Efficient Person Searching in a Peer-to-Peer Network

Meng I Lei S.-H. Gary Chan
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong

Albert Kai-Sun Wong
Dept of Electrical and Electronic Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong

***Abstract*— Internet applications such as Voice over IP and Instant Messaging require the support of efficient person search. In these applications, a user often has a list of people whom he/she often calls, the so-called "*hot list*." Traditional structured peer-to-peer networks for file sharing, despite of their support of unique item search, are not efficient for person search. This is mainly because they do not take into account human communication characteristics such as close social network, user-dependent hot list, and high skewness in access popularity. We show in this paper that person search can be much more efficient by taking into consideration of our communication features. Based on Pastry, our system make use of hot lists in the routing process and clusters people of the same community together in the search graph, thereof creating a "small-world" effect. Simulation results show that our scheme achieves significantly lower search time than Pastry (by as much as 50%).**

I. Introduction

In recent years, there has been much research in peer-to-peer (P2P) networks. As different from much of this previous work on file searching, we address in this paper the problem of locating a unique or specific item in a P2P network. In particular, we are interested in finding a *person* in the network.

Indeed, many exciting applications, such as Voice over IP (VoIP) and Instant Messaging (IM), would benefit from the support of efficient person search. Nowadays, the search mechanism of these applications still operate in a rather centralized manner. Users are required to register with a central server when they log into the system and query it for the addresses of their target search people. This is not scalable and the server introduces a single point of failure. We consider in this paper a completely decentralized approach, leading to a system scalable to large number of users.

In person-searching applications (as in VoIP and IM), a person usually has a *hot list* which contains the people he/she often calls,[1] such as colleagues, family members and friends. It has long been observed that human communication, and hence the hot list, exhibits the following characteristics:

- *Strong access skewness:* Our hot list is often short (of size around not more than ten), which means that our contacts often skewed towards only a few people. Some people also are popular and have a relatively dense social connection. They are not only often called, but also good candidates to be contacted when searching for an unknown person.
- *User-dependent:* The calls and hot lists are very user-dependent or user-specific, where a person popular in one domain may not be so in another.
- *Dense social network:* Our social network has high locality and is closely knit together. This is the so-called "small world" effect in which the person we call is likely known by our direct or non-distant friends [1], [2]. Furthermore, users in the same domain (say within the same company or institute) or community (say alumni of a university) often communicate with each other.

It is clear that in order to locate a person efficiently, the search mechanism should take advantage of the above communication patterns. Our work hence differs from the traditional work on file searching in two major ways:

- *Item uniqueness:* While multiple copies of files or resources exist in a file sharing system, the target item we are locating is unique. Consequently, schemes taking advantage of the popularity of items (such as random walk and flooding) would not be efficient.
- *Access pattern:* As mentioned before, the access pattern in person searching is very user-dependent and domain-specific. This is in marked contrast with file searching where the file popularity is generally assumed to be the same for all users. This fundamentally changes the underlying system assumption. If one naively applies file-search mechanisms to person-searching, the search therefore would not be effective.

Generally, P2P overlay networks can be categorized into unstructured and structured. In an *unstructured overlay* (such as Gnutella), peers are organized in a random graph and use flooding or random walks to locate items. Clearly, it is very resource-consuming to search for rare items on a large network. A negative answer to a search may mean that either the item does not exist or it is not found within a certain

This work was supported, in part, by the Areas of Excellence (AoE) Scheme on Information Technology funded by the University Grant Council in Hong Kong (AoE/E-01/99), and by the Research Grant Council in Hong Kong (HKUST6156/03E).

[1]We will use "call" to refer to a search command in this paper, due to the obvious context our study is conducted under.

0-7803-9415-1/05/$20.00 (C) 2005 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2005 proceedings.

search scope. A *structured overlay* such as Pastry overcomes the search inefficiency for rare items [3], [4]. It assigns keys to items using a hash table and organizes the participating peers into a graph based on their IDs (the so-called nodeID). The graph is structured to facilitate the search for an item based on its key, by routing the message to a responsible peer who has nodeID numerically closest to the key.

Obviously, a structured overlay is more efficient to search for unique items and can be applied in our study. However, since all items are treated equally in structured overlays, the search time for popular people (such as the ones in the hot list) cannot be fast based on Pastry. We therefore propose a scheme which makes use of the user's hot list to form the routing table for the search phase, leading to faster search time for the ones a user calls often. Furthermore, by intelligently assigning nodeIDs to peers, peers of the same domain can be close to each other in the overlay search graph. Due to access locality within a domain, the search goes to a domain first before the person is located. This greatly reduces search time. Simulation shows that the search time for our approach is much faster than the traditional Pastry (by as much as 50%).

This paper is organized as follows. We first discuss related work in Section II. We briefly review Pastry in Section III, which serves as a substrate of our structured overlay. We present our scheme in Section IV, followed by illustrative simulation results in Section V. We conclude in Section VI.

II. Related Works

There has been much work on how to improve the lookup latency of structured P2P networks, such as Pastry [3] and Chord [4]. These techniques can be generally categorized into *topology-aware* and *interest-based* [5], [6].

Pastry is a topology-aware approach. To reduce search time, the routing table of a joining peer is initialized by using the routing tables of nearby peers in the nodeID space. The intention is to have the entries of the routing table pointing to some peers of close location in the search graph. *Lookup-parasitic random sampling (LPRS)* [7], a variation of Pastry, initializes the routing table without considering entry locality, it discovers nodes in close location by random sampling during the lookup process. Our approach extends the idea by making use of the hot list to fill the routing table with nodes of high access frequency. Consequently, our search is more efficient because most of the searches can be completed in one hop. Other topology-aware approaches include *expressway* [8]. In expressway, resource heterogeneity of hosts is considered. The network is partitioned into regions, with the most resourceful peers of a region promoted as *express-way neighbors* and short-cuts made to them. Each express-way neighbor is a representative serving all other peers in that region. Our scheme clusters peers and provides short-cuts to the clusters. However, our scheme does not have any fixed cluster representative, hence avoiding representative overloading. In order to reduce search time, we group those likely to communicate with each other together.

Associative overlays is an interest-based approach based on unstructured P2P overlay to locate rare items [9]. A peer is connected to other peers which possess the same item. The intuition is that peers sharing the same item may share the same interest. It is hence more likely to locate the target item from the peers of the same interest, even the target may be a rare one. However, the more items a peer has, the more neighbors it needs to keep. Moreover, it shares the same weakness of unstructured overlay in that its answers to a query is non-deterministic. Our scheme runs on a structured P2P network and hence deterministic answers can be provided. In addition, with the size of routing table fixed, our overhead of maintaing routing table is low.

III. Pastry Overview

Pastry is a scalable, decentralized and self-organizing P2P object location and routing substrate. It automatically adapts to node arrivals, departures and failures.

Each peer in Pastry is assigned randomly a 128-bit node identifier (nodeID) when a node joins the system. The nodeID is used to indicate a node's position in a circular nodeID space (a logical ring), which ranges from 0 to $2^{128}-1$.

An item[2] in the network is given an unique key, whose key space is the same as that of the nodeID. An item is handled by a "responsible" node, whose nodeID is numerically closest to the item key. All search queries for a resource should be sent to the responsible node.

Every Pasty node maintains a routing table and a leaf set for message routing. The routing table is organized into $\log_{2^b} N$ rows and 2^b-1 columns, where N is the maximum number of nodes in the overlay and b is a parameter determining the size of the routing table (normally chosen to be 4). The entries in row n refer to nodes whose nodeIDs share exactly the first n prefix digits with the current node. Entries of the routing table are (nodeID, IP-address) pair. It is a node with known IP-address and whose nodeID corresponds to the shared prefix. The node is chosen to be near the current node according to some proximity metric, such as the number of hops in IP routing or geographic distance. If no such node is known, the entry is left empty. The leaf set contains L numerically close nodes with the current node in the logical ring.

In Pastry, messages are routed according to the longest prefix-matching principle. If the target item key is found in the leaf set, then the message is routed directly to the destination node in the next hop (the destination node is with nodeID numerically closest to the key). If the key is not in the leaf set, the current node looks up its routing table a node whose nodeID shares the longest prefix with its own nodeID and routes the message to this node. If there is no such node, the message is routed to a node that shares the same prefix length with the present node but is numerically closer to the target item key.

[2]An item is any resource we are searching for. Examples are music files or a person in our context.

TABLE I

HASHING OF USER IDENTIFIER

Hashing			
Alice	11	ust.hk	210
Bob	20	umac.mo	110
Cary	22		

NodeID	
Alice@ust.hk	21011
Bob@umac.mo	11020
Cary@ust.hk	21022

IV. SCHEME DESCRIPTION

Pastry is adopted as the basis of our system. Our goal is to minimize hop counts for a person search. We address this in two aspects, nodeID assignment and routing table construction.

A. NodeID Assignment

Note that in Pastry, nodeID and item key are assigned independently, thought they share the same key space. In our system, a person is represented by an unique username in the form of somebody@somedomain; the person key is generated from the username by a hash table. A machine is represented by a nodeID. Since we assume a person is accessed via a machine, we may regard the nodeID the same as the person key, and a search for the person corresponds to a search for the nodeID.

Note that if two nodes are close to each other in the logical ring, the number of hops required to route a query is small. Clearly, it is important to assign nodeID in such a way that people who are likely to look for each other share longer prefix of nodeID so that they are placed close in the logical ring. The challenge here is that we do not know in advance a person's hot list. Moreover, it is infeasible to assign nodeID to a person based on its hot list. A reason is that hot list may change over time, but a person's nodeID should represent his/her identity and should not be changed frequently. Given that people from the same domain are more likely to know, and hence call, each other, we propose a two-step hash function to generate the nodeID shown in Table I. We hash the username of the identifier and get a string of length m (which is 2 in the table), and the domain name of the identifier and get a string of length n (which is 3 in our example). We concatenate these two strings, with the domain name as the prefix. Our assumption is due to the fact that people of the same domain belong to an organization, company, or service provider, and are more likely to call each other due to locality.

B. Routing Table

All nodes participate in message routing. When a new node joins the network, it starts populating its routing table. A new node X knows about another nearby node Y by contacting the bootstrap server. X then asks Y to route a JOIN message back to X. The message is then forwarded to the node Z who is numerically closest to X. In response to receiving the JOIN message, nodes Y, Z, and all other nodes on the path send to X their routing tables, using which X initializes its own routing table. Figure 1 shows the hot list and the initial routing table of Alice.

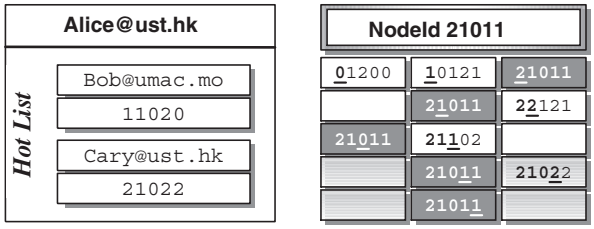


Fig. 1. Alice's hot list and initial routing table.

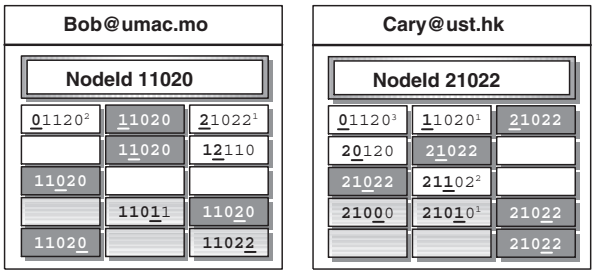
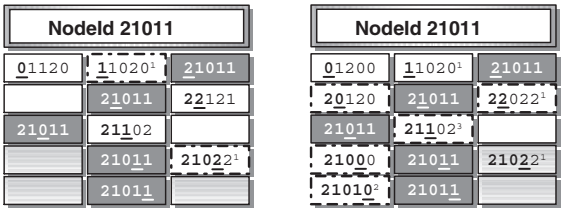


Fig. 2. Routing Tables of Bob and Cary.

After that, *table exchange* takes place. The aim is to populate a node's routing table with entries of high access rate. We define buddies in a person's hot list as first-hop buddies, and first-hop buddies' buddies as second-hop buddies, etc. We call such hop *social hop* which indicates a distance in a social network. Due to social network, a person accesses his/her first-hop buddies with the highest access rate, and the second-hop buddies with lower access rate and so on. Furthermore, since the hot list of a person and his/her buddies may be correlated, routing a message to one's buddies may increase the success probability. The following are the steps of table exchange to create a social network:

1) The new node first searches for all its buddies in its hot list and replaces entries of its routing table by its buddies. The entries are marked to indicate they are first-hop buddies.
2) The node randomly selects a buddy from its hot list, and acquires routing table from it.
3) The node looks for and replaces its routing table with entries of the acquired table, which are of fewer social hops.
4) Repeat Steps 2 & 3 until the routing table stabilized.

As an example, Fig. 2 shows the routing tables of Alice's first-hop buddies, Bob and Cary. Note that Alice and Cary are of the same domain, so their keys share the same prefix 210. Figure 3 shows the routing table of Alice after Step 1 of table exchange (Fig. 3 a), and the final routing table of Alices after acquiring and replacing entries with Alice's first-hop buddies (Fig. 3 b). Note that the table exchange procedure also fills



(a) Alice's routing table after filling with buddies.

(b) Alice's final routing table.

Fig. 3. First, Alice's routing table is filled with members of her hot list. Second, the routing table is populated with entries of high access rate. The entries are marked to indicate the ranking of access rate. By acquiring tables, more entries are filled.

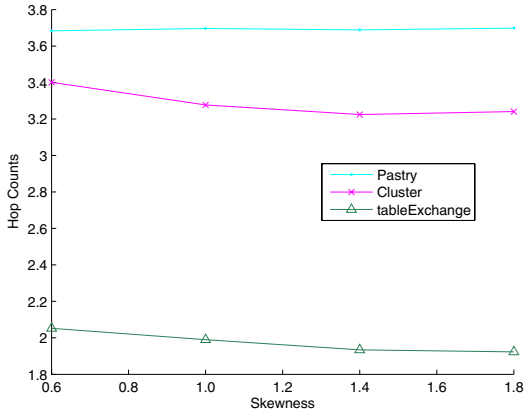


Fig. 4. Average hop counts against skewness (a).

Alice's routing table with more entries.

When a query is routed to a node, the node first checks if this is a lookup message for itself. If so, the node processes the query accordingly. If not, the node checks the target against its hot list. Failure to find the target in the hot list leads to normal routing procedure of Pastry: the node checks if the target can be found in leaf set followed by entries in the routing table. This reduces access time to one's first-hop buddies to one search hop, and ensures that accesses to other nodes are not worse than Pastry.

V. Illustrative Simulation Results

Simulations are performed to compare the performance of our scheme with that of Pastry. The performance metric we are interested is the average hop counts, which is defined by the average number of routing hops for a search to reach its destination. We study the enhancements made by clustering the nodes, and the improvements contributed by exchanging routing tables with buddies.

A. Simulation Setup

Simulations are run on the simulator, J-Sim, and a network topology is generated by GT-ITM with 1024 routers. Various number of nodes are attached to the routers randomly. The relations among nodes should be comparable to social relations among people. We use a multi-component static model [10] to simulate the social relation.

1) Generation of Hot List: The model supposes there are q groups and N people in the social network. m and f are two other system parameters determining how a person make acquaintances with others. m is related to the number, whereas f is related to the pattern.

The model is constructed as follows. Initially, there are N people in the society. Each person i is represented as a vertex and is assigned a q-component weight $(w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(q)})$. The weight represents the ranking of that person in that group. Edges are connected between two vertices if the two people know each other. First, we choose a group μ among the q groups. Then, two vertices (i, j) are chosen with probabilities equal to normalized weights, $p_i \equiv w_i^{\mu} / \sum_k w_k^{(\mu)}$ and $p_j \equiv w_j^{\mu} / \sum_k w_k^{(\mu)}$. Edges are attached to the two vertices unless they are already connected in the same group. The process is repeated until $(1 - f)mN$ edges are added to the system. To mimic social relations, people should know people from other groups. This social relationships are formed following the maximum weights among the q components each individual has. The normalized maximum weight of vertex i is defined as $w_i = \max(p_i^{(1)}, \ldots, p_i^{(q)})$. Then two distinct vertices i and j are chosen with probabilities, $w_i / \sum_k w_k$ and $w_j / \sum_k w_k$, respectively. The process is continued until fmN edges are formed.

In our simulation, we set m to 3 so that on average each person knows 5 others. f is chosen to be 0.2, which is shown to be optimal in other study [10].

2) Generation of nodeID: nodeID for each node is generated with parameters b set to 2, l set to 10 (prefix bit number l_p to be 6 and the suffix bit number l_s to be 4), and L set to 4. These parameters affect the size of routing tables and leaf sets. With these parameters set to a smaller value, the overhead size decreases. However, as Pastry route a message in $\lceil log_{2^b} N \rceil$ steps, a small value of b may increase the hop counts. Our scheme, which augments a node with hot list, can compensate the increase while keeping the overhead small.

3) Generation of Search Queries: We assume the call pattern of a person follows Zipf distribution. For each person, we generate a *call rank list* that contains all others in the system. The person's direct buddies rank the highest in the list, second hop buddies the next, and so on. Each individual searches another person with probability $p \sim 1/i^a$, where i is the ranking and a is set to 1.4, unless state otherwise.

B. Results

We first study how the skewness of search queries related to the average hop counts. We choose a network with $N = 1024$, $q = 4$ as the baseline. In Fig. 4, we see that the average hop counts generally drops with increasing skewness. Pastry is insensitive to the skewness, which is expected, since it does not consider search pattern. Clustering the user gives improvement to the original Pastry, the average hop counts drops with increasing skewness. The improvement is contributed by both

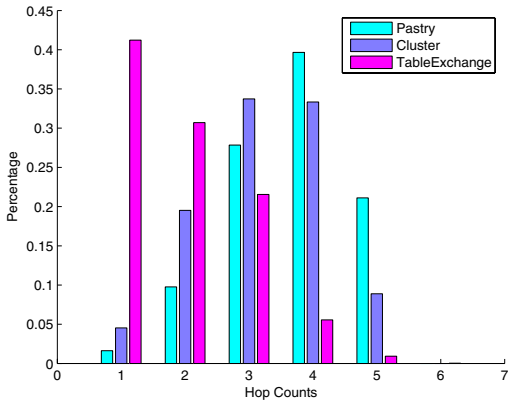


Fig. 5. Hop Distribution of Pastry, Cluster and Table Exchange. $N = 1024$, $q = 4$, $h = 5$ and $a = 1.4$.

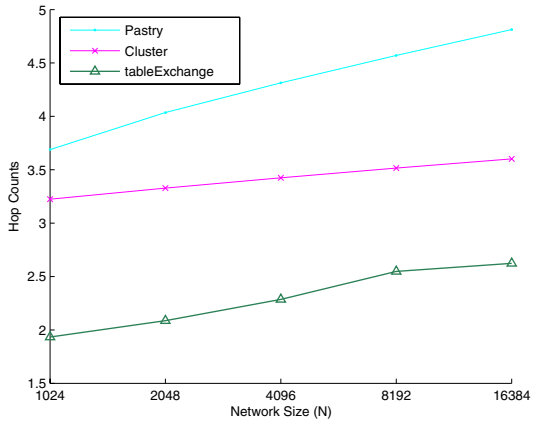


Fig. 6. Average hop counts against network size.

the use of hot lists and the shortcuts to various groups. There is further enhancement made by exchanging routing tables with buddies. In our experiments, we tested different numbers for the table exchange, and found 100 gives a steady system. For each exchange, a node randomly choose a buddy from its hot list, request its routing table and replace entries. After the process of exchange, a node's routing table is now populated with entries of high access rate.

Figure 5 shows the hop disturbution of the three scheme, Pastry, Cluster and Table Exchange. We see that most of the searches in Pastry are completed in several hops, (mostly four), whereas most of the searches in Table Exchange are done in one hop (with over 40%). Hot lists allow searches for the most frequently accessed people to be completed in one hop, and this contributes to the increase of percentage of one hop of Cluster compared to Pastry. Table Exchange puts peers of short social hops in the routing table so the percentage of one hop increases much.

We study how our scheme is related to large network size. In Fig. 6, it is shown that clustering is not affected much by increase in network size. This is because the number of groups increases with the network size, and the improvement made by shortcuts to various groups increases as well. Improvements made by exchanging routing tables decreases with larger network size. This is due to the fact that the size of routing table remains the same for all network size, and the entries it can hold remain the same.

VI. Conclusion

Previous studies made for *file sharing applications* are not applicable to *person searching applications*. To build a peer-to-peer network with efficient search performance for person searching applications, structured overlay should be used, user access pattern should be taken into account. Simulation results confirm that clustering users of same domain and populating routing table with entries of high access rate give decrease in routing hops. Our scheme has relatively the same overhead with Pastry in terms of storage of information, since the hot list kept by each node is small in size and other information is the same size.

References

[1] J. Kleinberg, "Navigation in a small world," in *Nature*, Aug. 2000.
[2] J. Guare, "Six degrees of separation: A play." in *Vintage Books*, New York, 2000.
[3] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, Nov. 2001, pp. 329–350.
[4] D. K. F. K. I. Stoica, R. Morris and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM*, Aug. 2000, pp. 149–160.
[5] D. Karger and M. Ruhl, "Diminished chord: A protocol for heterogeneous subgroup formation in peer-to-peer networks," in *International Workshop on Peer-to-Peer Systems (IPTPS '04)*, San Diego, Feb. 2004, pp. 288–297.
[6] M. N. Gurmeet Singh Manku and U. Wieder, "Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, Chicago, IL, USA, 2004, pp. 54–63.
[7] R. G. H. Zhang, A. Goel, "Incrementally improving lookup latency in distributed hash table systems," in *ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, vol. 31, San Diego, CA, USA, June 2003, pp. 114–125.
[8] M. M. Z. Xu and M. Karlsson, "Turning heterogeneity into an advantage in overlay routing," in *INFOCOM 2003*, vol. 2, San Francisco, CA, USA, Apr. 2003, pp. 1499–1509.
[9] A. F. E. Cohen and I. Kaplan, "A case for associative peer to peer overlays," in *ACM SIGCOMM Computer Communication Review*, vol. 33, Jan. 2003, pp. 95–100.
[10] B. K. D.-H. Kim and D. Kim, "Multi-component static model for social networks," in *The European Physical Journal B*, Feb. 2004, pp. 305–309.