses in detail how to exploit the overlay structure to solve security issues faced by P2P overlay networks. In section 4, we evaluate our approach using simulation. Section 5 concludes the paper.

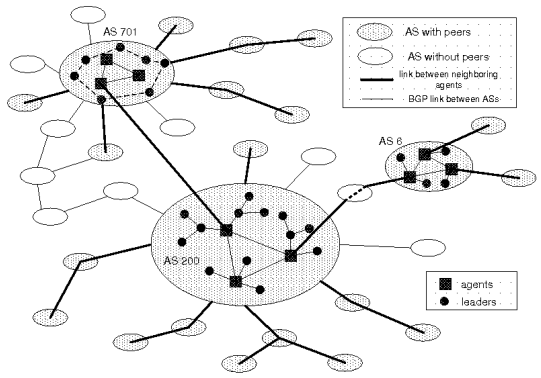

Figure 1. Building the system overlay closely matches the Internet topology

2. Overlay Design

In this section, overlay design will be briefly discussed. We will firstly discuss how to construct an overlay closely approximates the Internet topology, and then based on that overlay to build a P2P system with efficient lookup, low maintenance overhead and high availability.

The essential behind the overlay structure is to organize nodes by their physical network locality in the Internet. As Figure 1 illustrates, based on the Internet AS-level topology provided by [2, 7], all nodes can be divided into *groups* by their Autonomous Systems (AS) locus. Since the Internet is made up of ASes and each AS is a network under a single administration authority, it provides a good border for P2P nodes. Like the AS in the Internet, the group is the basic unit for routing and organizing nodes in the overlay. However, to partition nodes only by nodes' AS residences may be too coarse in many cases. A landmarks based vector technique [9] is used to further divide network physical nearby nodes into *teams* within an AS. The landmarks could be the default routers of previous joined nodes.

2.1. Peer-to-Peer Overlay Network

In order to support DHT-like ID lookup operation, similar ID mechanism is employed in the overlay. Like current DHT designs, each object in the overlay is assigned a 128bits ID. Instead of mapping a small range of objects to each node like current overlays, a two-level mapping mechanism is used. As mentioned early, our overlay is built up with AS-like groups. The first level is among those groups, and the second level is among teams. The number of nodes within a team is variable to facilitate clustering physical nearby nodes and increasing stabilization under system churn. Normally a team includes 10 to 50 nodes. Each team will choose a leader with decent network bandwidth and availability. Also, a 32bits team ID is assigned to each team. Those IDs are assigned to every group and their teams randomly as they first appear in the overlay. The first 64bits of the object ID is separated into two parts. The first 32bits is used to map group ID, and the second 32bits is for team ID. Each group charges the objects with the ID between itself and the next group, and so is the team, which is similar to Chord [17].

The team is the basic unit to store objects. Two copies of objects will be kept within one team for improved reliability and availability. One copy is kept in the leader to respond queries for all other peers. The other one will be striped into blocks by erasure code technique and stored among teammates. Since most peers are connected with asymmetric network connections, such as DSL and cable modem, to read from many nearby peers will be evidently faster than from one. This characteristic is very helpful to quickly rebuild a new leader when the old one fails.

2.1.1. Routing Instead of involving many unstable nodes into system routing, only selected nodes, which have decent bandwidth and availability, will become the agents to route for the overlay. Normally, 3 agents can provide enough availability and performance without impacting their hosting machines. A detailed analysis is discussed in section 2.4.

Since our overlay closely matches the Internet topology, the routing mechanism is actually simple compared with current DHT designs. Since a two-level mapping mechanism is used in the overlay, the routing table is made up of two parts. One records the ID and agents information of each group 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 agents. It records the ID and leaders information of all teams within a group, called *delivering table*. Given an object ID, the responsible group and its agent can be located by the routing table. Instead of approaching the destination hop by hop, a message will go directly to an agent with the target group. When the message reaches that agent, it is simply forwarded to the response team leader by the delivering table. Finally, the leader resolves and replies the query.

2.2. Node Joining and Leaving

The procedure of nodes' joining and leaving is simple. When a new node joins the overlay, it's AS locus can be determined by its IP address. The bootstrap protocol is straightforward. Through any node within the overlay, the joining request will be forwarded to an agent of that AS. After measuring its landmark vector, the node will join one team according to the network locality. If a team is too populous, it will split into two teams based on nodes' network locality. If the joining node is the first one in the AS, the join request will be forwarded to a physical nearby group. Instead of forming a new group, the node will initially become a teammate within the group, called *mother group*. When nodes within that AS are enough for three teams, agents will be selected and an individual new group is born.

For the normal node, its leaving or failure is automatically tolerated by the team. If a leader or agent leave, a new one will be selected, and it information will quickly spread by a piggyback mechanism, which will be discussed later.

2.3. Maintenance

Under the P2P environment, not only each peer's leaving and failure is unpredictable, but also the whole overlay is highly dynamic, since every node can arbitrarily join and leave the overlay. This is really a challenge for all P2P overlay networks.

Although agents and leaders are considered to be more available than normal one, they are not well-maintained servers. A *ring* protocol is used here to monitor them. For example, all leaders will form a ring as their ID relationship, and every second each node sends a keep-alive message to its successor and predecessor. Although this protocol is simple, it is enough to detect one or more leaders' failure. The same method is also used among agents. Normally, the leader of a team can monitor teammates' changing through their query messages. However, in order to accurately distinguish each node's behavior for candidates of leaders and agents, each node is designed to send either a query or a keep-alive message to its leader every 30 seconds.

Since leaders cache the routing table from group agents, some method is needed to keep them consistence. For small groups, which have tens of teams like the AS6 in Figure 1, leaders can directly exchange information with a nearby agent. However, if a group is very large, which has hundreds of teams like the AS200 in the figure, such simple schedule will not be scalability. Fortunately, all nodes within a group in our system are under the same network administration, i.e. AS. Thus, Administratively Scoped IP Multicast [10], which is widely supported by current routers, can be exploited to efficiently deliver latest agents information. For the AS without multicast protocol supported, *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 demonstrates. When the group is even larger, it will be partitioned into two or more ones.

As mentioned early, each agent maintains a routing table which records all agents' information. To keep those routing tables up to date is critically important to the correctness of lookups. Also the overhead of such maintenance should be low, otherwise, both system performance and scalability will be impacted. Thus, our overlay is designed to integrate this maintenance work into common operations, *lookups*. When the leaders send out or answer messages, latest information about changed agents, which is just 4 bytes for one agent, will be appended to the messages reach an agent of the destination group, and then spread to all leaders. Again, that information will be sent out to more groups, and finally reaches every group. Our experiments reveal that in an overlay with 20,000 nodes distributed in 200 ASes and a median session time of 5 minutes for each agent, the success rate of first attempt with such piggyback technique is 98.7%.

2.4. Scalability Analysis

Currently, hosts of real world systems are distributed in 4 to 5.5 thousands ASes, the total number is from 200,000 to 1,000,000, and the density of hosts within each AS varies from several to thousands [15]. Since each team can accommodate 10 to 50 nodes, the number of teams within a group will be no more than hundreds. Thus, the size of routing table with 5000 groups and their agents' information will occupy 60KB memory, and size of delivering table with 300 teams is 3KB. Also, the network bandwidth of this overlay network is highly efficient.

Considering a overlay similar to the real world one,

System Functions	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
Dissem Trees	44	264	44	220	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

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

which has 1,000,000 nodes distributed in 5,000 ASes, 3 agents with 15 minutes each median session time in each group, average 20 nodes a team, query rate of 0.5 per node/second and 20 bytes per message, Table 2.3 breaks down consumed bandwidths of all roles and all operations, including lookup, piggyback, ring protocol and multicast within the group. The agent changing rate is 11 per second, which is calculated by the formula in [13, 6]. All groups are assumed to use dissemination trees and 5 fanouts for agents and leaders. It is obvious that our system can scale to the upper bound of real world P2P systems with little effect of hosting machines of agents and leaders.

3. Security

In this section, we will discuss in detail how to exploit the overlay structure to solve security problems faced by P2P overlay networks.

3.1. A Better Identifier

To identify each principal (node) from others is an essential issue for the security of P2P overlay networks. However, this is a really challenge under the P2P environment. Previous research [1] has pointed out that the Sybil attack and the collusion among malicious nodes are almost unavoidable, even the assignment of node IDs is delegated to a central, trusted authority. Current P2P overlay networks normally identify nodes by their IP addresses. However, due to different network configurations and connection techniques, more than 40% nodes do not have true IP addresses or change their IP addresses from time to time [19]. Although other solutions have been proposed to securely assign node IDs, such as paying money for certificate and binding node IDs to real-world identities, none of them is practical under the P2P environment.

Instead of just using mutable IP address as the identifier of each node, we propose to identify each node by its accurate network physical characteristic, called *net-print*, which is a set of information that can be used to accurately locate node's position and network device. Two reasons make it practical and efficient. As mentioned early, the nodes are organized by their AS locus and the AS is under single network administration, thus the machines within one AS are under same network configuration and policies, such as DHCP, NAT and firewall policy. The other reason is that many powerful protocols, such as ICMP, which are normally prohibited by network administrators for outside accessing due to security considerations, can be directly used, and the overhead is small since traffic is between physical nearby nodes.

Before a node joins a team, it will measure the RTTs to several routers within the AS to form its landmark vector. Instead of directly using the vector claimed by the node, a net-print vector is formed by the RTTs measured by randomly selected node within the sub-network of those routers. By comparing the two vectors, the cheating can be easily detected. Also, this method can pretend the collusion of malicious nodes from forging the net-print vector. Moreover, since all those nodes are within the same AS, the IP of node's default router can be further found by Internet Control Message Protocol (ICMP) with the IP Record Route [11]. Also, the Media Access Control (MAC) address can be used as part of net-print, and be further identified by other peers under the same sub-network. Thus, a net-print, which includes the node's default router IP, MAC and landmark vector, is an accurate and trustable identifier for each node, even under the environment that IP addresses of nodes are allowed to change dynamically.

Actually, the net-print itself is a *self-certifying data*, which can be directly verified by other nodes. For example, one node M wants to pretend to be another node A,

and even know the net-print of A, however, other nodes can easily verify M by comparing the claimed net-print and directly measured one. As a result, the collusion among malicious nodes will be almost impossible. Although a malicious node may pretend to be different nodes under the same sub-network for Sybil attack, the net-print technique can efficaciously limit it within a small network scope. By challenging those nodes with a unique computational puzzle concurrently, the malicious node can be identified.

3.2. Secure Routing

Based on the better identifier, net-print, we will discuss about other security issues of this design. The overlay network runs on a set of N nodes that form an overlay using the protocol descried in the previous section. We assume a bound f ($0<f<0.5$) on the fraction of nodes for every roles, such as agents and leaders, that may be faulty. Despite of different protocols, routing is still the core operation for structured P2P overlays. Thus, to make our routing secure is principal. Previous research [1] has pointed out the three key problems to implement secure routing primitive: securely assigning node IDs to nodes, securely maintaining the routing tables, and securely forwarding messages.

Firstly, it is for securely assign node IDs to nodes. Due to the self-certifying net-print of each node, collusion and the Sybil attack are almost impossible in our overlay network. The effect of regular malicious node is very little, since it only stores part of objects and its failure is tolerated by erasure code. Although previous research [1] has pointed out to give the P2P overlay a public key infrastructure is important for the overlay security, to assign each node a certificate is obviously not necessary in our overlay. Instead, only the agents will be granted certificate by a trusted certification authority (CA), which binds a public key to its net-print. Beside the normal net-print properties mentioned early, the net-print for agents also includes its current interior IP and exterior IP and port for routing service, since the nodes may under ISPs using DHCP and NAT. Those public keys will be known by all agents, and altered with changes of them. The Leader's certificate can be granted by any agent within the group, it includes interior, exterior IP and team ID in addition.

Secondly, we discuss how to securely maintaining the routing table. As mentioned early, the maintenance of routing tables within agents is based on the piggybacked information within the messages. In order to make this procedure secure, we combine using redundancy and electric signature techniques. The detailed procedure is following. When the agent $A1$ filtrates out an updating information from an incoming message, it will check the signature and then share the information with the other two agents by the ring protocol. Since the update information about one agent can be reported from different groups to different agents, the results will be inconsistent if some node forges it. When there is no conflict, this update will be accepted by all agents and delivered to leaders with signature of $A1$. Also, this signed update information will be piggybacked on outgoing messages by the leaders to other groups.

Thirdly, based on above discussion, the procedure to forward message securely in our system is straightforward. When a node requires its leader to route a message, it will first check the leader's certificate and then drop the message. Then, the leader will randomly choose one agent within the destination group to forward the message with its certificate. As that agent receives the message, it will check the IP and port number within the certificate to make sure it come from the expected leader. After that, it will sign the message and sent to the responsible leader. Finally, that leader will send back the result with its certificate. Normally, it is reassuring for the node to use that information. If the result is found out to be incorrect, the node can provide the message including all certificates of involved nodes to an agent, by tracking back the message, the fault node can be found out. Since the leader for the required result may itself be faulty. Replica will be reached by the key generated by the hash function initialized with another seed, which is similar with CFS [3]. Instead of just tolerating the faulty node, that node will be impeached. A node can report the suspect leaders and agents to its agent with the evidence of redundant routing results. If that node is continually impeached by different nodes, the agent will consider it faulty and keep it in a *black list*.

3.3. Dealing with Faulty Leaders and Agents

Although malicious nodes do not have much chance to interfere the overlay under the security routing, we still need to prevent the accumulation of malicious nodes for agents and leaders. Since malicious nodes may have more powerful CPU and longer session time, they may be good candidates for leaders and agents. The solution is to limit the term of every leader and agent. The maximum period of leaders and agents will be no more than 24 hours. Since a leader's certificate may just be issued by an expiring agent, the public key for each agent will be kept for another 24 hours as grace period. After that, the public key will be deleted. A previous agent may be selected to be agent again and be issued a new certificate, however, reappointment is forbidden.

In addition to limit the maximum term of leaders and agents, our overlay can actively eject the misbehaving nodes. As mentioned early, agents will keep a black list for malicious agents and leaders. If a malicious agent or leader is the black lists of two or more agents of one group, they may accuse that node jointly. They will send an appeal with their signatures to an agent in that group. Then,

the term of that node will be reduced. The more appeals received, the more quickly the node will be ejected. Finally, a new leader or agent will selected to replace the malicious one, and its certificate will be revoked. As the result, that node will be isolated by stopping sending it messages and ignoring ones from it.

4. Experiments and Results

In this section, we evaluate our approach using simulations with the real Internet AS-level topology and compare it with current P2P designs.

4.1. Experiments Setups

The AS-level topology graph used in our experiments was from CIDR [2] Report in Oct. 2003. In the graph, there are more than 15,800 ASes and 65,000 links. In order to compare with other P2P systems, we use FreePastry1.3 from Rice University and latest Chord simulator from UC Berkeley. An event-driven simulator of our overlay is developed under Java 1.4.2. For Pastry and Chord, all nodes and objects are set to have a 32bits ID. Since the default leaf set of FreePastry1.3 is 24 and the routing table size for 32 bits ID is 128, Chord simulator is also set to have 24 successors and 128 fingers. All experiments were performed in a Dell Dimension PC, which has one 2.8GHz Pentium IV processor and 1.2GB RAM, running Linux. Since our overlay is based on network topology, as fair, all that information is revealed to other two DHT designs. We modified FreePastry1.3 to build network proximity overlay based on AS path length. The Chord simulator is also adapted to support network proximity routing.

The density and distribution of hosts are important parameters in our experiments. Recent research [15] has showed that the average nodes for KaZaA and Gnutella were about 1 million and 200 thousands respectively. All nodes distributed in about 5000 ASes, and the average node density for one AS was 200 and 60 for KaZaA and Gnutella respectively. Also, 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. Since the detailed distribution of nodes in each AS is unknown, we use the Zipf distribution as approximation in our experiments. The average node density used in our experiments is 100 nodes per AS, which is between the ones of KaZaA and Gnutella. The system is modeled to have 20,000 nodes, which spread over 200 ASes under the Zipf distribution. All ASes are randomly selected in the AS-level topology graph, and topology within the AS is ignored. Every result in our experiments is the average of ten times repetitions. For the Zipf distribution, the largest AS has 2,000 nodes and the smallest one has 29 nodes. Since our routing schedule directly uses the Internet routing itself, the latency stretch is not a major consideration. Our experiments focus on the impact of various parameters of system environment, such as system churn rate, different node quest rate and size of group or team.

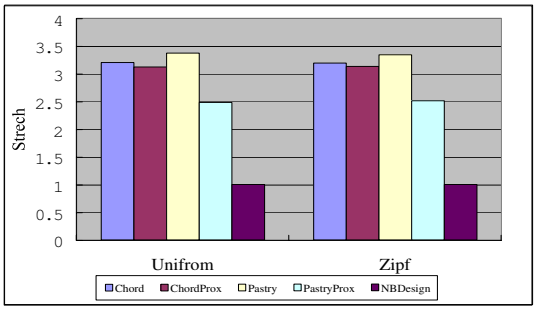

Figure 2. Stretch comparison between network-based overlay and other structured ones for 1,000,000 queries under different distributions.

4.2. Experiments and Results

In the first experiment, we compare three systems, Pastry, Chord and our network-based system, with 20,000 nodes under uniform and Zipf distribution in 200 ASs. The system overlay is built firstly, and then 1,000,000 queries are performed, no node joining or leaving during that period. Figure 2 shows the result of stretches among three systems under the different distributions and systems. Network proximity systems for Pastry and Chord perform better than original ones. Also, the distribution of nodes has little impact on systems. The performance under uniform is slightly better than Zipf ones. This is because uniform distribution of nodes gives averagely more benefit to each node than skewed one. However, the stretches of all DHT systems are still more than 2.5. On the contrary, the stretch of our network-based system is equal to one under all distributions.

In our following experiments, we test the system under highly dynamic environment. All 20,000 nodes are distributed in 200 ASes under Zipf distribution. We firstly bring up all nodes within one group to make it active, then other nodes are brought up, one every 1 second, each with a randomly assigned group agent. We then churn nodes until

the system performance levels out, this phase normally lasts about 10 minutes. Nodes' joining and leaving are timed by a Poisson process and therefore uncorrelated and bursty. In our experiments, we use churn rates from 200 per second to 10 per second, equal to median session times from 1.15 minutes to 1 hour. All nodes are chosen randomly. Although leaders and agents are considered to have longer session time than normal nodes, the algorithm used for selecting them is based on network bandwidth. Each group has three nodes with highest bandwidth as the agents. In order to examine the exact impact under churn, the statistic starts 30 minutes after churn. This is to make sure that every node has joined or left the system once and all original data structures within nodes are expired. In this set of experiments, we focus on the impact of leaving and failure of agents. The leaving of leader is ignored, because nodes within one team are physically nearby, as we mentioned before, a nearby node may replace it even before the old one's completely leaving. In each message of query and answer, the information about latest three changed agents will be appended.

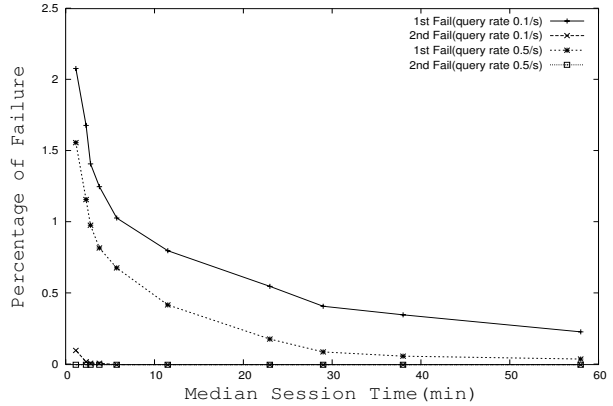

Figure 3. Network-based system under churn. Shown is the percentage of failure lookups under increasing levels of churn and different request rate. Churn rate increases to the left. Each point is the average rate during 30 minutes.

Figure 3 shows that our system has very high success rate even under extremely dynamic environment. For the query rate of 0.1 queries per node per second and median session time of 5.75 minutes, the success rate of the first query is 98.7%. The success rate for the second query is above 99.99%. The results of other P2P overlays have been reported by Rhea et al in [13]. In their experiments, within a 1000 nodes system under modest churn rate, the median session time of each node was 23 minutes, a Pastry system (FreePastry) failed to complete 70% requests. They explained the failure as that nodes waited so long on lookup requests to time out that nodes frequently left the network with several requests still in their queues. Although almost all lookups in a Chord network were completed, however, the lookup latency increased more than 20 times. Compared with their results, our overlay is significantly better in both successful rate and lookup latency. Moreover, due to the piggyback, higher workload can help update routing tables between agents. For example, under the query rate of 0.5 queries per node per second and median session time of 2.8 minutes, the first success rate is above 99%. That is to say, more frequent queries will have higher success rate than less ones. Since the leaving of any node is independent of each other, the median session time for each node in the system is also that time for leaders and agents. Thus, most of nodes will be suitable for agents, and some algorithm can be designed to make each suitable node agent. The most important thing is that most of those updates/maintenance are done during common lookup operation and do not involve any addition message.

5. Conclusions

Being aware of the difficulties faced by structure P2P overlays under an open Internet environment, from a different angle, this paper proposes a new approach to build P2P overlay network. In contrast to current designs, which focus on their own system overlays, our approach focuses on the physical network, and builds the overlay network following it. The physical network characteristics are naturally exploited to build an efficient and secure P2P overlay network. Based on node's physical network properties, the net-print provides a practical and self-certifying identifier for nodes under the open Internet environment. Combined with the straightforward overlay routing mechanism, the routing procedure is secure and can be audited. Moreover, the overlay network has the ability to repair itself by ejecting the malicious nodes.

This paper is the first step towards building large-scale peer-to-peer infrastructures based on Internet physical overlay. Many difficulties faced by current systems are smoothly solved in our design. We believe that to build system overlay following the physical network is a promotion way toward P2P and other large-scale distributed applications.

References

[1] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, Dec 2002.

[2] CIDR-Report. The CIDR Report. http://www.cidr-report.org.

[3] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stocia. Wide-area Cooperative Storage with CFS. In *Proceedings of ACM SOSP'01*, Oct. 2001.

[4] 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.

[5] J. R. Douceur. The Sybil Attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, 2002.

[6] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proceedings of ACM PODC*, July 2002.

[7] M. Network. Internet Routing Registry. http://www.irr.net.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, San Diego, CA, 2001.

[9] 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.

[10] RFC2365. Administratively Scoped IP Multicast. http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2365.html.

[11] RFC791. Internet Protocol DARPA Internet Program Protocol Specification. http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc0791.html.

[12] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. Technical Report UCB//CSD-03-1299, University of California, Berkeley, December 2003.

[13] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference*, 2004.

[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] 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.

[16] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, 2002.

[17] 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)*, pages 149–160, San Diego, CA, 2001.

[18] H. Wang, Y. Zhu, and Y. Hu. To Unify Structured and Unstructured P2P Systems. In *Proceeding of the 19th International Parallel and Distributed Processing Symposium (IPDPS05)*, Denver, Colorado, April 2005.

[19] M. Yang, Z. Zhang, X. Li, and Y. Dai. An Empirical Study of Free-Riding Behavior in the Maze P2P File-Sharing System. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'05)*, 2005.

[20] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, Apr. 2001.