To Unify Structured and Unstructured P2P Systems

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

Abstract

Most of current peer-to-peer designs build their own system overlays independent of the physical one. Nodes within unstructured systems form a random overlay, on the contrary, structured designs normally organize peers into an elegant identifier ring. However, all of those overlays are far from the physical one.

Noticed that the system overlay is crucial for building a distributed system, this paper proposes to build system overlays based on the physical overlay. By making full use of physical network characteristics and taking advantages of both structured and unstructured protocols, a network-based peer-to-peer system is built in this paper. Not only the system is highly efficient (the stretch is equal to one), but also it can adapt extremely system churning. The most important is that the maintenance overhead is very low, even under highly dynamic environment.

1 Introduction

The last few years have seen a tremendous increase in the interest and research activities of Peer-to-Peer (P2P) systems. While P2P research covers a wide spectrum of topics, such as routing/lookup, security, file systems, load-balancing, etc, one of the most fundamental research topics is how to provide efficient lookup services in a large-scale network which is completely distributed and decentralized.

Currently, P2P systems can be classified into two main categories: unstructured and structured. For unstructured P2P systems, such as Gnutella [1] and KaZaA [2], peers are organized arbitrarily. Through simple Ping/Pong mechanism, each peer in Gnutella connects with other peers randomly. An arbitrary overlay network is formed by those connections among peers. Although the protocol is simple, it can keep nodes highly connected even in event of major disasters. The search protocol uses simple flooding mechanism. When a peer receives a query message, it simply forwards the query to all neighboring peers. After several turns' forwarding, one query can reach most of peers in the network. The peers with related results will answer the query and send the results back. However, such mechanism generates a large mount of messages per query, which makes the scalability problem when the number of peers grows. Based on the proprietary Fast-track technology that uses special supernodes design, KaZaA becomes popular. Those supernodes always have higher bandwidth and connectivity, and they make an index of all nearby peers' sharing files. As a result, all queries are routed to those supernodes and processed there. Gnutella2 [3] also adopts similar technique to make system more scalable.

Unstructured P2P systems have advantages on the simple protocol, friendly key words searching and powerful ability of locating well-duplicated objects, such as music files. However, they face difficulties on scalability and locating rare objects. Structured P2P systems, such as Chord [4], CAN [5], Tapestry [6] and Pastry [7], leverage Distributed Hash Tables(DHT) to archive an administration-free, fault-tolerant overly network and guarantee to deliver a message to the destination within $O(Log(N))$ hops. In contrast to the random overlay in unstructured systems, all peers in structured systems are organized into a clear logical overlay, which is always an identifier ring. Also, each object in structured systems has been recorded in its unique place, which is opposite to unstructured ones. By a small entries DHT in each peer, a query will reach its destination within $O(Log(N))$ hops.

While elegant from a theoretical perspective, these systems have two obvious mismatches or disadvantages. The first mismatch is between the ideal logical overlay, which is a linear identifier ring, and the physical overlay, which is a power-law network. To a great extent, DHTs ignore the characteristics of the physical network. As a result, a single "hop" may be across two nodes connected via a high speed LAN, or two nodes separated by a low-bandwidth, long-latency link across half the world. The second one is between node's capabilities or resources and their responsibility. Many systems give every peer in the system equal responsibility, assuming all nodes are uniform in resources such as CPU performance and network bandwidth. However, this assumption does not hold for real systems. For example, half of nodes in a typical system are "week-nodes", with low performance, poor reliability and intermittent network connection. In addition, nodes in real P2P systems would like to join and leave the system very frequently [8, 9]. Granting those nodes the responsibility beyond their capabilities significantly impacts the performance and reliability of the system.

Not only structured systems, unstructured ones suffer from the mismatch between the system overlay and the physical one. Previous research [8] showed that 40% nodes in

Gnutella were within 10 top ASs, but only 5% communication happened within the same AS. The P2P traffic is becoming the major part of Internet traffic. Being aware of the mismatch problem, many works have been done. For unstructured systems, network proximity is used to optimize connections within a peer's neighbor and neighbors' neighbors for Gnutella by Liu, etc. [10]. For structured systems, network and geography proximity are widely used to optimize overlay construction and selection of next hop. All those works improve performance significantly, nevertheless, they have limitation because of the restriction of logical overlay.

The gap between the system overlay and the physical one becomes a serious obstacle for the growing of P2P systems. Noticed that the system overlay is crucial for a distributed system, this paper, from a different angle, proposes a new protocol for building P2P systems. Not only it makes full use of physical network characteristics, but also it combines advantages of structured and unstructured protocols and supports both of them at the same time. Opposite current P2P designs, which focus on their own system overlays, our approach focus on physical overlay, and builds system overlay based on it. By directly using Internet routing mechanism, the whole system is highly efficient, the stretch is equal to one. Also, it can adapt highly dynamic environment, and has very low maintenance overhead, because most of maintenance can be done during system common procedures.

The contributions of this paper:

1. To our best knowledge, it is our first to propose building P2P systems based on Internet physical overlay. While pervious topology-aware designs used network topology information just as the auxiliary approach, our system directly exploits network topology to construct system overlay. Instead of deploying system own routing mechanism, the Internet routing is naturally employed, which highly improves system efficiency.

2. A novel and practical approach is proposed to build system overlay closely approximating the physical network. Network topology and positioning techniques are perfectly integrated to provide accurate Internet topology.

3. Network characteristics, such as the power low, structured and unstructured protocols are incorporated together. Not only the whole system is highly efficient in looking up, but also it can adapt extremely system churning. The most important thing is that maintenance works are naturally integrated into common procedures of system. Thus, the overhead of maintenance is very low, even under highly dynamic environment.

The remainder of this paper is organized as follows. Section 2 discusses previous works to optimize P2P systems by network topology information. Section 3 provides the background of Internet topology and related techniques. Section 4 describes in detail how to build the system overlay closely approximating Internet topology. Section 5 provides the system designs to build structured and unstructured systems based on our network-based system overlay. In section 6, we evaluate our approach using simulation. Section 7 concludes the paper.

2 Related Work

To exploit physical network information to optimize P2P systems is not a new idea, current works can be classified into three main categories: proximity routing, topology-based node ID assignment and proximity neighbor selection [11].

Proximity routing is employed in Chord [4], CAN [5] and their improvements, such as [12]. With proximity routing, physical network information is not taken into account when building system overlay. However, 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 [7] and Tapestry [6]. Routing table entries are selected according to the proximity metric among all peers that satisfy the constraints of logical overlay.

In Brocade [13] and Expressway [14], 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. Routing from node *A* to *B* in those systems involves three sections. Firstly, *A* sends the message to local supernode *SA*. Then this message is routed in the second overlay to the supernode *SB* which stores the object for *B*. Finally, the message reaches node *B*. Although their benefits are limited by logical overlay restrictions, and they face the challenge of choosing appropriate supernodes and frequently updating information of connected nodes for supernodes, those systems produce significant improvements compared to original designs.

Based on BGP routing table snapshots, Krishnamurthy and Wang [15] proposed a promising technique to cluster web clients, called *network-aware clustering*. It can effectively group clients that are topologically close and under common administrative control. By this technique, they further proposed a Cluster-based Architecture for P2P(CAP) [16]. Nodes in CAP were clustered into groups by IP prefixes provided by BGP routing tables. Then those groups acted as the normal nodes in Gnutella. Although the method is simple, it significantly improves scalability and reduces the number of messages. However, CAP needed a centralized server to perform network-aware clustering, which would be a bottleneck and be vulnerable for the single failure. Furthermore, the BGP routing tables is not directly available to end-user applications. They always require privileged access to internal network information from ISPs.

3 Internet Topology and Related Techniques

3.1 Internet Topology

The Internet consists of many Autonomous Systems (ASs). An AS is a network under a single administration authority using a common Interior Gateway Protocol (IGP) for packet routing. Computers within an AS are always geographically close and connected by a high-speed network. Some border routers running Border Gateway Protocol (BGP) [17] connect the AS with its neighboring ones to form the Internet.

A good structure for system overlays should be clear, stable and easy to acquire. The topology of the Internet has three levels, IP-level, router-level and Autonomous System (AS)-level [18]. The granularity of IP-level topology is too small to reflect Internet overlay structure. Also, it is almost impossible to obtain such kind of topology for the sheer number of IP addresses and a large number of frequently joining and leaving machines at any given moment. The granularity of router-level topology is the best to reflect Internet structure. However, to acquire such kind of information needs either the privileged access to BGP border routers or huge overhead of probing. Both of them are not a general way to applicant end users. At AS-level, Internet has an even cleaner topology since the Internet is made up of those ASs, and many public services provide that information, such as CIDR Report [19] and IRR [20]. Although not as detailed as BGP routing tables, that information is publicly published and easily obtained. The only drawback is that the granularity of AS-level topology is too coarse to provide inside structure of the network especially for large ASs.

Although router-level topology provides a better granularity than AS-level, it is hard to be obtained. Also, the distribution of the nodes in P2P systems is far from uniform [21, 9]. The average nodes within one AS are 224 and 56 for KaZaA and Gnutella respectively. Thus, in many cases, such detailed Internet topology is overkill. Our researches show that AS-level topology really provides a good frame for building system overlay. Within the AS, a light topology-aware technique can be used to provide further details.

3.2 Topology Aware Techniques

There are many methods and techniques to acquire network topology information, such as BGP routing tables [22, 23, 19, 20], widely used landmark techniques [24, 25] and the network of physical springs [12]. In terms of provided information, those techniques can be classified into two categories: real world network topology and proximate network positioning.

As we mention before, the whole Internet is formed by thousands of ASs with border routers running BGP, which connect one AS with its physical neighbors. Two ASs are called connected when there is a direct BGP link between them. By exchanging information with BGP peers, each BGP router forms a detail routing table with AS paths to every existing AS. As a result, BGP routing tables can provide real world Internet topology. ASs or routers and their connections can be retrieved from those valuable tables. The advantages of BGP tables are obvious. However, they have their disadvantages. Firstly, those BGP routing tables are not easily obtained. Secondly, because BGP tables include lots of information for routing selection, those tables are always too large and complex to directly use. Fortunately, public services, such as the CIDR Report [19] and WHOIS service from IRR [20], announce that information at AS-level. All information about one AS, such as AS number, IP address range and AS connectivity, can be listed in details. The CIDR Report even updates that information daily and provides thorough analyses.

Landmark techniques and the Vivaldi belong to network positioning. They can provide some kind of network coordinates to reflect each node's relative network position or distance in the Internet. Different from network topology techniques, which provide actual structure information based on real routing tables, network positioning ones always use RTTs between one node and other famous or pre-selected hosts, called landmarks, to calculate network coordinates, which can estimate related network distance between hosts. Since many unpredictable factors, such as changes of routing, network bandwidth and traffic, will affect the RTT between nodes, those techniques are not as accurate and stable as network topology ones, and do not directly reflect network structure. However, network positioning techniques are distributed and self-dependent, which are especially suitable for P2P systems.

4 Building a System Overlay based on Internet Topology

Previous works make us believe that to make system overlay closely approximating physical overlay is the right direction to build large-scale P2P systems. Compared with current designs, it at least has two major advantages. Firstly, under consistency between system overlay and physical one, system operations will be more efficient. Secondly, underlying network mechanisms, such as routing, can be directly used to simplify system design. In this section, we will show how to build a system overlay closely matching the physical one by Internet topology information.

Based on published Internet AS-level topology information, all nodes are divided into groups by their AS residence. Like the AS in the Internet, the group is the basic unit for routing and organizing nodes in the system. To partition nodes only by their AS residences is too coarse in many cases, especially in large ASs with lots of nodes. The network-aware clustering technique from Krishnamurthy and Wang is a sufficient method to sub-divide nodes in large ASs. However, not only such detailed informant is hardly obtained, but also it is overkill, since nodes may sparsely distribute among routers in the AS. Thus, we propose to use landmark technique to further divide nodes into teams within an AS. Landmarks can be prejoined nodes or their default routers. Because all probes are within the same AS, no traffic across Internet backbone, the overhead for network coordinates will be little. Also all those packets will not be denied as intrusion since they are under the same administration. By landmark vectors technique proposed by Ratnasamy, etc. [25], physical nearby nodes can be easily clustered together to form a team.

The connections between groups will be also agree with the Internet. We cannot directly use ASs connectivity for groups, since not every AS is a group in P2P environment. Also, previous research [26] showed that the distance information provided by the AS-level topology could be too coarse as it was only specified in AS-level hops. Though AS path lengths are really too coarse to estimate most of distances, which is always four or more AS hops, they are accurate enough to predict short distance [26], which is one or two AS hops. Thus, we define groups within two ASs path length are physical neighbors. Some nodes with high bandwidth and availability will be elected within each group as agents to exchange information with neighboring ones. Although each peer arbitrarily joins and leaves the system, previous researches [8, 9] showed that their behaviors were highly skew. 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 about 90% of the total traffic. Thus, about 10% nodes have high availability are decent bandwidth, which are suitable candidates for agents. Our experiments also show nodes with normal session time, about 30 minutes, will be enough for agents.

Within a team, a leader will be elected to serve teammates and connect with agents. In small groups, which have tens of teams, leaders will directly link to nearby agents. When a group is very large, which has hundreds of teams, *dissemination trees* with agents as the root will be used to efficiently transfer messages between agents and leaders. Based on each team's network coordinates, the agent can easily build that dissemination tree. A two level tree is enough to support hundreds of teams. In order to keep teams and nodes highly connected, unstructured-like epidemic (or gossip) protocol is used simultaneously. Physical nearby teams will make neighbors each other. Nodes within one team will randomly select some others as their neighbors. Those arbitrary connections are important to keep nodes highly connected even in the event of major disasters. Also, they can help the system to quickly rebuild the organization within groups.

4.1 Hub Groups

In our research, we find an interesting characteristic of the Internet. Due to the power law of the Internet, there are some hub-like ASs that always connect with hundreds of ASs. Those hub-like ASs are also highly connected with each other, and average AS path length between them is only 1.47. Moreover, every AS in the Internet will at least encounter one of such hub-like ASs within the AS path length of four. By those hub-like ASs, called *hub AS*, all ASs of the Internet can be further divided into *areas*, which include a hub AS and all nearby ASs. This characteristic can be further used for organizing groups, and efficient communication, because all groups within one area naturally form a two level dissemination tree as we define groups within two ASs path length are physical neighbors. The hub group connects nearby groups within 2 AS path length, and those groups connect remote groups. If no nodes in the hub AS, the physical nearest group will become the hub group.

4.2 Keep System Overlay Up-to-date

As we all known, the Internet keeps evolving. Although previous studies showed that AS level topology of the Internet was quite stable [27] and stale information world not affect the correctness of system routing, some mechanism was needed to update the system overlay. It is obvious not scalable to update information from a centralized server, such as ICDR report or IRR record. A novel scheme is developed in our system to update topology information without overloading the centralized server. Instead of storing a whole Internet topology graph within each group's agents, only the related part, including only the ASs and links within 2 ASs path length from current group, will be recorded. Moreover, previous research [26] pointed out that previous measured RTTs were the best way to detect network changes. Instead of periodically update topology information, our system only updates information in need. When agents exchanging information with their neighbors, RTTs are also recorded. If the agent observes major and persistent changes of the RTT, the nearby network is considered changed. Instead of retrieving the whole topology graph from the centralized server group by group, only the nearest hub group collects this graph, which is no than 300KB, from the server daily and delivers to other hub groups during information exchanges. Thus, when the nearby network changes, the group will ask its nearest hub group for related topology information.

4.3 Routing and Maintenance

Another advantage of building a system following a physical overlay is that the system can make use or even directly use the routing mechanism of the physical overlay. Instead of building a whole new routing mechanism, our system directly uses the Internet routing. The routing table for the agents is a list of group numbers and related IP addresses of their agents. Currently all nodes in P2P systems are distributed in 4 to 5.5 thousands ASs [9]. The table size is about 60KB for three agents each group. An old PC (with one 800MHz Pentium-III CPU and 128M memory) can perform 24,000 sequential lookup operations per second on this table. The routing procedure is much simplified. Given a destination group number, the agent checks its list and selects one agent in that group to drop the message. Like BGP routers in the Internet, agents will exchange their tables with neighboring ones. Thus, all groups will know each other. The exchanging of information dose not need to be frequent, a period of 10 minutes is considered enough. Although agents are considered to have higher availability than normal nodes, they are not well-maintained routers. They will fail or arbitrarily leave the system. To update agent information just depending on periodically exchanging information between nearby groups is not efficient enough. Some mechanism is needed to fast up this kind of updating.

The piggyback technique is used here. When agents send or answer queries, latest information about agents, which is 12 bytes for three agents, will be appended to the messages. In addition, the latest agents information will be periodically reported to nearby hub group, and within 30 minutes all groups will know that. In fact, the piggyback technique is very efficient and can perform most of those updates even under highly dynamic environment. Our experiments show that in a system with 20,000 nodes in 200 ASs, assuming a median session time of 5.5 minutes for each node, the piggyback technique can guarantee that the success rate of first attempt is above 99%. The most important thing is it dose not involve any addition message. Thus, the whole system has very low maintenance overhead or is even *maintenance-exempted*. Moreover, the piggyback technique actually provides the ability of *hot swap* for the system, which can highly improve the availability and reliability of the system under drastic environments.

4.4 Node Joining and Leaving

When a node joins the system, its AS number can be determined by its IP address through either WHOIS service or the agents in hub groups, which update that information daily. Then the joining request is forwarded to the agent of that AS. Normally the node will join one team according to its network locality. The overhead of node joining is almost minimal since only the leader has to update some book-keeping information.

If a team is too populous, it will split into two teams based on network locality automatically. If the joining node is the first node in that AS, the requirement will be forwarded to an agent in hub group, which will find out the nearest group from that AS to forward the request. Instead of forming a new group, the node will initially become a teammate within the nearest 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. The information of the new group will appear in the routing table of the mother group and be reported to the nearby hub group. Within 30 minutes, all groups will know it.

For the normal node, its leaving or failure is automatically tolerated by the team. If many nodes leave a team, the team may disappear. The remainder will join a physical nearby team. The state will be similar when a group is disappearing.

All above are a framework for the system overlay, cooperating with addition data structures, it can form different purposes distributed systems. In the following section, we will show how it can be enhanced to support structured and unstructured P2P systems.

5 System Design for Structured and Unstructured P2P Systems

5.1 Structured P2P Systems

In order to support structured DHT designs, similar ID mechanism is added to our system. Like current DHT designs, each object in our system has a 128-bit ID, which can be generated by a basic hash function such as SHA-1 [28]. Instead of mapping a small range of objects to each node in current DHT designs, a two levels mapping mechanism is used in our system. As mentioned before, our system is built up with AS-like groups. The first level is among those groups. The second level is among teams. Since valid Internet AS number is from 1 to 64511 and there are only about 17,000 active ASs currently, we believe that a 32bits ID for groups is enough. The number of nodes within one team is around 10 for consideration of performance, stability and asymmetric throughput of major network connection, such as cable modem and DSL. Each team will elect a leader with decent network bandwidth and availability. Also, a 32bits team ID is considered enough. Each group and teams within it will be assigned a unique ID by random when they joins the system. The first 64bits of the object ID is separated into two parts. The first 32bits is for group ID, and the second 32bits is for team ID. Each group charges the range of ID from its own to the one before next group ID, and so is for teams.

The team is the basic unit to store objects. Two copies of objects will be kept within one team. Because the object is always the location information of a published file, 100,000 objects will occupy no more than 3MB hard disk, which is obvious not a burden for current PCs. One copy is kept in the leader to answer queries for all other peers. The other one will be divided into blocks with erasure code technique to store among teammates. Previous research [29] pointed out erasure code technique can significantly improve availability and reliability of objects under P2P environment. The overhead to transfer and store those objects between physical nearby nodes is minimal and especially suitable for asymmetric network of DSL and cable modems. When the leader is suddenly dead, blocks from different nodes will quickly rebuild a whole copy for the new leader within seconds. Every teammate will periodically report to the leader. As nodes frequently joining and leaving, some teams may become uninhabited. When the leader finds that the rest blocks among teammates are less than the number for rebuilding a copy, it will connect with nearby teams to borrow some nodes to store those blocks. This is called *flexing* among teams. Such kind of flexing mechanism can not only significantly absorb the impact of frequently joining and leaving, but also can make the whole system gracefully degrade instead of churning. For example, a team may disappear under continually leaving of nodes, however, before the leader or the last teammate leave the system, at least one copy of stored objects will be available in the nearby team under the flexing mechanism. The state will be similar when a group is disappearing.

Normally, there are 3 to 5 agents within one group. In addition to routing table for all other groups, they also keep a table for all teams within it. All those agents work as backup of each other, route queries for other groups and update routing tables. Leaders will cache the latest routing table from nearby agent and provide routing services for their teammates. The piggyback technique is also can be used to update routing tables for leaders. When agents forward messages to responsible leaders, the latest update of routing table will be appended.

5.2 Unstructured P2P Systems

For unstructured P2P systems, we directly use the system overlay to organize all nodes within the system. The leader of each team will collect information of shared files among teammates, and form a Query Hash Table (QHT), which is used in Gnutella2 for filtering queries. A QHT is a table of 2^N bits, where each bit represents a unique word-hash values. Normally the N is 20, which has 1048576 possible word hash values. When a searchable plain-text word is contained in node's content, the related bit is marked. QHTs provide enough information to know with certainty that a particular node will not be able to provide any matching objects for a given query. Also, it is very efficient both in terms of exchange and maintenance and lookup cost.

Instead of blind flooding, search is done by *expanding ring* model. When a node issues a query message. The query will be firstly resolved by the leader, then following dissemination trees, the query will reach the agents, which keep QHTs of all teams. If the query still cannot be resolved, agent will forward that query to nearby groups and then all groups within the same area. If the query still cannot be solved, it will be forwarded to groups within other areas, and finally reach the all groups. Since this is a framework, the latest technique for unstructured P2P systems mentioned by Chawathe etc. [30], such as topology adaptation, flow control, one-hop replication and random walk search, can be easily used or even get more benefits from our network-based overlay. Of course, further researches are needed.

6 Experiments and Results

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

6.1 Experiments Setups

The AS-level topology graph used in our experiments was from CIDR [19] Report in Oct. 2003. We also downloaded all BGP tables of that time from Oregon Routeviews [22]. By inferring from those raw BGP dumping tables, we obtained our own AS-level topology file. Those two topology files are perfectly matched. In the graph, there are 15,800 ASs 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 system is developed under Java 1.4.2. For Pastry and Chord, all nodes and objects are set to have a 32bits ID, because of data structure limitation of the Chord simulator. The default leaf set of FreePastry1.3 is 24, so simulator of Chord is also set to have 24 successors and 128 fingers, which is the number of entries for 32bits ID in Pastry. All experiments were performed in a Dell PC, which had one 2.8GHz Pentium IV processor and 1.2GB RAM, running Linux.

Since our system is based on network topology, as fair, all that information is also revealed to DHT designs. We modified FreePastry1.3 to build network proximity overlay based on AS path length. We also adapted the Chord simulator to support network proximity routing. In each hop of routing, a node chooses the "best" node from three candidates to forward the query. Our experiments show that to easily choose the nearest node as the next hop will lead worse performance in both average hops and stretch. Although the network distance for next hop maybe reduced, the overall hops for the query is increased and the whole distance rises. A balance point is needed to make benefit from network topology information. As many times of experiments, we found out such point in our environment, though it may not be the best one.

The density and distribution of hosts are important parameters in our experiments. Recent research [9] showed that the average nodes for KaZaA and Gnutella were about 1 million and 200 thousands respectively. All nodes distributed in about 5000 ASs, and the average nodes within one AS was 200 and 60 for KaZaA and Gnutella respectively. Although the host density, connectivity and traffic volume of P2P systems cannot be well modeled by Zipf's distribution, they were highly skewed and exhibited heavy tails. Since the detailed distribution of nodes in each AS is not known, we will use Zipf distribution instead in most of 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 ASs under Zipf distribution. All ASs 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 Zipf distribution, the largest AS has 2,000 nodes and the smallest one has 29 nodes.

6.2 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. Finger 1 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 takes little impact to systems. The performance under uniform is very slightly better

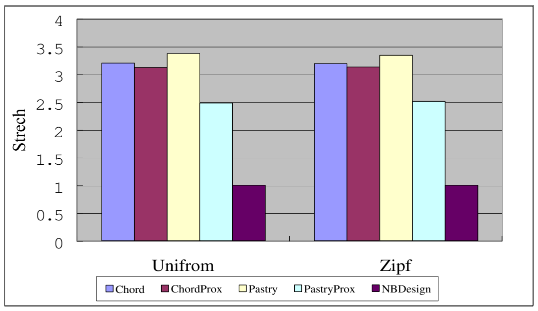


Figure 1: Stretch comparison between Network-based system and structured P2P systems. Shown is the average stretch for 1,000,000 queries under different systems and distributions.

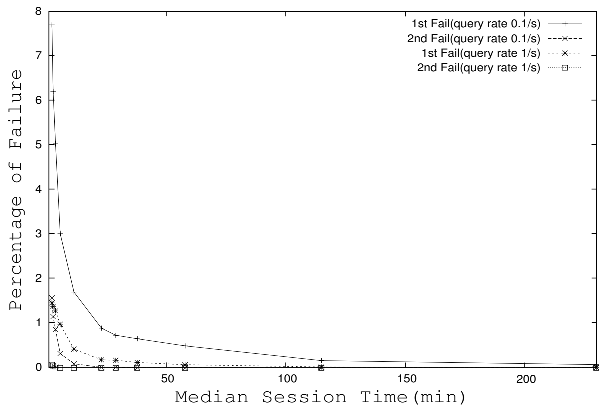


Figure 2: Network-based System under churn. Shown is the percentage of failure lookups under increasing levels of churn and different request rate. Churn increase to the left. Each point is the average rate during 30 minutes.

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 the following experiments, we test our system under highly dynamic environment. All 20,000 nodes are distributed in 200 ASs under Zipf distribution. We firstly bring up all nodes within one group to make it active, then other nodes are brought up, one every 2 second, each with a randomly assigned hub group agent. We then churn nodes until the system performance levels out, this phase normally lasts about 20 30 minutes since groups will exchange information every 10 minutes, and 3 hops is enough to reach every group. Nodes' joining and leaving are timed by a Poisson process and therefore uncorrelated and bursty. Previous research [31, 32] pointed out, in a Poisson process, an event rate λ corresponded 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

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

In our experiments, we use churn rates from 100/second to 1/second, equal to median session times from 2.3 minutes to 3.8 hours. All nodes are chosen randomly, though leaders and agents are consider to have longer session time than normal nodes. In this set of experiment, we focus on the impact of leaving of agents. The effect of leaving of leader is ignored, because nodes within one team are physically nearby, before the leader finally leaving that team, a nearby node can replace it in seconds. The piggyback technique is used to fresh routing tables for remote agents.

Finger 2 shows our system has very high success rate even under extremely dynamic environment. For the query rate of one query per node per second and median session time of 5.5 minutes, the first query fail rate is less than 1%. Because of the piggyback technique, higher overload can help update routing tables between agents. That is to say more frequent queries will have higher success rate than less ones. Under modest churn rate, the median session time of 23 minutes, and lower request rate, the first query success rate is also above 99%. In another words, most of nodes with decent session time will be suitable for agents. The most important thing is that all those updates/maintenance are done during common procedures and do not involve any addition message. In contrast to current structured P2P designs, which always suffer from maintenance overhead, our system is *maintenance-exempted*, even under highly dynamic environment.

Node capacity	Percentage of nodes
5	6%
50	29%
500	48%
5000	7%

Table 1: Distribution of Node Capacity

Since all requests must go to the agents before finally reaching their destination team, those agents may become the bottleneck of the whole system. In following experiments, we will examine the capacity of handling queries of the system. To capture the effect of query load on the system, our simulator imposes capacity constraints on each node within the system. Every node is modeled to have a capacity *C*, which represents the number of messages that it can issue or process per unit time. If a node receives queries at a rate higher than its capacity, it is considered overloaded and messages will be discarded. We assign capacities to nodes based on the distribution that is derived from the measured bandwidth distribution for Gnutella as reported by Saroiu et al. [8]. Our capacity distribution has four levels as shown in Table 1. This distribution reflects reality that Gnutella clients are made up of fair dial-up connections, major cable-modem or DSL and small part of high-speed connections. In order to explore the highest throughput of the whole system, all normal nodes and leaders are assigned unlimited capacity, and the each group has 3 nodes with highest capacity as its agents. The whole system is made up of 20,000 nodes that are distributed within 200 ASs as Zipf distribution. Each query is issued from the random node to the random destination.

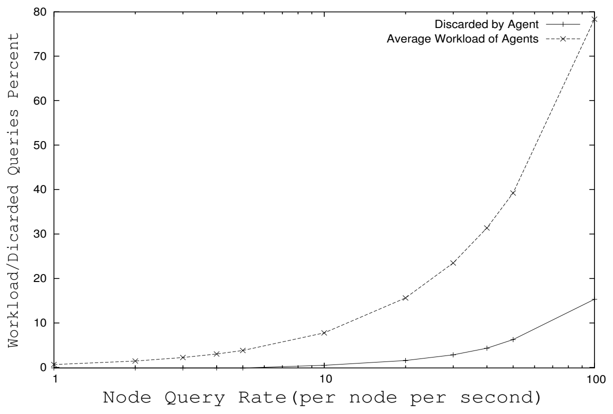


Figure 3: Throughput of the network-based system. Shown is the percentage of queries discarded by overflowed agents and average workload of the agents under different query rates.

Finger 3 shows the percentage of queries discarded by overloaded agents and average workload of all agents under different query rates. Under query rate of 10 per node per second, queries discarded by agents are no more than 1%. Although the average workload of the agents is no more than 80% under query rate of 100 per node per second, more than 15% queries are discarded. The main reason for this is the highly skewed distribution of node resources and randomly assigned group IDs. The agents within the small group and charging a large ID range will be easily overloaded. This may imply that some kind of balance algorithm is needed to assign responsibility, such as charged ID range, to each group according their capacities. Since the information of each group can be reflected in the routing table within every agent, suck kind of balance algorithm is practical. Of course, further researches are needed. In fact, in the real system, the average query rate of 10 per node per second is very bursty. The network traffic volume for just query messages at that rate of a 20,000 nodes system is about 350GB per day, which is more than the total volume, about 300GB per day [9], of the Gnutella with more than 300,000 nodes in the real world. In addition, the capacity of the system can be easily improved by increasing agents or *Hop Swapping* overloaded ones. That is, when an agent is being overloaded, a new agent will be selected and broadcast to other groups by the piggyback technique.

In the real system, the team size and the capacity of leaders are also crucial to the system. By imposing capacity constraints for normal nodes and leaders, we repeat above experiments under different team size. As the left one in finger 4 shows, discard rates of leaders for smaller team is much less than the ones for bigger team, for the leader is easily overflowed by queries from lots of nodes. On the contrary, discard rates of agents for smaller team are a little higher than the ones for bigger team. This is because a fair fraction of queries are discarded by overloaded leaders and cannot reach the agents. In another words, the leader of smaller team can service more queries and improve the throughput of the system. However, larger team can significantly improve the stability of the system. We measure the percentage of defect teams, which have less than half of default number of nodes, under different node failure rates. As the right one in finger 4 shows, as nodes failure rate increasing, much more smaller teams become defect than bigger ones. When 50% nodes failure, about 18% small teams, which have average 5 nodes, become defect, but only affect about 4% big teams, which have average 15 nodes. The size of team is a tradeoff between the performance and the stability of the system.

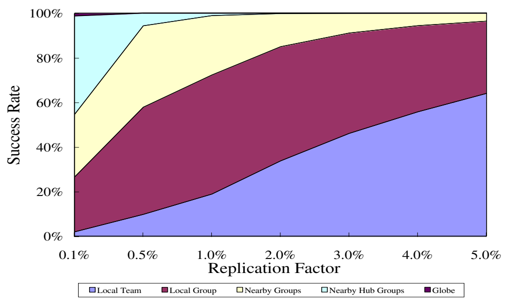
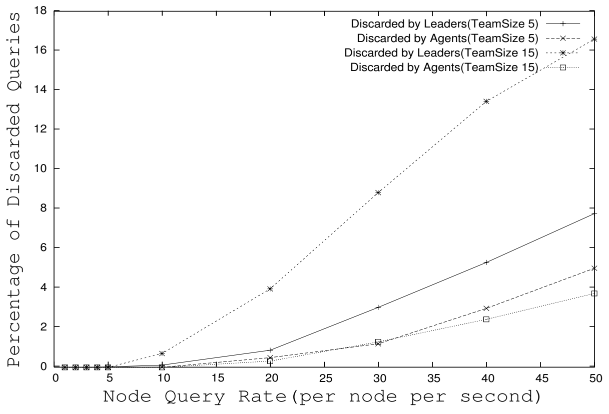


Figure 5: Keywords search for unstructured-like P2P systems. Shown is the search success rate of different search range under various replication factors. The system has 20,000 nodes, which Zipf distributed in 200 ASs, and the average team size is 15.

At last, we give the result of our expanding search method for unstructured-like P2P system. Queries are modeled as searching for specific keywords. Each keyword maps on to a set of files. All files associated with specific keywords are potential answers for the query of that keyword. *Replication factor* is used to refer to the fraction of nodes at which answers to queries reside. A query will be firstly resolved by the leader, then following dissemination trees, the query will reach the agents, which keep QHTs for each team. If the query still cannot be resolved, agent will forward that query to all nearby groups and other groups within the same area. And then the query will be sent to nearby hub groups, and finally reach the all groups. Finger 5 shows the success rate of different range of search under various replication factor. With the replication factor increasing, more queries can be solved within the



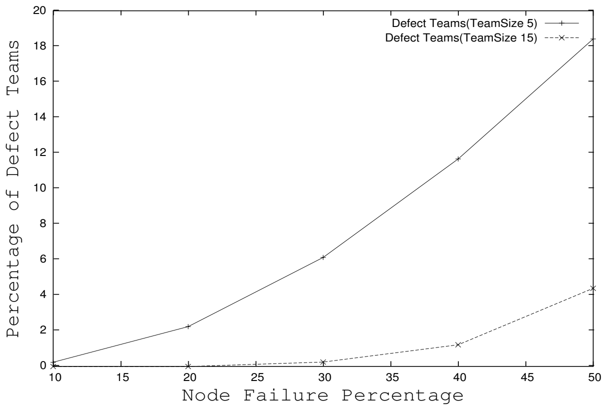


Figure 4: Impact of team size to the system. Shown on left is the percentage of queries discarded by overflowed agents and leaders under different query rates. Shown on right is percentage of defect teams under different node failure rates.

range of nearby groups or even local group. With a not high replication factor of 0.5%, 48% queries are solved within the group, and no more than 5% queries will go out to nearby hub groups. Though our experiments are simple, we believe that, for locating well-duplicated objects, our network-based design will be more efficient for less traffic and lower latency compared with current unstructured P2P systems.

7 Conclusions

From a different angle, this paper proposes a new approach to build P2P systems. In contrast to current P2P system designs, which focus on their own system overlays, our approach focus on physical overlay, and build system overlay following it. By making full use of physical network properties and characteristics of both structured and unstructured protocols, our network based P2P system has the advantages of both of them. Not only the whole system is highly efficient in looking up, the stretch of which equals to one, but also it can adapt extremely system churning. The most important thing is that maintenance work is naturally integrated into common procedures of system. Thus, the overhead of maintenance is very low, even under highly dynamic environment.

This paper is the first step towards building large-scale P2P infrastructures based on Internet physical overlay. Many difficulties faced by current P2P systems, such as scalability, searching overhead of unstructured ones, efficiency and maintenance overhead, are smoothly solved in our design. We believe that to build system overlay following physical one is a promotion way toward peer-to-peer, large-scale distributed applications.

References

[1] Gnutella, "Gnutella hosts." http://www.gnutellahosts.com.

[2] KaZaA, "KaZaA Media Desktop." http://www.kazaa.com.

[3] Gnutella2, "Gnutella2 Developer' Network." http://www.gnutella2.com.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems," in *Proceedings of the 23st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-04)*, 2004.

[11] 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.

[12] 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.

[13] 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.

[14] Z. Xu, M. Mahalingam, and M. Karlsson, "Turning Heterogeneity into an Advantage in Overlay Routing," in *Proceedings of the 22st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-03)*, 2003.

[15] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," in *Proceedings of the 2000 conference on applications, technologies, architectures, and protocols for computer communications SIGCOMM*, pp. 97–110, 2000.

[16] B. Krishnamurthy, J. Wang, and Y. Xie, "Early measurements of a cluster-based architecture for P2P systems," in *ACM SIGCOMM Internet Measurement Workshop (San Francisco, Nov. 2001)*, (San Francisco, CA), 2001.

[17] RFC1771, "A Border Gateway Protocol 4 (BGP-4)." http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1771.html.

[18] H. Chang, S. Jamin, and W. Willinger, "Inferring AS-level Internet topology from router-level path traces," in *Proceeding of SPIE ITCom 2001*, (Denver, CO), August 2001.

[19] CIDR-Report, "The CIDR Report." http://www.cidr-report.org.

[20] M. Network, "Internet Routing Registry." http://www.irr.net.

[21] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the Gnutella network: Properties of large-scale Peer-to-Peer systems and implications for system design," *IEEE Internet Computing Journal*, vol. 6, no. 1, 2002.

[22] Routeviews.org, "Route Views Archive." http://www.routeviews.org.

[23] Traceroute.org, "Public Route Server List." http://www.traceroute.org.

[24] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," in *Proceeding of the 23nd Internatinal Conference on Distributed Computing System(ICDCS03)*, 2003.

[25] 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.

[26] B. Huffaker, M. Fomenkov, D. J. Plummer, D. Moore, and k claffy, "Distance Metrics in the Internet," in *IEEE International Telecommunications Symposium(ITS) 2002*, 2002.

[27] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "The Origin of Power-laws in Internet Topologies Revisited," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, vol. 2, (Piscataway, NJ), pp. 608–617, June 23–27 2002.

[28] F. 180-1, "Secure Hash Standard." U.S. Department of Commerce/NIST, National Technical Information Service, Apr. 1995.

[29] H. Weatherspoon and J. D. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

[30] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," in *Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2003.

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

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