PAIS: A Proximity-aware Interest-clustered P2P File Sharing System

Haiying Shen
Department of Computer Science and Computer Engineering
University of Arkansas, Fayetteville, AR 72701
hshen@uark.edu

Abstract

Efficient file query is important to the overall performance of Peer-to-Peer (P2P) file sharing systems. Clustering peers by their common interests can significantly enhance the efficiency of file query. On the other hand, clustering peers by their physical proximity can also improve file query performance. Few current works are able to cluster peers based on both peer interest and physical proximity. It is even harder to realize it in structured P2Ps due to their strictly defined topologies, although they provide higher file query efficiency than unstructured P2Ps. In this paper, we introduce a proximity-aware and interest-clustered P2P file sharing system (PAIS) based on a structured P2P. It groups peers based on both interest and proximity. PAIS supports sophisticated routing and clustering strategies based on a hierarchical topology. Theoretical analysis and simulation results demonstrate that PAIS dramatically reduces the overhead and enhances efficiency in file sharing.

1 Introduction

A peer-to-Peer (P2P) system is a distributed system without any centralized control or hierarchical organization, in which each node has equal functionality. Over the past years, the immense popularity of the Internet has produced a significant stimulus to P2P file sharing systems. It was reported [1] that the total global P2P simultaneous users reached 9,670,552 in December 2005, and it represents an increase of over 1.28 million users, or 13.28%, since January 2005. For example, the KaZaA [2] application has had upwards of 20 million downloads and it can have anywhere up to 800,000 users connected at one time.

There are two classes of P2P systems: unstructured and structured. Unstructured P2P networks such as Gnutella [3] and Freenet [4] do not assign responsibility for data to specific nodes. Nodes join and leave the network according to some loose rules. Currently, their file query method is based on either flooding [3] where the query is propagated to all neighbors, or random-walkers [5] where the query is forwarded to randomly chosen neighbors until the file is found. Flooding-based search mechanism brings about heavy traffic in a large-scale system because of exponential increase in messages generated per query. Though random-walkers reduce flooding by some extent, they still create heavy overhead to the network due to the many requesting peers involved. Furthermore, flooding and random walkers cannot guarantee data location. They do not ensure that querying terminates once the file is located, and cannot prevent one node from receiving the same query multiple times, thus wasting bandwidth. Structured P2P network [6–9], i.e. Distributed Hash Tables (DHTs), can overcome the drawbacks with their features of higher efficiency, scalability and deterministic data location. They have strictly controlled topologies and their data placement and lookup algorithms are precisely defined based on a DHT data structure and consistent hashing function [10]. The node responsible for a key can always be found even if the system is in a continuous state of change. Most of the DHTs require $O(\log n)$ hops per lookup request with $O(\log n)$ neighbors per node, where n is the number of nodes in the system.

A key criterion to judge a P2P file sharing system is file location efficiency. To improve the efficiency, numerous methods have been proposed. Recently, a new wave of P2P systems is advancing an architecture of centralized topology embedded in decentralized systems; such a topology forms a super-peer network [11–14]. A super-peer topology consists of supernodes with fast connections and regular nodes with slower connections. A supernode connects with other supernodes and some regular nodes at the same time, and a regular node connects with a supernode. In a super-peer topology, the nodes at the center of the network are faster and therefore produce a more reliable and stable backbone. This allows more messages to be routed than if the backbone was slower and therefore allows greater scalability. Super-peer networks occupy the middle-ground between centralized and entirely symmetric P2P networks and have the potential to combine the efficiency of a centralized search with the efficiency provided by distributed search.

Another class of methods to improve file location efficiency is proximity-aware structure [15–17]. Recall that

P2P overlay network is a logical structure constructed upon a physical network. That is, logical proximity abstraction derived from a P2P doesn't necessarily match the physical proximity information in reality. The shortest path according to the routing protocol (i.e. the least hop count routing) is not necessarily the shortest physical path. This mismatch becomes a big obstacle for the deployment and performance optimization of P2P file sharing systems. A P2P system should utilize proximity information to reduce file query overhead and improve its efficiency. Proximity-aware clustering to group physically close peers is an effective technique to improve the efficiency. The third class of methods to improve file location efficiency is to cluster nodes based on their interests [18–27]. They lead to clusters of peers with similar interests, and in turn allows to limit the latency of searches required to find files.

Although numerous proximity-based or interest-based super-peer topologies have been proposed with different features, few methods are able to cluster peers according to both proximity and interest. In addition, most of these methods are on unstructured P2P systems that have no strict policy for topology construction. They cannot be applied to general DHTs directly though DHTs provide higher file location efficiency. The problems addressed in the paper include whether the interest-based and proximity-aware clustering methods can be integrated with super-peer topology, and whether the methods can be applied to structured P2P for high performance.

This paper presents a proximity-aware and interest-clustered P2P file sharing system (PAIS) on a structured P2P. It groups peers not only with the same interests but also in close proximity. It also places files semantically together, and organize them in a fashion similar to a Yellow Pages. More importantly, it keeps all advantages of DHTs over unstructured P2Ps. Relying on DHT lookup policy rather than broadcasting, the PAIS construction consumes much less cost in mapping nodes to clusters and mapping clusters to semantic descriptions.

The remainder of this paper is structured as follows. Section 2 presents a concise review of representative approaches for file location efficiency improvement in P2P systems. Section 3 describes PAIS, focusing on its structure and algorithms. Section 4 analyzes the performance of PAIS in both static and dynamic environments. Section 5 provides conclusion of the paper.

2 Related Work

In the past few years, numerous methods have been proposed to improve the performance of P2P systems. One method is super-peer topology that introduce hierarchy into the network in the form of supernodes, peers which have extra capabilities and duties in the network. FastTrack [13] is super-peer topology. Its routing is almost the same as on Gnutella, but broadcasting is between the supernodes only and a node sends its query to its supernode only. Every supernode searches an index that contains all the files of its connected nodes. Two file sharing applications, KaZaA [2] and Morpheus [28] are based on the FastTrack protocol. It is also proposed to apply the hierarchical designs of FastTrack to the Gnutella network [14]. Yang *et al.* [29] pointed out a potential drawback of super-peer network; that is, when a supernode fails or simply leaves, all its clients (the nodes in the same cluster with that supernode) become temporarily disconnected until they find a new supernode to connect to. To address this problem, they proposed method of super-peer redundancy. Instead of a single supernode in a cluster, there are some redundant supernode partners with the same responsibility in a round-robin manner. To reduce the individual query load, each partner is connected with half of the clients.

The super-peer network in [11] is for efficient and scalable file consistency maintenance in structured P2P systems. A file update message is propagated among super peers, and the super peers further forward the message to their own children. Our previous work built a super-peer network for load balancing [15]. A supernode and a number of physically close regular nodes constitute a group. In each group, regular nodes report the information of their file status to their supernode, which arranges the file replication between the regular nodes. The fast connection of supernodes lessens the workload produced by a large amount of messages. In addition, super-peer network uses the node heterogeneity to its advantage as the nodes with limited capabilities are shielded from query processing and traffic.

Another class of methods to improve file location efficiency is to take into account the physical structure of the underlying network during file locations. Techniques to exploit topology information in overlay routing include geographic layout, proximity routing and proximity-neighbor selection. *Geographic layout method* maps the overlay's logical ID space to the physical network so that neighboring nodes in the ID space are also close in the physical network. It is employed in topologically-aware CAN [16]. *In proximity routing method*, the logical overlay is constructed without considering the underlying physical topology. In a routing, the node with the closest physical distance to the object key is chosen among the next hop candidates in the routing table. The entries of a routing table are selected based on proximity metric among all nodes that satisfy the constraint of the logical overlay (e.g., in Pastry [7], the constraint is the nodeID prefix). This method has been adopted to Chord [6] and CAN [30]. Its benefits are dependent on the determination of an appropriate number of the hop candidates. *Proximity neighbor selection* is a middle-ground approach between the above methods. It selects the routing table entries pointing to the topologically nearest among all nodes with node ID in the desired portion of the ID space. Castro *et al.* [31] implemented this method in Pastry and Tapestry [8] with low overhead. It maintains system load balancing and robustness with comparable lookup latency compared to Geographic layout method. Xu *et al.* [32] proposed a method to build topology-aware overlays using global soft-state. It combines landmark clustering to generate proximity information and stores the system information (such as proximity and load information) as objects on the system itself, which is easy to update and retrieve. Waldvogel *et al.* [17] proposed a topology-aware overlay network called Mithos. Mithos does not require full topology knowledge. It provides a close conceptual integration between geographic layout and proximity routing, as well as a powerful addressing scheme directly suitable for use in DHTs. It also provides locality-aware connectivity, thereby ensuring that a message reaches its destination with minimal overhead.

The third method to improve P2P file location efficiency is to consider nodes with common interests or semantically close files. One category of such networks is called schema based networks [24–27, 33, 34]. They use explicit schemas to describe peers' contents based on semantic description, and allow the aggregation and integration of data from autonomous, distributed data sources. They provide complex query facilities but no sophisticated means for semantic clustering of peers, and their broadcasting does not scale well. Ramanathan *et al.* [21] proposed a mechanism to let a peer connect to the peer that frequently provides good results, leading to clusters of peers with similar interests. Hang *et al.* [22] proposed a method for clustering peers that share similar properties together and a new intelligent query routing strategy. In contrast to broadcasting query message, the query message will first walk around the network from peer to peer randomly, once it reaches the target cluster, the query message is broadcasted by peers inside the cluster. Crespo *et al.* [23] proposed a semantic overlay network (SON) based on the semantic relations among peers. SON establishes semantic relations between queries and peers, and routes queries directly to relevant peers. Furthermore, layered SON is proposed in order to increase the availability of results. Liu and *et al.* [18] proposed methods for supporting efficient keyword-based file search in peer-to-peer file sharing systems. The works in [19, 20] consider node interest for publish and subscribe. The works collect the information of nodes with the same or similar interest together.

The work in [35] provides a survey of the research towards robust peer-to-peer networks. In spite of the efforts devoted to efficient file location in P2P systems, there are few works that combine the super-peer topology with both interest and proximity based clustering methods. In addition, though DHTs have higher file location efficiency over unstructured P2Ps, it is a challenge to realize super-peer topology with both interest and proximity based clustering on them, due to their strictly defined topology and data allocation policy. This paper presents PAIS tackles the challenge by taking advantage of the hierarchical structure of a DHT. It significantly improves the efficiency of file location in DHTs.

3 PAIS: A Proximity-aware Interest-clustered P2P File Sharing System

3.1 PAIS Structure

PAIS is developed based on Cycloid [9] structured P2P network. Cycloid is a lookup efficient constant-degree overlay with $n{=}d \cdot 2^d$ nodes, where d is its dimension. It achieves a time complexity of $O(d)$ per lookup request by using $O(1)$ neighbors per node. Each Cycloid node is represented by a pair of indices $(k, a_{d-1}a_{d-2}\ldots a_0)$, where k is a cyclic index and $a_{d-1}a_{d-2}......a_0$ is a cubical index. The cyclic index is an integer ranging from 0 to $d-1$, and the cubical index is a binary number between 0 and 2^d-1. The nodes with the same cubical index are ordered by their cyclic index mod d on a small cycle, which we call *cluster*. All clusters are ordered by their cubical index mod 2^d on a large cycle. The Cycloid DHT assigns keys onto its ID space by a consistent hashing function [10]. For a given key, the cyclic index of its mapped node is set to its hash value modulated by d and the cubical index is set to the hash value divided by d. A key will be assigned to a node whose ID is closest to its ID. Cycloid has self-organization mechanisms to deal with node joins, departures and failures. It has APIs, including `Insert(key,object)`, `Lookup(key)`, `Join()` and `Leave()`. Cycloid routing algorithm involves three phases. A file request is routed along the cluster of the requester, between clusters, and along the cluster in the destination's cluster. For more details of Cycloid, please refer to [9].

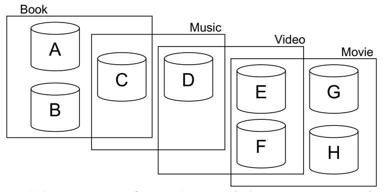


Figure 1: Clusters of nodes with common interests.

A node's interests are described by a set of attributes with globally known string description such as "image" and "music". The strategies that allow to describe content in a peer with metadata [24–26]can be used to derive the interests of each peer. Due to the space limit, we don't explain the details of the strategies.

Taking advantage of the hierarchical structure of Cycloid, PAIS gathers physically close nodes in one cluster, and further groups nodes in each cluster into sub-clusters based on their interests. We define a sub-cluster (SC) as a link structure within a network N given a set of links from client (c) to a particular supernode server (s). That is:

$$(SC_l = c_i, s_j \in N | \exists\ a\ link(c_i, s_j, l)),$$

Each SC_l supports functions: `join`(c_i, l), where links

(c_i, s_j, l) between a server and a client are created, and `leave`(c_i, l) where they are dropped.

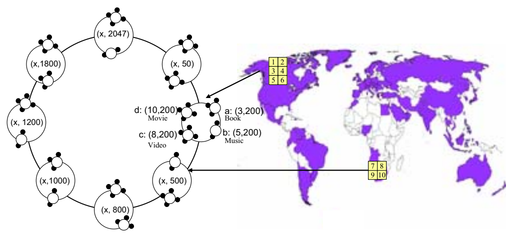


Figure 2: PAIS structure.

One node may appear in more than one sub-cluster if it has more than one interest. The sub-cluster functions as a super-peer network that has one server and a number of clients connected to it. The servers are connected into a cluster in Cycloid. All nodes in a sub-cluster have the same Cycloid ID. Figure 2 illustrates the PAIS structure corresponding to Figure 1 that shows clusters of nodes with common interests. Physically close nodes are in the same cluster, and common-interest nodes are grouped into one sub-cluster. The physically close nodes $1 - 6$ are mapped to cluster 200. The nodes interested in "book" are further grouped into sub-cluster a. All nodes in sub-cluster a have ID $(3, 200)$.

3.2 PAIS Construction and Maintenance

A question is how to build PAIS structure. Before we present the details of PAIS construction, let's introduce a landmarking method to represent node closeness on the network by indices used in [15]. Landmark clustering has been widely adopted to generate proximity information [16, 36]. It is based on the intuition that nodes close to each other are likely to have similar distances to a few selected landmark nodes, although details may vary from system to system. In DHTs, the landmark nodes can be selected by overlay itself or the network. We assume m landmark nodes that are randomly scattered in the Internet. Each node measures its physical distances to the m landmarks, and uses the vector of distances $< d_1, d_2, ..., d_m >$ as its coordinate in Cartesian space. Two physically close nodes will have similar vectors. We use space-filling curves [37], such as Hilbert curve [36], to map m-dimensional landmark vectors to real numbers. Space-filling curves map points in a m-dimension Cartesian space into a domain of real numbers; that is, $R^m \longmapsto R^1$, such that the closeness relationship among the nodes is preserved. This mapping can be regarded as filling a curve within the m-dimensional space until it completely fills the space. We partition the m-dimensional landmark space into 2^{mx} grids of equal size (where m refers to the number of landmarks and x controls the number of grids used to partition the landmark space), and number each node according to the grid into which it falls. We call this number *Hilbert number* of the node, denoted by $\mathcal{H}$. $\mathcal{H}$ indicates the physical closeness of nodes on the Internet.

Consistent hash function such as SHA-1 is widely used in DHT networks for node or file ID due to its collision-resistant nature. Using such a hash function, it is computationally infeasible to find two different messages that produce the same message digest. The consistent hash function is effective to cluster messages based on message difference.

Recall that a Cycloid ID is represented by a cyclic index and a cubical index. The cubical indices differentiate clusters, and the cyclic indices indicate different node positions in a cluster. Based on the Cycloid topology and ID determination, PAIS intelligently uses cubical indices to distinguish nodes in different physical location, and uses cyclic indices to further classify physically close nodes based on their interests. Specifically, PAIS uses node i's Hilbert number, $\mathcal{H}_i$, as its cubical index, and the consistent hash value of node i's interest mod d, $S_i\%d$, as its cyclic index to generate node i's ID, denoted by $(S_i\%d, \mathcal{H}_i)$. If a node has a number of interests, it generates a set of IDs with different cyclic indices. Using this ID determination method, the physically close nodes with the same $\mathcal{H}$ will be in a cluster, and those with similar $\mathcal{H}$ will be in close clusters in PAIS. Physically close nodes with the same interest have the same ID, and they constitute a sub-cluster.

When node i joins the system, if there already exit nodes with IDs equal to $(S_i\%d, \mathcal{H}_i)$, in the case that node i is a regular node, it becomes a client in that sub-cluster. In the case that node i is a supernode, it becomes a backup for the server in the sub-cluster. Before the server leaves, one of the backups replaces the leaving server. If there is no node with IDs equal to $(S_i\%d, \mathcal{H}_i)$, in the case that node i is a supernode, it becomes the server of the sub-cluster, and other newly-joint nodes with IDs $(S_i\%d, \mathcal{H}_i)$ will connect to it. If node i is not a supernode, it temporarily functions as the server until there is a joining supernode to replace it.

The clusters in PAIS function as super-peer network. The server in a sub-cluster acts as a centralized server to a subset of clients. Clients submit queries to their server and receive results from it, as in a hybrid system. Servers are also connected to each other as peers in a Cycloid, routing messages over this overlay network and submitting and answering queries on behalf of their clients and themselves.

To build each peer's routing table in the Cycloid, PAIS uses proximity-neighbor selection method. That is, it selects the routing table entries pointing to the physically nearest nodes among all nodes with IDs in the desired portion of the ID space. As a result, in PAIS, the logical proximity between neighbors abstraction derived from overlay approximately matches the physical proximity information in reality. Nodes in one cluster are physically close to each other, and a node is physically closer to another node in a neighbor cluster than that in a cluster far away. Due to uneven distribution of nodes in physical space, nodes may not be distributed in balance in the ID space of PAIS.

Node failures lead to intact topology and degrade DHT performance. PAIS uses stabilization to deal with node failures. Specifically each server refreshes its routing table entries and predecessor periodically to make sure they are correct. PAIS uses lazy-update to handle the influence of a server failure on its clients. Each client probes its server periodically. If a client c does not get a reply from its server s after a certain time period T, c assumes s fails, it uses PAIS node join algorithm to connect to another supernode again.

Unlike other methods that use broadcasting to cluster nodes, PAIS leverage the DHT ID determination and `lookup(key)` function to cluster nodes based on their proximity and interest, thus reducing the overhead for its construction. With the assumption that there are n_s servers in PAIS, we analyze the overhead of node dynamics in PAIS and achieve the following results.

Theorem 3.1 *In PAIS, with high probability*[1]*, a node join will incur overhead of* $O(\log^2 n_s)$ *messages.*

Proof When a node joins in PAIS, to find its closest server for one of its interests, $O(\log n_s)$ messages are required. If the new node joins as a server, another $O(\log^2 n_s)$ messages are required for neighbor update. Thus, if the node has m interests, the number of messages needed is $m(O(\log n_s) + \alpha O(\log^2 n_s)) \approx O(\log^2 n_s)$, in which α is the probability that the new node joins as a server. ▮

Theorem 3.2 *In PAIS, w.h.p., a node departure will incur overhead of* $O(\log^2 n_s)$ *messages, and a node failure will incur overhead of* $O(\log N_s)$ *messages.*

Proof According to the PAIS node leaving algorithm, a leaving client only needs $O(1)$ message (i.e. notifying its server). If a leaving server has a backup supernode, it needs $O(\log^2 n_s)$ messages. Otherwise, its clients need to rejoin the system again. Each client joining requires $O(\log n_s)$ messages. Therefore, the average number of messages caused by a node leaving is $O(1) \times \beta + (1 - \beta)(O(\log^2 n_s) + \gamma \times O(\log N_s)) \approx O(\log^2 n_s)$, where β is the percent of clients among all nodes, and γ is the average number of clients a server connects. It is easy to derive that a node failure incurs overhead of $O(\log n_s)$ messages. ▮

3.3 File Distribution

In PAIS, file ID is determined using the same way in Cycloid. That is, a file's cyclic index is its key's hash value modulated by d and its cubical index is set to the hash value divided by d, represented as $(H\%d, H/d)$, where H is consistent hash value of its key. A file's key can be its name or a string describing its contents. The file key must be consistent with the node interest. A node stores its files to the system via Cycloid interface `Insert(fileID,file)`. According to Cycloid key assignment policy, each sub-cluster is responsible for the files whose cyclic indices falls into the key space sector it supervises. Thus, files with similar keys will be in the same sub-cluster in a cluster. The supernode in a sub-cluster further distributes files among its clients in balance. For example, in Figure 2, a file with key "book" has ID $(3, 200)$, then it will be stored in a node in sub-cluster a. In node joins and departures, the files are transferred between nodes based on the key assignment policy.

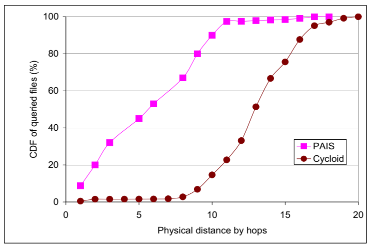
In PAIS, if node i becomes very interested in file f in its cluster, it joins the sub-cluster $(H_f\%d, \mathcal{H}_i)$. If node i is not interested in a file category any longer, it departs the sub-cluster of the interest. By this way, PAIS tunes to time-varying node interest and file popularity dynamically. PAIS relies on file replication to further improve its file location efficiency. Basically, when the request frequency of a file from a cluster of nodes with $\mathcal{H}$ exceeds a predefined threshold, the file owner replicates a file in a node in the sub-cluster closest to $(H\%d, \mathcal{H})$ in that cluster.

3.4 File Query Algorithm

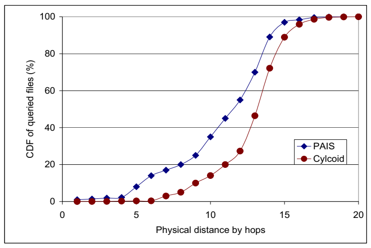
When node i requests for a file, it gets the ID for the file: $(H\%d, H/d)$. If the file's key is one of the requester's interest attributes, the node sends the request to its server in the sub-cluster of the interest. The server has the index of all files in its sub-cluster. If there is a requested file in its sub-cluster, the server sends the file location to the requester directly. Otherwise, node i checks the existence of a replica of the file. If there is a replica of the file, it is stored in a sub-cluster closest to ID $(H\%d, \mathcal{H}_i)$. Therefore, the requester send a request with $(H\%d, \mathcal{H}_i)$ as a target. If there is no replica of the file requested, the file request routing is performed based on Cycloid routing algorithm. Every time a server receives a request, it checks if its sub-cluster has the requested file. This routing algorithm does not lead to more overhead. Routing among physically close nodes greatly improves file location efficiency.

For example, in Figure 2, different files are classified into different sub-clusters based on their keys. When a node queries for a file in "book", it sends request `Lookup(3,200)` to its server. The requester gets the location of the file from the server if there is a requested file in its sub-cluster. Otherwise, the request is routed along its own cluster. Each server in the sub-cluster of the cluster checks if there is a requested file. If there is no requested file in the cluster, the request will be routed based on Cycloid routing algorithm which will forward the request to sub-cluster a. Then, the server of the sub-cluster a replies to the requester of the location of the file. Theorem 3.3

[1]An event happens with high probability (w.h.p.) when it occurs with probability $1 - O(n^{-1})$.



(a) ts5k-large



(b) ts5k-small

Figure 3: CDF of total queries distribution.

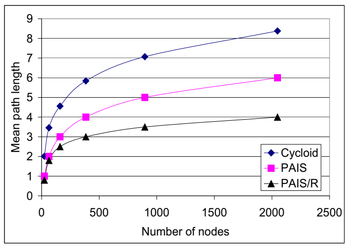


Figure 4: Hops per file query.

demonstrates the efficiency of file location in PAIS.

Theorem 3.3 *W.h.p., the lookup path length for a file query in PAIS is bounded by $O(d)$ hops.*

Proof In PAIS, it takes a requester one hop to inquire its server for a requested file, and takes $O(d)$ to check a possible replicated file. Using Cycloid routing algorithm, the path length is $O(d)$. Therefore, w.h.p., the lookup path length for a file query is bounded by $O(d)$ hops. ▮

4 Performance Evaluation

This section presents the performance evaluation of PAIS in average case in comparison with Cycloid. We chose Cycloid as reference is because it is comparable to PAIS regarding the fundamental DHT structure and performance. The experiment results demonstrate the improvement of PAIS over Cycloid in terms of proximity-aware performance, file query efficiency, communication cost in both static and dynamic situations. In the experiments, the DHT's dimension was set to 8 and it had 2048 nodes. We assumed there were 200 interest attributes (i.e. file keys), and each attribute had 500 files. We assumed a bounded Pareto distribution for the capacity of nodes. This distribution reflects the real world situations where machine capacities vary by different orders of magnitude. The number of queried files was set to 50, and the number of queries per file was set to 1000, unless otherwise specified. The file requesters and the queried files were randomly chosen.

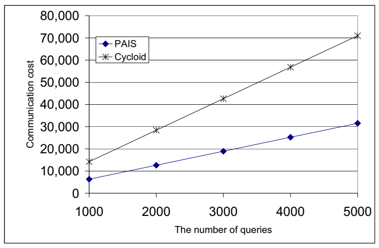
We used two transit-stub topologies generated by GT-ITM [38]: "ts5k-large" and "ts5k-small". "ts5k-large" has 5 transit domains, 3 transit nodes per transit domain, 5 stub domains attached to each transit node, and 60 nodes in each stub domain on average. "ts5k-small" has 120 transit domains, 5 transit nodes per transit domain, 4 stub domains attached to each transit node, and 2 nodes in each stub domain on average. "ts5k-large" has a larger backbone and sparser edge network (stub) than "ts5k-small." "ts5k-large" is used to represent a situation in which DHT overlay consists of nodes from several big stub domains, while "ts5k-small" represents a situation in which DHT overlay consists of nodes scattered in the entire Internet and only few nodes from the same edge network join the overlay. To account for the fact that interdomain routes have higher latency, each interdomain hop counts as 3 hops of units of latency while each intradomain hop counts as 1 hop of unit of latency.

4.1 Effectiveness of Proximity Awareness

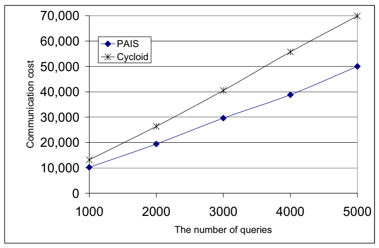
In this section, we show the effectiveness of PAIS on achieving proximity-aware file query in which requesters get files from physically close nodes. Figure 3(a) and (b) show the cumulative distribution function (CDF) of the number of queried files versus the physical distance in "ts5k-large" and "ts5k-small" respectively. We can see that in "ts5k-large," in PAIS, 95% of total files are offered by nodes within 10 hops, while in Cycloid, 15% of files are offered by nodes within 10 hops. Almost all files in PAIS are offered by nodes within 15 hops, while only 75% of the total files are offered by nodes within 15 hops in Cycloid. The results show that most files can be offered by servers in short physical distances in PAIS, while most files are offered in long distances in Cycloid. From Figure 3(b), we can have the same observations as in "ts5k-large," although the performance difference between systems is not so significant as in "ts5k-large." The more files offered by nodes in the shorter distances, the higher proximity-aware performance of a P2P system with less query cost. The results indicate that PAIS improves Cycloid with regards to the proximity-aware performance in offering files by physically close nodes either when nodes are from several big sub domains or when nodes are scattered in the entire Internet.

4.2 Efficiency of File Query

The second experiment was designed to evaluate the efficiency of file query. In this experiment, we simulated networks with $n = d \cdot 2^d$ nodes and varied the dimension d from 3 to 8. Each node made a total of $n/4$ lookup requests to random destinations. Figure 4(a) shows the average logical hops for a file query in average case. "PAIS/R" represents PAIS with file replication algorithm. The path length metric is measured by the number of hops traversed during a search until a file query is resolved. We can see that PAIS incurs less number of hops than Cycloid. By clustering common interest nodes, a file query does not need to be forwarded in the entire system. In most cases, a query only needs to be routed in the requester's cluster, thus leading to less lookup path length. We can also



(a) ts5k-large



(b) ts5k-small

Figure 5: Communication cost for file query.

observe that PAIS/R generates shorter path length than PAIS. It illustrates the effectiveness of PAIS to replicate files in nodes with high query frequency, which enhances the utilization of replicas and hence reduces the lookup path length.

4.3 Communication Cost

The cost of file searching is directly related to message size and physical distance of the message travelled; we use the product of these two factors of all file queries to represent communication cost. It is assumed that the size of a file query is 1 unit. Figure 5(a) and (b) plot the file searching communication cost of PAIS and Cycloid in "ts5k-large" and "ts5k-small" respectively. From these figures, we can see that the cost increases as the number of file queries increases, and Cycloid incurs considerably higher cost than PAIS. There are two reasons for the observation. First, PAIS reduces the lookup path length of Cycloid. Second, because Cycloid neglects proximity, file query messages travel long physical distances. In contrast, PAIS proactively considers proximity in P2P construction for file query, such that the messages only travel between physically close nodes. Its shorter lookup path length and shorter physical message travel distance result in low-overhead and timely file queries.

4.4 Performance in a Highly Dynamic Environment

This section evaluates the efficiency of PAIS in a highly dynamic environment. In this experiment, as in [6], file lookups are generated according to a Poisson process at a rate of one per second. Node joins and voluntary leaves are modelled by a Poisson process with a mean rate of R, which ranges from 0.1 to 0.4. A rate of R = 0.1 corresponds to one node joining and leaving every 10 seconds on average. Each node invokes the stabilization protocol once every 30 seconds and each nodes stabilization routine is at intervals that are uniformly distributed in the 30 seconds interval.

Experiment results show that there were no lookup failures in all test cases. Figure 6 shows the average number of logical hops for the requests as R changes. Compared with the logical hops evaluation in Figure 4, we can see that the measured number of hops in the presence of node joining and leaving is very close to that value and does not change greatly with the rate R. The results imply that DHT self-organization mechanisms can deal with dynamics. Figure 7(a) and (b) plot the communication cost for system maintenance in "ts5k-large" and "ts5k-small" respectively. The communication cost includes the cost for file query and system stabilization in dynamics. From these figures, we can see that the communication cost increases slightly with the node join/leave rate, and that of Cycloid is dramatically higher than PAIS. Fast node joins and departures make more nodes involved in system stabilization, leading to more messages and higher communication cost. Note that the communication cost is due to the number of messages for file query and stabilization and the physical distances between nodes. Cycloid does not consider physical distance in system construction, so that nodes need to contact physically far nodes for file query and system stabilization in dynamics. This is the main reason for its high total communication cost. In contrast, PAIS takes into account proximity in system construction which enables nodes to communicate physically close nodes for file query and stabilization. Therefore, PAIS incurs less communication cost for file query and system stabilization in dynamics. In conclusion, the experiment results confirm that PAIS can efficiently resolve file queries in a dynamic environment with lower communication cost.

5 Conclusions

To enhance file location efficiency in P2P systems, interest-clustered super-peer network or proximity-clustered super-peer networks have been proposed. Though both strategies improve the performance of P2Ps, few works cluster peers based on both peer interest and physical proximity. Moreover, it is harder to realize it in structured P2Ps due to their strictly defined topologies, although they have high efficiency of file location than unstructured P2Ps. In this paper, we introduce a proximity-aware and interest-clustered P2P file sharing system (PAIS) based on a structured P2P. It groups peers based on both interest and proximity by taking advantage of a hierarchical structure of a structured P2P. Theoretical analysis and simulation results demonstrate the efficiency of PAIS in comparison with another P2P file sharing system. It dramatically reduces the overhead and yields significant improvements in file location efficiency.

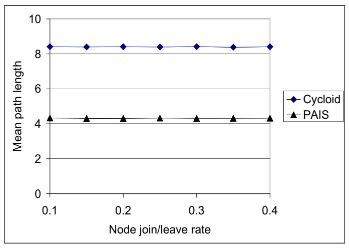
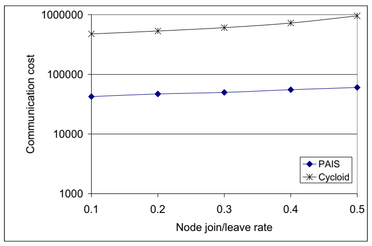
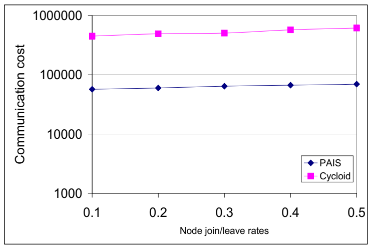


Figure 6: Hops per query in dynamics.



(a) ts5k-large



(b) ts5k-small

Figure 7: Communication cost for file query in dynamics.

Acknowledgements

This research was supported in part by U.S. NSF grants CNS-0834592 and CNS-0832109.

References

[1] More p2p Users Than Ever. http://www.dvd-recordable.org/Article2439.phtml.

[2] Kazaa. heep://www.kazaa.com.

[3] Gnutella home page. http://www.gnutella.com.

[4] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proc. of the International Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2001.

[5] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proc. of ICS*, 2001.

[6] I. Stoica, R. Morris, D. Liben-Nowell, and et al. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *TON*, 1(1):17–32, 2003.

[7] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, pages 329–350, 2001.

[8] B. Y. Zhao, L. Huang, and et al. Tapestry: An Infrastructure for Fault-tolerant wide-area location and routing. *J-SAC*, 12(1):41–53, 2004.

[9] H. Shen, C. Xu, and G. Chen. Cycloid: A Scalable Constant-Degree P2P Overlay Network. *Performance Evaluation*, 63(3):195–216, 2006. An early version appeared in IPDPS, 2004.

[10] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proc. of STOC*, pages 654–663, 1997.

[11] Z. Li, G. Xie, and Z. Li. Efficient and Scalable Consistency Maintenance for Heterogeneous Peerto- Peer Systems. *TPDS*, 2008.

[12] H. Shen and C. Xu. Hash-based proximity clustering for efficient load balancing in heterogeneous DHT networks. *Journal of Parallel and Distributed Computing (JPDC)*, 2008.

[13] Fasttrack. http://www.fasttrack.nu/index_int.html.

[14] Gnutella development forum.

[15] H. Shen and C.-Z. Xu. Hash-based proximity clustering for efficient load balancing in heterogeneous dht networks. *JPDC*, 2008.

[16] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proc. of INFOCOM*, 2002.

[17] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. In *Proc. of HotNets-I*, 2002.

[18] L. Liu, K. D. Ryu, and K.-W. Lee. Keyword fusion to support efficient keyword-based search in peer-to-peer file sharing. In *Proc. CCGrid*, pages 269–276, 2004.

[19] Y. Zhu and H. Shen. An efficient and scalable framework for content-based publish/subscribe systems. *PPNA*, 2008.

[20] D. Tran and T. Nguyen. Publish/Subscribe Service in CAN-based P2P Networks: Dimension Mismatch and the Random Project Approach. In *Proc. of ICCCN*, 2008.

[21] M. K. Ramanathan, V. Kalogeraki, and J. Pruyne. Finding Good Peers in Peer-to-Peer Networks. In *Proc. of IPDPS*, 2002.

[22] C. Hang and K. C. Sia. Peer Clustering and Firework Query Model. In *Proc. of WWW*, 2003.

[23] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. of ICDCS*, 2002.

[24] W. Nejdl, W. Siberski, M. Wolpers, and C. Schmnitz. Routing and clustering in schema-based super peer networks. In *Proc. of IPTPS*, 2003.

[25] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *Proc. of WebDB*, 2002.

[26] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *Proc. of WWW*, 2003.

[27] K. Aberer, P. Cudrè-Mauroux, and M. Hauswirth. The chatty web: Emergent semantics through gossiping. In *Proc. of WWW*, 2003.

[28] Morpheus. http://www.musiccity.com.

[29] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proc. of ICDE*, 2003.

[30] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM*, pages 329–350, 2001.

[31] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. In *Proc. of FuDiCo*, 2002.

[32] Z. Xu, C. Tang, and Z. Zhang. Building topology-aware overlays using global soft-state. In *Proc. of ICDCS*, 2003.

[33] A. Löser, W. Nejdl, M. Wolpers, and W. Siberski. Information integration in schema-based peer-to-peer networks. In *Proc. of CAiSE*, 2003.

[34] W. Nejdl, M. Wolpers, W. Siberski, A. Löser, I. Bruckhorst, M. Schlosser, and C. Schmitz. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *Proc. of WWW*, 2003.

[35] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks. *Computer Networks*, (17), 2006.

[36] Z. Xu and et al. Turning Heterogeneity into an Advantage in Overlay Routing. In *Proc. of INFOCOM*, 2003.

[37] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmaier. Space filling curves and their use in geometric data structure. *Theoretical Computer Science*, 181(1):3–15, 1997.

[38] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. of INFOCOM*, 1996.