# Location Awareness in Unstructured Peer-to-Peer Systems

Yunhao Liu, *Member*, *IEEE*, Li Xiao, *Member*, *IEEE*, Xiaomei Liu, Lionel M. Ni, *Fellow*, *IEEE*, and Xiaodong Zhang, *Senior Member*, *IEEE*

**Abstract**—Peer-to-Peer (P2P) computing has emerged as a popular model aiming at further utilizing Internet information and resources. However, the mechanism of peers randomly choosing logical neighbors without any knowledge about underlying physical topology can cause a serious topology mismatch between the P2P overlay network and the physical underlying network. The topology mismatch problem brings great stress in the Internet infrastructure. It greatly limits the performance gain from various search or routing techniques. Meanwhile, due to the inefficient overlay topology, the flooding-based search mechanisms cause a large volume of unnecessary traffic. Aiming at alleviating the mismatching problem and reducing the unnecessary traffic, we propose a location-aware topology matching (LTM) technique. LTM builds an efficient overlay by disconnecting slow connections and choosing physically closer nodes as logical neighbors while still retaining the search scope and reducing response time for queries. LTM is scalable and completely distributed in the sense that it does not require any global knowledge of the whole overlay network. The effectiveness of LTM is demonstrated through simulation studies.

**Index Terms**—Peer-to-peer, topology matching, flooding, location-aware topology, search efficiency.

## 1 Introduction

Since the emergence of peer-to-peer file sharing applications, such as Napster, Gnutella, and KaZaA, millions of users have started using their home computers for more than browsing the Web and exchanging e-mails. Peers join the network by connecting to at least one of the active peers in the P2P overlay network. Each peer acts as both a client who requests information and services and a server who produces and/or provides information and services.

There are mainly three different architectures for P2P systems: centralized, decentralized structured, and decentralized unstructured [18]. In the centralized model, such as Napster [6], central index servers are used to maintain a directory of shared files stored on peers so that a peer can search for the whereabouts of a desired content from an index server. However, this architecture creates a single point of failure and its centralized nature of the service also makes systems vulnerable to denial of service attacks [15]. Decentralized P2P systems have the advantages of eliminating reliance on central servers and providing greater freedom for participating users to exchange information and services directly between each other. In decentralized structured models, such as Chord [33], Pastry [27], Tapestry [38], and CAN [24], the shared data placement and topology characteristics of the network are tightly controlled based on distributed hash functions.

This paper focuses on decentralized unstructured P2P systems, such as Gnutella [3] and KaZaA [5]. File placement is random in these systems, which has no correlation with the network topology [37]. Unstructured P2P systems are most commonly used in today's Internet. The most popular search mechanism in use is to blindly flood a query to the network among peers (such as in Gnutella) or among supernodes (such as in KaZaA). A query is broadcast and rebroadcast until a certain criterion is satisfied. If a peer receiving the query can provide the requested object, a response message will be sent back to the source peer along the inverse of the query path and the query will not be further forwarded from this responding peer. This mechanism ensures that the query will be "flooded" to as many peers as possible within a short period of time in a P2P overlay network. A query message will also be dropped if the query message has visited the peer before.

Studies in [31] and [29] have shown that P2P traffic contributes the largest portion of the Internet traffic based on their measurements on some popular P2P systems, such as FastTrack (including KaZaA and Grokster) [2], Gnutella, and DirectConnect. Measurements in [25] have shown that, even given that 95 percent of any two nodes are less than seven hops away and the message time-to-live (TTL = 7) is preponderantly used, the flooding-based routing algorithm generates 330 TB/month in a Gnutella network with only 50,000 nodes. A large portion of the heavy P2P traffic caused by inefficient overlay topology and the blind flooding is unnecessary, which makes the unstructured P2P systems far from scalable [26]. There are three reasons for this problem. First, the mechanism of a peer randomly choosing logical neighbors without any knowledge about the underlying physical topology causes topology mismatch between

- *Y. Liu and L.M. Ni are with the Department of Computer Science, Hong Kong University of Science and Technology, Kowloon, Hong Kong. E-mail: {liu, ni}@cs.ust.hk.*
- *L. Xiao and X. Liu are with the Department of Computer Science and Engineering, 3115 Engineering Bldg., Michigan State University, East Lansing, MI 48824. E-mail: {lxiao, liuxiaom}@cse.msu.edu.*
- *X. Zhang is with the Department of Computer Science, College of William and Mary, Williamsburg, VA 23187-8795. E-mail: zhang@cs.wm.edu.*

*Manuscript received 21 Sept. 2003; revised 18 Apr. 2004; accepted 3 June 2004; published online 20 Dec. 2004.*
*For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0173-0903.*

the P2P logical overlay network and physical underlying network. Because of the mismatch problem, the same message may traverse the same physical link multiple times, causing a large amount of unnecessary traffic. Second, a query may be flooded to multiple paths that are merged to the same peer. In this case, only the traffic along one of the paths is necessary. Finally, two neighboring peers may forward the same query message to each other before they receive the query message from each other. Thus, the same query message may traverse the same logical link twice.

Aiming at alleviating the mismatch problem, reducing the unnecessary traffic, and addressing the limits of existing solutions, we propose a location-aware topology matching (LTM) scheme. In LTM, each peer issues a detector in a small region so that the peers receiving the detector can record relative delay information. Based on the delay information, a receiver can detect and cut most of the inefficient and redundant logical links and add closer nodes as its direct neighbors. Our simulation studies show that the total traffic and response time of the queries can be significantly reduced by LTM without shrinking the search scope. We also show that the overhead of issuing detectors is trivial compared with the query cost savings.

Our proposed LTM can be used to complement other search techniques, such as forwarding-based search mechanisms [37] or cache-based schemes [14], [19], [20] that will be discussed in Section 2. In this paper, we will show this effectiveness by a case study of combining LTM and the response index caching scheme, in which query responses are cached in passing peers along the returning path. Our study shows that only one tenth of the original traffic cost is necessary to cover the same number of peers and the average response time is reduced by around 80 percent.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents the location-aware topology matching (LTM) scheme. Section 4 describes our simulation methodology. Performance evaluation of the LTM is presented in Section 5 and we conclude the work in Section 6.

## 2 Related Work

Many efforts have been made to avoid the large volume of unnecessary traffic incurred by the flooding-based search in decentralized unstructured P2P systems. In general, three types of approaches have been proposed to improve search efficiency in unstructured P2P systems: *forwarding-based, cache-based, and overlay optimization*. The three different kinds of approaches can be used together to complement each other.

In forwarding-based approaches, instead of relaying the query messages to all its logical neighbors except the incoming peer, a peer selects a subset of its neighbors to relay the query. In Directed BFS, proposed in [37], each peer maintains statistic information based on some metrics, such as the number of results received from neighbors for previous queries or the latency of the connection with that neighbor. A peer selects a subset of the neighbors, such as the neighbors that have returned the largest number of results for previous queries or the neighbors that have low latency, to send its query. A *k-walker* query algorithm is proposed in [18] in which a query is sent to $k$ different walkers (relay neighbors) from the source peer. For a peer in each walker, it just randomly selects one neighbor to relay the query. For each walker, the query processing is done sequentially. A hybrid periodical flooding (HPF) approach proposed in [39] improves the search efficiency by selecting forwarding neighbors based on multiple metrics and addressing the partial coverage problem to balance the search cost and response time.

The second approach is cache-based, including data index caching and content caching. Centralized P2P systems provide centralized index servers to keep indices of shared files of all peers. KaZaA utilizes cooperative superpeers, each of which is an index server of a subset of peers. Some systems distribute the function of keeping indices to all peers [20]. In Local Indices policy [37], each peer maintains an index of files available in the nodes within given hops of itself. When a peer receives a query, it can process the query on behalf of all nodes within the given hops of itself. Having observed the locality of queries, the authors in [19], [32] further propose that each peer cache query strings and results that flow through it. Three different strategies to replicate data (file content or query responses) on multiple peers have been evaluated in [14]. The three strategies are different on the ratio of allocations according to the ratio of query rates. Transparent query caching [22] is proposed to cache query hits at a gateway of an organization based on an observation of query locality in peers within the gateway. Caching file contents has also been studied. For example, an ideal cache (infinite capacity and no expiration) simulator [29] is built for KaZaA P2P traffic to cache file contents, which has shown that caching would have a large effect on a wide-scale P2P system on reducing traffic volume and bandwidth demands.

The third approach is based on overlay topology optimization that is closely related to what we are presenting in this paper. Here, we briefly introduce three types of solutions and their comparisons with our approach. End system multicast, Narada, is proposed in [13], which first constructs a rich connected graph on which to further construct shortest path spanning trees. Each tree rooted at the corresponding source using well-known routing algorithms. This approach introduces large overhead for forming the graph and trees in a large scope and does not consider the dynamic joining and leaving characteristics of peers. The overhead of Narada is proportional to the multicast group size. This approach is infeasible to large-scale P2P systems. Our proposed LTM is easy to implement and adaptive to the dynamic nature of P2P systems with the overhead that is only proportional to the square of the average number of neighbors. Researchers have also considered clustering close peers based on their IP addresses (e.g., [17], [21]). We believe there are two limitations for this approach. First, the mapping accuracy is not guaranteed by this approach. Second, this approach may affect the searching scope in P2P networks. In contrast, our technique is measurement-based and can accurately and dynamically connect the physically closer peers, and disconnect physically distant

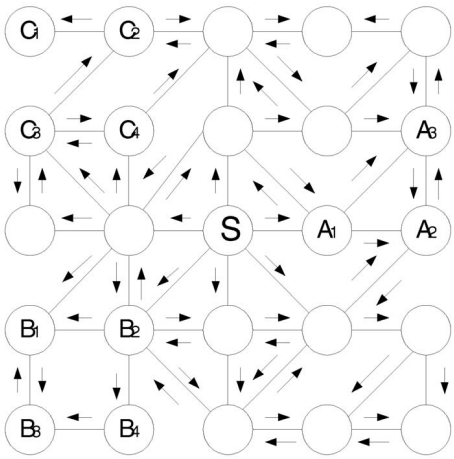


Fig. 1. An example of P2P overlay.

peers. Furthermore, our scheme does not shrink the search scope. Recently, researchers in [36] have proposed to measure the latency between each peer to multiple stable Internet servers called "landmarks." The measured latency is used to determine the distance between peers. This measurement is conducted in a global P2P domain and needs the support of additional landmarks. Similarly, this approach also affects the search scope in P2P systems. In contrast, our measurement is conducted in many small regions, significantly reducing the network traffic with high accuracy. Gia [12] introduced a topology adaptation algorithm to ensure that high capacity nodes are indeed the ones with high degree and low capacity nodes are within short reach of high capacity nodes. It addresses a different matching problem in overlay networks, but does not address the topology mismatch problem between the overlay and physical networks.

## 3 Location-Aware Topology Matching

In a P2P system, all participating peers form a P2P network over a physical network. A P2P network is an abstract, logical network called an overlay network. Maintaining and searching operations of a Gnutella peer are specifically described in [4]. When a new peer wants to join a P2P network, a boot-strapping node provides the IP addresses of a list of existing peers in the P2P network. The new peer then tries to connect with these peers. If some attempts succeed, the connected peers will be the new peer's neighbors. Once this peer connects into a P2P network, the new peer will periodically *ping* the network connections and obtain the IP addresses of some other peers in the network. These IP addresses are cached by this new peer. When a peer leaves the P2P network and then wants to join the P2P network again (no longer the first time), the peer will try to connect to the peers whose IP addresses have already been cached. The mechanism by which a peer joins a P2P network, the fact of a peer randomly joining and leaving, and the nature of flooding search make an inefficient mismatched overlay network and cause a large amount of unnecessary traffic. In this section, we first use examples to explain the unnecessary traffic and the mismatching problem. We then describe our proposed LTM technique in detail.

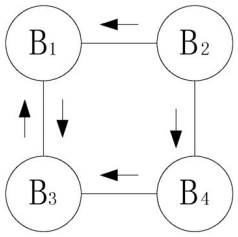


Fig. 2. Unnecessary traffic on logical link $B_1B_3$.

### 3.1 Unnecessary Message Duplications in Overlay Connections

Fig. 1 shows an example of a P2P overlay topology where solid lines denote overlay connections among logical P2P neighbors. Consider the case when node S sends a query. A solid arrow represents a delivery of the query message along one logical connection. The query is relayed by many peers, which incurs a lot of unnecessary traffic. Fig. 2 shows a subset of the P2P overlay in Fig. 1. In Fig. 2, after $B_2$ forwards the query to $B_1$, $B_4$ forwards to $B_3$. Since neither of $B_1$ or $B_3$ knows the other one has received the same query, they will forward the query to each other. It is clear that the search scope of the query from node S will not shrink without logical connection $B_1B_3$.

### 3.2 Topology Mismatch

Studies in [25] have shown that only 2 to 5 percent of Gnutella connections link peers within a single autonomous system (AS). But, more than 40 percent of all Gnutella peers are located within the top 10 ASes. This means that most Gnutella-generated traffic crosses AS borders so as to increase topology mismatch costs. Even if there is no message duplication in overlay connections, the same message can still traverse the same physical link multiple times due to mismatch problem, causing a large amount of unnecessary traffic.

For example, Fig. 3a is another subset of the P2P system of Fig. 1. Fig. 3a and Fig. 3b are two overlay topologies on top of the underlying physical topology shown in Fig. 3c. Suppose $C_1$ and $C_3$ are in the same autonomous system (AS), while $C_2$ and $C_4$ are in another AS. We assume that the physical link delay between $C_1$ and $C_4$ is much longer than all of the other links in Fig. 3c. Clearly, in the inefficient overlay of Fig. 3a, the query message from source S will traverse the longest link $C_1C_4$ four times, which is a scenario of topology mismatch. If we can construct an efficient overlay shown in Fig. 3b, the message needs to traverse all the physical links in Fig. 3c only once.

### 3.3 Three Main Operations of LTM

Optimizing inefficient overlay topologies can fundamentally improve P2P search efficiency. If the system can detect and disconnect the logical connections, such as $B_1B_3$ shown in Fig. 2, and switch the connections of $C_1C_2$, $C_2C_3$, and $C_3C_4$ shown in Fig. 3a to $C_3C_1$, $C_1C_4$, and $C_4C_2$ shown in Fig. 3b, the total network traffic could be significantly reduced without shrinking the search scope of queries. This

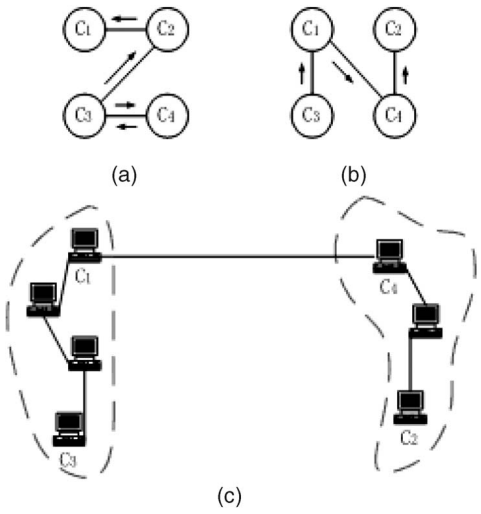


Fig. 3. Topology mismatch problem. (a) Inefficient overlay. (b) Efficient overlay. (c) Underlying physical topology.

is the basic principle of our proposed location-aware topology matching technique. Three operations are defined in LTM: TTL2 detector flooding, slow connection cutting, and source peer probing.

### 3.3.1 TTL2-Detector Flooding

Based on the Gnutella 0.6 P2P protocol [4], we design a new message type called TTL2-detector. In addition to the Gnutella's unified 23-byte header for all message types, a TTL2-detector message has a message body in two formats, as shown in Table 1. The short format is used in the source peer, which contains the source peer's IP address and the timestamp to flood the detector. The long format is used in a one-hop peer that is a direct neighbor of the source peer, which includes four fields: Source IP Address, Source Timestamp, TTL1 IP Address, and TTL1 Timestamp. The first two fields contain the source IP address and the source timestamp obtained from the source peer. The last two fields are the IP address of the source peer's direct neighbor, who forwards the detector, and the timestamp to forward it. In the message header, the initial TTL value is 2. The payload type of the detector can be defined as 0x82.

Each peer floods a TTL2-detector periodically. We use $d(i, S, v)$ to denote the TTL2-detector who has the message ID of $i$ with TTL value of $v$ and is initiated by $S$. We use $N(S)$ to denote the set of direct logical neighbors of $S$ and use $N^2(S)$ to denote the set of peers that are two hops away from $S$. A TTL2-detector can only reach peers in $N(S)$ and $N^2(S)$. We use network delay between two nodes as a metric for measuring the cost between nodes. The clocks in all peers can be synchronized by current techniques in an acceptable accuracy.[1] By using the TTL2-detector message, a peer can compute the cost of the paths to a source peer. As an example, in Fig. 4a, when peer $P$ receives a $d(i, S, 1)$, it can calculate the cost of link SP from Source Timestamp and the time P receives the $d(i, S, 1)$ from $S$. When $P$ receives a $d(i, S, 0)$, it can calculate the cost of link $\mathrm{SN}_1$ from TTL1 Timestamp and Source Timestamp, and $\mathrm{N}_1\mathrm{P}$ from TTL1 Timestamp and the time P receives the $d(i, S, 0)$ from $\mathrm{N}_1$. As we can see in an inefficient overlay topology, the peers in set $N^2(S)$ may receive $d(i, S, v)$ more than once, such as peer $P$ in Fig. 4a, Fig. 4b, and Fig. 4c. If a peer receives $d(i, S, v)$ multiple times, it will conduct the operations in the second step of LTM, slow connection cutting.

### 3.3.2 Slow Connection Cutting

There are three cases for any peer $P$ who receives $d(i, S, v)$ multiple times.

*Case 1*: $P$ receives both $d(i, S, 1)$ and $d(i, S, 0)$, as shown in Fig. 4a. In this case, $d(i, S, 1)$ comes from path SP, while $d(i, S, 0)$ comes from $\mathrm{SN}_1\mathrm{P}$. The costs of SP, $\mathrm{SN}_1$, and $\mathrm{N}_1\mathrm{P}$ can be calculated from the timestamps recorded in $d(i, S, 0)$ and $d(i, S, 1)$. If SP or $\mathrm{N}_1\mathrm{P}$ has the largest cost among the three connections, P will put this connection into its *will-cut* list that is a list of connections to be cut later. If $\mathrm{SN}_1$ has the largest cost, P will do nothing. Note that LTM is fully distributed and all peers do the same LTM operations. In the case of $\mathrm{SN}_1$ having the largest cost, $\mathrm{N}_1$ will put this connection into $\mathrm{N}_1$'s will-cut list. A peer will not send or forward queries to connections in its will-cut list, but these connections have not been cut in order for query responses to be delivered to the source peer along the inverse search path.

*Case 2*: P receives multiple $d(i, S, 0)$s from different paths, as shown in Fig. 4b. In LTM, P randomly takes two of the paths, such as $\mathrm{SN}_1\mathrm{P}$ and $\mathrm{SN}_2\mathrm{P}$ in Fig. 4b, to process at each time. Other paths, if any, will be handled in the next round of optimization. Thus, one important factor to affect the performance of LTM is the frequency for each peer to issue TTL2-detector messages. Our simulation results in Section 5 will show that the optimal LTM frequency is determined by the average peer lifetime and query frequency. Peer P can calculate the costs of $\mathrm{SN}_1$, $\mathrm{SN}_2$, $\mathrm{N}_1\mathrm{P}$, and $\mathrm{N}_2\mathrm{P}$. If $\mathrm{PN}_1$ or $\mathrm{PN}_2$ has the largest cost, P will put it into its will-cut list. If $\mathrm{SN}_1$ or $\mathrm{SN}_2$ has the largest cost, P will do nothing. As we have discussed above, $\mathrm{SN}_1$ or $\mathrm{SN}_2$ having the largest cost will be cut by one of the other three nodes.

*Case 3*: P receives one $d(i, S, 1)$ and multiple $d(i, S, 0)$s, as shown in Fig. 4c. In this case, P will process the path receiving $d(i, S, 1)$ and one path randomly selected from the multiple paths of $d(i, S, 0)$s, forming a scenario of Case 1.

A connection in a will-cut list will be disconnected when it has been in the list for a certain time period. Our simulation results in Section 5 will show that a period of 50 seconds is optimal in a system with the average number of neighbors of six, peer average lifetime of 10 minutes, and query frequency of 0.3 queries per peer per minute. If a connection is cut by $P$, the IP address of the other node in this connection and the cost of the connection will be recorded in $P$'s *cut list*. The cut list in a peer records the information of the connections disconnected by this peer and is designed to be used when the peer attempts to make a new connection so that the connections in the cut list will not be established again.

### 3.3.3 Source Peer Probing

For a peer $P$ who receives only one $d(i, S, 0)$ during a certain time period (e.g., 10 seconds), and $P \in (N^2(S) - N(S))$, it will

1. The current implementation of NTP version 4.1.1 in the public domain can reach synchronization accuracy down to 7.5 milliseconds [7]. Another approach is to use distance to measure the communication cost, such as the number of hops weighted by individual channel bandwidth.

TABLE 1
TTL2-Detector Message Body

| Byte offset | 0 | 3 | 4 | 7 |
|---|---|---|---|---|
| | Source IP Address | | Source Timestamp | |

| Byte offset | 0 | 3 | 4 | 7 | 8 | 11 | 12 | 15 |
|---|---|---|---|---|---|---|---|---|
| | Source IP Address | | Source Timestamp | | TTL1 IP Address | | TTL1 Timestamp | |

try to obtain the cost of PS by checking its cut list first. If $S$ is not in the list, $P$ will probe the distance to S (see Fig. 5). After obtain the cost of PS, $P$ will compare this cost with the costs of $\mathrm{SN}_1$ and $\mathrm{PN}_1$. If PS has the largest cost, $P$ will not keep this connection. Otherwise, this connection will be created. In the Internet, the cost of SP and the cost of PS may not be the same. We use the cost of PS to estimate the cost of SP.

Now, let's look back at the inefficient overlay topology shown in Fig. 3a. Fig. 6a, Fig. 6b, Fig. 6c, Fig. 6d, Fig. 6e, and Fig. 6f illustrate the process where LTM optimizes the overlay by using three operations discussed above.

### 3.4 Traffic Overhead of LTM

The simplicity of blind flooding makes it very popular in practice. This mechanism relays a query message to all its logical neighbors, except the incoming peer. For each query, each peer records the neighbors that relay the query to it. Therefore, in the worst case, the same query message can be sent on each link at most twice, as illustrated in Fig. 1. For an overlay network with $n$ peers, we use $c_n$ to denote the average number of neighbors and use $c_e$ to denote the average cost of the logical links. The total traffic caused by a query is less than or equal to $n\ c_n\ c_e$. In a typical P2P system, the value of $n$ (more than millions) is much greater than $c_n$ (less than tens) [31]. So, we can view both $c_n$ and $c_e$ as constant numbers. Thus, in the flooding-based search, the traffic incurred by one query from an arbitrary peer in a P2P network is $O(n)$. As observed in [32], each peer issues 0.3 queries per minute in average. Thus, the per minute traffic incurred by a P2P network with $n$ peers is $O(n^2)$.

Recall that each $d(i, S, v)$ has a TTL value of 2 in a source peer. So, the traffic for one-time LTM optimization in all peers is at most $2n\ c_n^2\ c_e$. If each peer conducts LTM $k$ times per minute, the total traffic is $2kn\ c_n^2\ c_e$. Our simulation results will show that the best value for $k$ is 2 or 3. Thus, the per minute traffic overhead incurred by LTM to the P2P network is $O(n)$.

Compared with the query traffic savings, the traffic overhead from LTM is trivial, which will be quantitatively shown in Section 5.3.

One question is why we do not use TTLj-detector with a TTL of $j > 2$ in a source peer so that cycles with more than four links can be detected and broken. There are two reasons for not doing so. First, if $j > 2$, the traffic caused by detector flooding will be increased significantly. Second, if the most expensive connection in a cycle is cut and its cost is not substantially larger than the costs of other links in the cycle, a query initiated from any of the two end peers in the broken cycle will need to traverse a path much more expensive than the cost on the cut connection to reach another end peer.

## 4 Simulation Methodology

We describe the three performance metrics we use in our simulations, our simulation setup, and parameter settings.

### 4.1 Performance Metrics

A well-designed search mechanism should seek to optimize both efficiency and Quality of Service (QoS). Efficiency focuses on better utilizing resources, such as bandwidth and processing power, while QoS focuses on user-perceived qualities, such as number of returned results and response time. In unstructured P2P systems, the QoS of a search mechanism generally depends on the number of peers

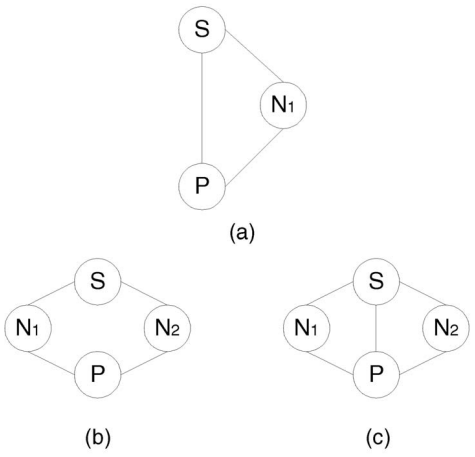


Fig. 4. Peer P receives d(i, S, v) multiple times.

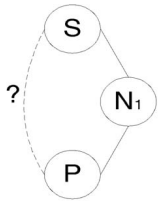


Fig. 5. Source peer probing.

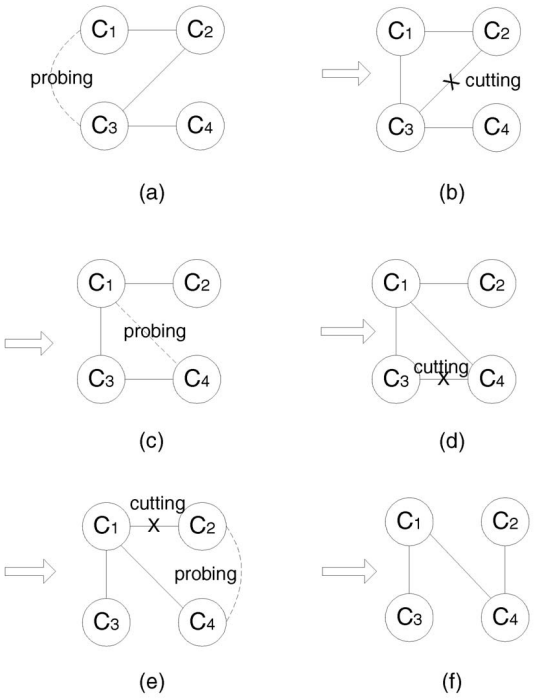


Fig. 6. An example of LTM.

being explored (queried), response time, and traffic overhead. If more peers can be queried by a certain query, it is more likely that the requested object can be found. So, we use three performance metrics: average traffic cost versus search scope, average neighbor distance, and query response time.

*Traffic cost* is one of the parameters with which network administrators are seriously concerned. Heavy network traffic limits the scalability of P2P networks [26] and is also a reason why a network administrator may prohibit P2P applications. We define the traffic cost as network resource used in an information search process of P2P systems, which is mainly a function of consumed network bandwidth and other related expenses. Specifically, in this work, we assume all the messages have the same length. When messages traverse an overlay connection during the given time period, the traffic cost ($Tc$) is given by: $Tc = M \times L$, where $M$ is the number of messages that traverse the overlay connection and $L$ represents the number of physical links in this overlay connection.

*Search scope* is defined as the number of peers that queries have reached in an information search process. Thus, with the same traffic cost, we aim to maximize the search scope, while, with the same search scope, we aim to minimize the traffic cost.

*Average neighbor distance* ($D$) is used to evaluate the optimization results of a logical topology. Let $D_i$ be the average delay between the source peer $i$ and all its logical neighbors. The value $D$ is defined as the average of all $D_i$s (i.e., all peers in the P2P network). Minimizing average neighbor distance implies a better matching with the underlying physical network.

The *response time* of a query is one of the parameters with which P2P users are concerned. We define the response time of a query as the time period from when the query is issued until when the source peer receives a response result from the first responder. In P2P systems, each query normally receives more than one response from different peers. To better measure the search quality, we define search latency as the time period from when the query is issued until when the source peer receives response results from 50 percent of all the content holders in the system.

### 4.2 Simulation Setup

To evaluate the effectiveness of LTM, we first generate network topologies. Based on generated networks, we simulate P2P flooding search, host joining/leaving behavior, LTM, and index caching.

#### 4.2.1 Topology Generation

Two types of topologies, physical topology and logical topology, are generated in our simulation. The physical topology should represent the real topology with Internet characteristics. The logical topology represents the overlay P2P topology built on top of the physical topology. All P2P nodes are in a subset of nodes in the physical topology. The communication cost between two logical neighbors is calculated based on the physical shortest path between this pair of nodes. To simulate the performance of different search mechanisms in a more realistic environment, the two topologies must accurately reflect the topological properties of real networks in each layer.

Previous studies have shown that both large scale Internet physical topologies [34] and P2P overlay topologies [28] follow the small world and power law properties. Power law describes the node degree while the small world describes characteristics of path length and clustering coefficient [11]. The study in [28] found that the topologies generated using the AS Model have the properties of the small world and power law. BRITE [1] is a topology generation tool that provides the option of generating topologies based on the AS Model. Using BRITE, we generate five physical topologies each with 22,000 nodes. The logical topologies are generated with the number of peers (nodes) ranging from 2,000 to 8,000. The average number of neighbors of each node ranges from 4 to 10.

We simulate LTM for all the generated logical topologies on top of each of the five generated physical topologies.

#### 4.2.2 Flooding Search Simulation

Our simulation is based on observed distributions as follows: The content popularity of a publisher follows a Zipf-like distribution (aka Power Law) [8], [10], where the relative probability of a request for the $i$th most popular page is proportional to $1/i^{\alpha}$, with $\alpha$ typically taking on some value less than unity. The observed value of the exponent varies from trace to trace. The request distribution does not follow the strict Zipf's law (for which $\alpha = 1$), but, instead, does follow a more general Zipf-like distribution. Query word frequency does not follow a Zipf distribution [16], [35]. The user's query lexicon size does not follow a Zipf distribution [35], but with a heavy tail. Both the overall traffic and the traffic from 10 percent of the popular nodes are heavy-tailed in terms of the host connectivity, traffic volume, and average bandwidth of the hosts [31]. Studies in [30] have suggested a

log-quadratic distribution ($10^{-\alpha_2}$) for stored file locality and transfer file locality. The time length that nodes remain available follows a log-quadratic curve [30], which could be approximated by two Zipf distributions.

In our simulation, we simulate the flooding search used in the Gnutella network by conducting the Breath First Search algorithm from a specific node. A search operation is simulated by randomly choosing a peer as the sender, and a keyword according to Zipf distribution. In our first simulation, 100,000 search operations are simulated sequentially.

#### 4.2.3 A Dynamic P2P Environment

P2P networks are highly dynamic, with peers joining and leaving frequently. The observations in [31] have shown that over 20 percent of the logical connections in a P2P last 1 minute or less and around 60 percent of the IP addresses keep active in FastTrack for no more than 10 minutes each time after they join the system. The measurement reported in [28] indicated that the median up-time for a node in Gnutella and Napster is 60 minutes. Studies in [9] have argued that measurement according to host IP addresses underestimates peer-to-peer host availability and have shown that each host joins and leaves a P2P system 6.4 times a day on average and more than 20 percent of the hosts arrive and depart every day. Although the numbers they provided are different to some extent, they share the same point that the peer population is quite transient. We simulate the joining and leaving behavior of peers via turning on/off logical peers. In our simulation, every node issues 0.3 queries per minute, which is calculated from the observation data shown in [32], i.e., 12,805 unique IP addresses issued 1,146,782 queries in 5 hours. When a peer joins, a lifetime in seconds will be assigned to the peer. The lifetime of a peer is defined as the time period the peer will stay in the system. The lifetime is generated according to the distribution observed in [28]. The mean of the distribution is chosen to be 10 minutes [31]. The value of the variance is chosen to be half of the value of the mean. The lifetime will be decreased by one after passing each second. A peer will leave in the next second when its lifetime reaches zero. During each second, there are a number of peers leaving the system. We then randomly pick up (turn on) the same number of peers from the physical network to join the overlay.

#### 4.2.4 Combining LTM with Index Cache

To investigate whether LTM could be employed together with other approaches, we also simulate a strategy of combining LTM with a response index caching scheme in which query responses are cached in passing peers along the returning path. In our simulation, each peer keeps a local cache and a response index cache. The size of a response index cache is bounded by 200 items. The average number of neighbors is six.

## 5 Performance Evaluation

We present our simulation results in this section. Our simulation results on overlay networks of 2,000 nodes, 3,000 nodes, 5,000 nodes, and 8,000 nodes on top of a 22,000-node Internet-like physical networks are consistent.

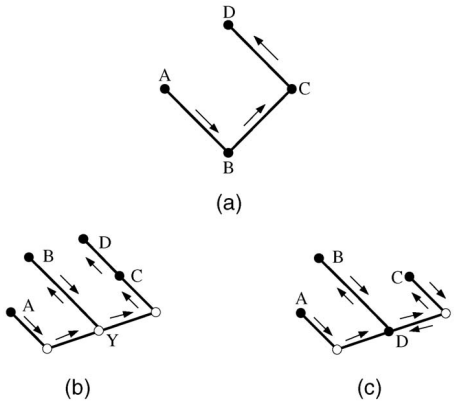


Fig. 7. Topology mismatch problem.

Due to page limitations, we only present the results based on the overlay network with 8,000 nodes.

### 5.1 The Amount of Mismatching

We first quantitatively evaluate how serious the topology mismatch problem is in Gnutella-like networks. We generate 1,000,000 queries on different topologies with an average number of neighbors being 4, 6, 8, and 10, respectively. We track the response of each query message to see if the response path is a mismatching path. We count a path as a mismatching path if a peering node in the path has been visited more than once. Fig. 7 gives an example to show the rational of our measurement. Fig. 7b and Fig. 7c illustrate two different underlying physical network topologies of the overlay shown in Fig. 7a. Only nodes A, B, C, and D are peering nodes. Node Y is not a peering node. We can see that, in Fig. 7c, the overlay path of $A \rightarrow B \rightarrow C \rightarrow D$ traverses physical link $BD$ twice and the peering node $D$ is visited three times. In Fig. 7b, node Y is also visited more than once, but we do not count the overlay path of $A \rightarrow B \rightarrow C \rightarrow D$ as a mismatching path. This is because a revisit to Y cannot be avoided since Y is not a peering node. Our results show that about 75 percent of the paths suffer from the topology mismatch problem. Although the mismatch problem is a little less serious when the average number of neighbors increases, the mismatching degree is not very sensitive to the average number of neighbors when the number is changed from 4 to 10. However, we expect the mismatching degree can be significantly reduced if the average number of neighbors increases significantly. The extremely large average number of neighbors is not realistic in existing P2P networks.

### 5.2 Effectiveness of LTM in Static Environment

In our first simulation, we study the effectiveness of LTM in a static P2P environment where the peers do not join and leave frequently. This will show, without changing the overlay topology, how many LTM optimization steps are required to reach a better topology matching.

#### 5.2.1 Traffic Cost versus Search Scope

The goal of the LTM scheme is to reduce traffic cost as much as possible while retaining the same search scope. Fig. 8 compares the traffic cost incurred by the original Gnutella-like system and by the system after one-step

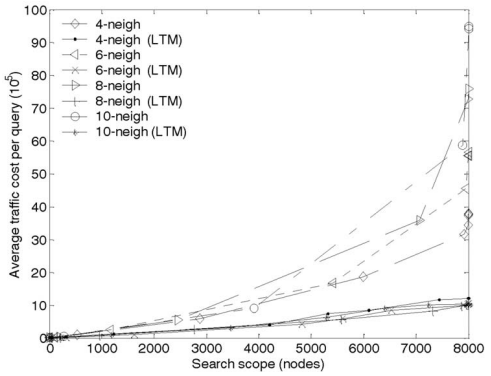


Fig. 8. Traffic cost versus search scope.

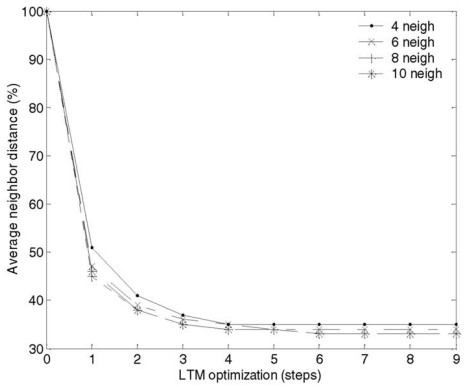


Fig. 10. Average neighbor distance versus optimal step,

LTM optimization. One-step means every peer makes LTM optimization only once. Since this simulation is based on a static P2P environment, we do not include traffic cost incurred by LTM operations. The traffic overhead of LTM is included by simulation results in dynamic environments shown in Fig. 15.

In Fig. 8, the curve of "$c_n$-$neigh$" shows the average traffic cost caused by a query to cover the search scope in the x-axis, where, in the system, the average number of logical neighbors is $c_n$. The dashed curves represent performance results without using LTM, while solid curves represent the results with LTM optimizations. Fig. 8 shows that, to cover the same search scope, one-step LTM reduces the traffic cost significantly and the reduction rate increases as the search scope increases. In other words, with a given traffic cost, LTM will increases its search scope. Fig. 9 shows that the traffic cost decreases when LTM is conducted multiple times, where the search scope is all 8,000 peers. We can see that the traffic cost reduction reaches a threshold after the second or third step LTM optimization. LTM can be convergent *as fast as in two to three steps*.

#### 5.2.2 Average Neighbor Distance, Response Time, and Search Latency

Average neighbor distance reflects the effectiveness of LTM on topology match problem. Fig. 10 shows the average neighbor distance versus LTM optimization steps.

Compared with the original Gnutella-like network without LTM scheme (zero optimization steps), one-step LTM optimization reduces average neighbor distance by about 55 percent and more steps of LTM may cut average neighbor distance to around 65 percent.

Short query response time is always desirable in P2P systems. The simulation results in Fig. 11 and Fig. 12 show that LTM can effectively shorten the query response time and search latency by about 62 percent and 55 percent, respectively. The trade off between query traffic cost and response time has been discussed in [39]. P2P systems with a large number of average connections offer a faster search speed while increasing traffic. One of the strengths of the LTM scheme is that it reduces both query traffic cost and response time without decreasing the query success rate.

Our other simulation results also show that different densities of logical peers or physical nodes will not impact the effectiveness of LTM. The average traffic cost is only proportional to the average number of neighbors and average cost logical links, which is consistent with previous analysis.

### 5.3 LTM in Dynamic Environment

We further evaluate the effectiveness of LTM in dynamic P2P systems and explore the best frequency for each peer to conduct LTM. We first discuss the performance impact of the will-cut list and the cut list. The average number of logical neighbors we use is six.

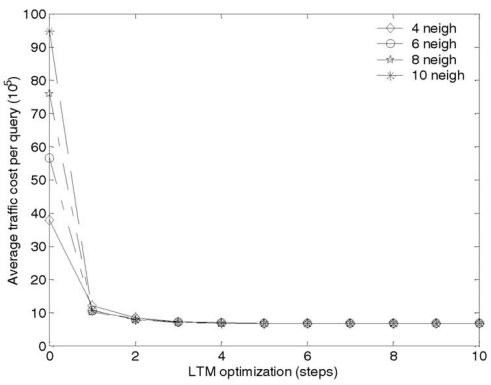


Fig. 9. Traffic reduction versus optimization step.

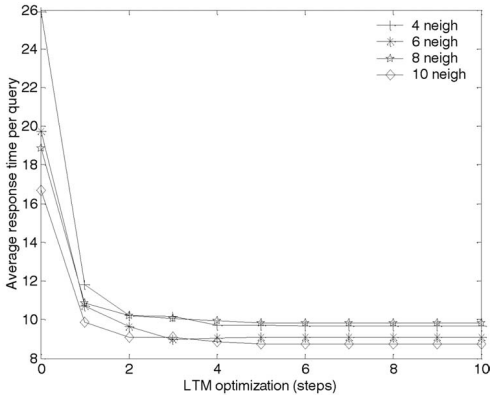


Fig. 11. Average response time versus optimal step.

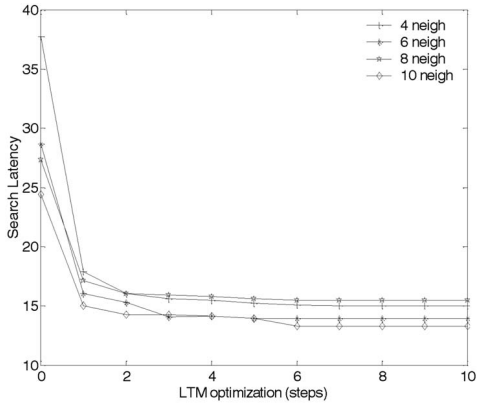


Fig. 12. Average search latency versus optimal step.

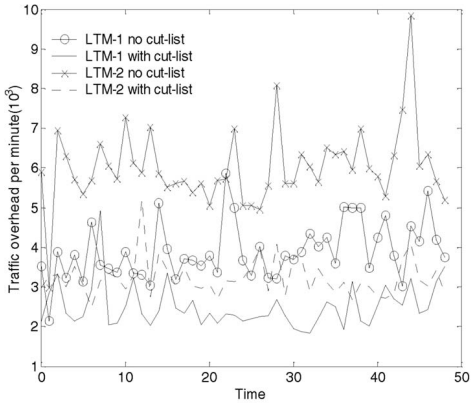


Fig. 14. Effectiveness of cut list.

#### 5.3.1 Effectiveness of Will-Cut List (W-C)

From our simulation results in dynamic environments, we found that, with the same search scope, the query success rate in dynamic environments is decreased by about 5 percent compared with the static environment, as shown in Fig. 13 (compare curves of static *Gnutella-like* and *dynamic Gnutella-like*). One extreme case is when the search scope is 100 percent, which means that each query can reach all peers and we guarantee the query result is available in at least one of the peers. The search success rate is expected to be 100 percent in this case, but it is only 95 percent. The reason for the 5 percent loss in query success rate is that the query responses cannot be returned due to peers' dynamic leaving behavior. We call this phenomena the ***response loss problem***.

If we do not use the will-cut list in LTM, a connection will be cut immediately when it is found to be a slow connection, which will cause a very serious response loss problem because many responses may not be returned due to the cut connections. The curve of *LTM without W-C* in Fig. 13 shows that the query success rate is significantly decreased by 30-40 percent without using the will-cut list. The LTM is conducted once every minute in this simulation. Retaining the query success rate is the reason we design the will-cut list, each of which can hold 20 connections in our simulation. The up to 20 slow connections will not be used to forward queries, but only used to return query results. The lifetime of the connections in a will-cut list determines the query success rate. In Fig. 13, a curve of *LTM with W-C-n* means the lifetime of a will-cut connection is $n$ seconds. We can see that the query success rate can be retained if the connections can be kept in the will-cut list for 50 seconds.

#### 5.3.2 Effectiveness of Cut List

If we do not use the cut list, a connection that has just been cut may be established again. Thus, the LTM optimization rate will be limited. Fig. 14 compares the overhead incurred by LTM with and without the use of the cut list. The fluctuations of the curves represent the dynamic nature of the network as time goes. The curve of *LTM-k* means each peer conducts LTM for $k$ times per minute. We can see that the use of the cut list reduces traffic overhead by about 50 percent compared with the case without using the cut list.

#### 5.3.3 Effectiveness and Frequency of LTM

We use the will-cut list and the cut list in this part of simulation. Compared with a Gnutella-like system, Fig. 15 and Fig. 16 show the effectiveness of LTM on reducing average traffic cost and query response time. Since LTM adds some traffic overhead due to the TTL2 detector flooding, there exists an optimal frequency for each peer to conduct LTM independently. We simulate LTM in different frequencies ranging from 1/4 to 4 times every minute. We consider a frequency to be optimal if the next higher frequency does not increase the optimization by more than 3 percent compared with the current frequency.

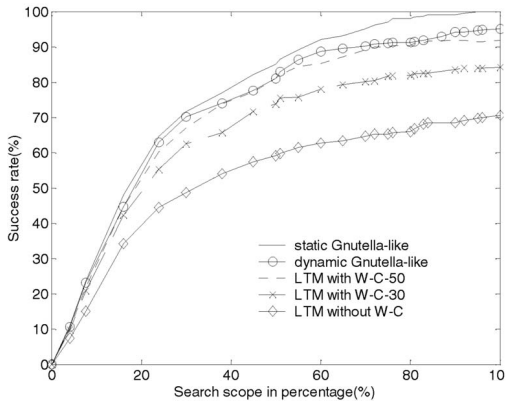


Fig. 13. Effectiveness of will-cut list.

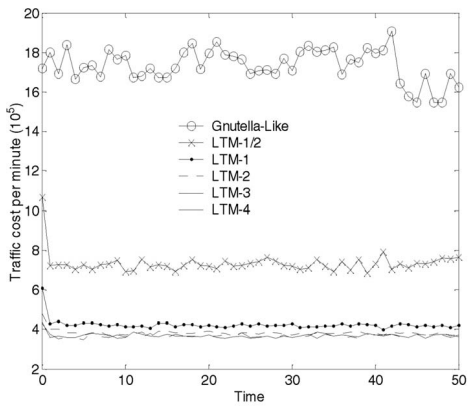


Fig. 15. Total traffic versus LTM frequency.

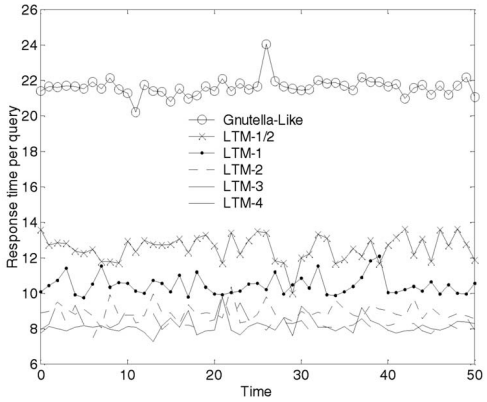


Fig. 16. Response time versus LTM frequency.

Results in Fig. 15 and Fig. 16 show that, under the assumption that peer average lifetime in a P2P system is 10 minutes and 0.3 queries are issued by each peer per minute, the optimal frequency for every peer to conduct LTM is twice per minute. With this frequency, about 75 percent reduction on traffic cost and 65 percent reduction on response time can be achieved.

As we have mentioned, different values of peer average lifetime and query frequency have been presented by previous studies [9], [28], [31], [32]. We further tune the two parameters (average lifetime and query frequency) in our simulation. Fig. 17 shows that LTM can be conducted less frequently if peer average lifetime is longer. Fig. 18 shows that LTM should be conducted more frequently if more queries are issued. Both figures show that a larger average number of neighbors require a higher LTM frequency.

### 5.4 Combining LTM and Query Index Caching

In this part, we evaluate a strategy of combining LTM with response index caching scheme. We compare the traffic cost and response time in a Gnutella-like system without any optimization, with query index caching only, with one-step LTM optimization only, and with one-step LTM optimization plus query index caching. Results in Fig. 19 and Fig. 20 show that, by combining LTM and query index, caching the traffic cost is reduced by about 10 times without shrinking the search scope and the average query response time is reduced by about seven times.

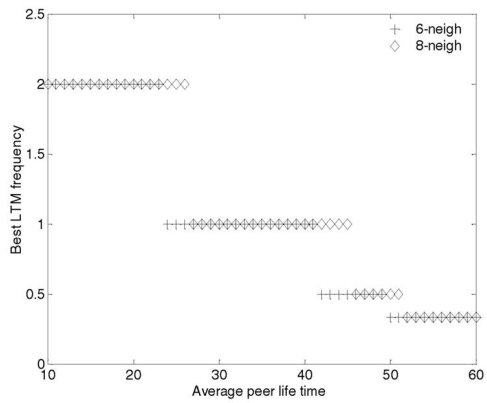


Fig. 17. Optimal LTM frequency versus average peer lifetime.

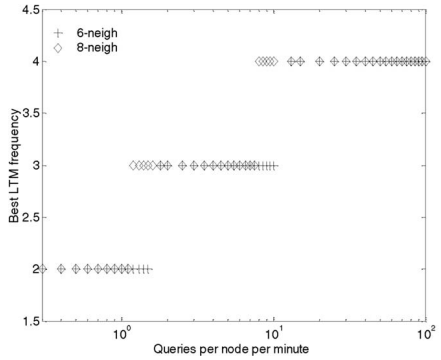


Fig. 18. Optimal LTM frequency versus average query frequency.

## 6 Conclusion and Future Work

We have evaluated our proposed location-aware topology matching, LTM, in both static and dynamic environments. Simulations in static P2P environments show that the significant performance benefit of LTM is consistent with various network sizes and average numbers of neighbors. In simulation studies of dynamic environments, we have investigated the optimal LTM frequency in a more realistic P2P environment. The results show that LTM achieves about 75 percent reduction on traffic cost and about 65 percent reduction on query response time. The impacts of peer average lifetime and query frequency on optimal LTM

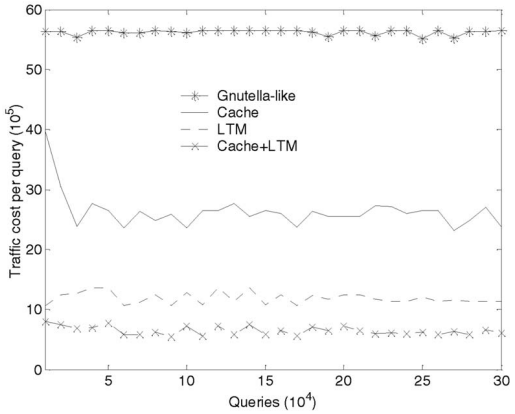


Fig. 19. Traffic cost of four schemes.

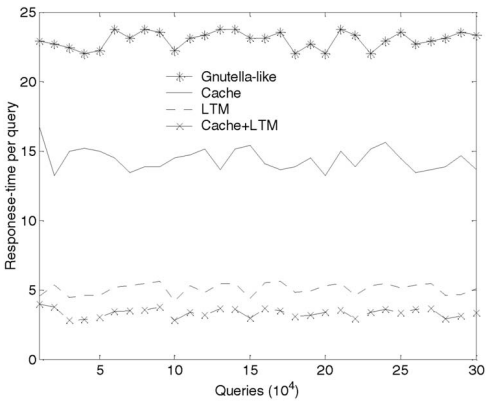


Fig. 20. Average response time of four schemes.

frequency have also been studied. We also show that our design of the will-cut list and the cut list can improve the performance of LTM. The ability that LTM can complement other advanced search approaches has been shown by a combination strategy of LTM with query index caching.

In LTM, each peer is aware of the location of other peers within a distance of two hops. The major advantage of LTM is that it is not only able to timely match the logical topology with the physical topology to significantly improve the search efficiency; it also guarantees to retain the search scope. This contribution makes LTM truly unique and highly effective. In addition, LTM is completely decentralized and scalable.

## ACKNOWLEDGMENTS

This work is supported in part by the US National Science Foundation under grants CNS-0098055, CCF-0129883, and CCF-0325760, by Michigan State University IRGP Grant 41114, and by Hong Kong RGC Grant HKUST6264/04E. Some preliminary results of this work were presented at the INFOCOM 2004 Conference.

## REFERENCES

[1] BRITE, http://www.cs.bu.edu/brite/, 2003.
[2] Fasttrack, http://www.fasttrack.nu, 2003.
[3] Gnutella, http://gnutella.wego.com/, 2003.
[4] The Gnutella protocol specification 0.6, http://rfc-gnutella.sourceforge.net, 2003.
[5] KaZaA, http://www.kazaa.com, 2003.
[6] Napster, http://www.napster.com, 2003.
[7] NTP: The Network Time Protocol, http://www.ntp.org/, 2003.
[8] V. Almeida, A. Bestavros, M. Crovella, and A.D. Olivera, "Characterizing Reference Locality in the WWW," *Proc. IEEE Conf. Parallel and Distributed Information Systems (PDIS),* 1996.
[9] R. Bhagwan, S. Savage, and G.M. Voelker, "Understanding Availability," *Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '03),* 2003.
[10] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM,* 1999.
[11] T. Bu and D. Towsley, "On Distinguishing between Internet Power Law Topology Generators," *Proc. IEEE INFOCOM,* 2002.
[12] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM,* 2003.
[13] Y. Chu, S.G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM SIGMETRICS,* 2000.
[14] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," *Proc. ACM SIGCOMM,* 2002.
[15] O.D. Gnawali, "A Keyword-Set Search System for Peer-to-Peer Networks," master's thesis, Massachusetts Inst. of Technology, June 2002.
[16] B.J. Jansen, A. Spink, J. Bateman, and T. Saracevic, "Real Life Information Retrieval: A Study of User Queries on the Web," *Proc. SIGIR Forum,* 1998.
[17] B. Krishnamurthy and J. Wang, "Topology Modeling via Cluster Graphs," *Proc. SIGCOMM Internet Measurement Workshop,* 2001.
[18] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," *Proc. 16th ACM Int'l Conf. Supercomputing,* 2002.
[19] E.P. Markatos, "Tracing A Large-Scale Peer-to-Peer System: An Hour in the Life of Gnutella," *Proc. Second IEEE/ACM Int'l Symp. Cluster Computing and the Grid,* 2002.
[20] D.A. Menasce and L. Kanchanapalli, "Probabilistic Scalable P2P Resource Location Services," *ACM SIGMETRICS Performance Evaluation Rev.,* vol. 30, pp. 48-58, 2002.
[21] V.N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," *Proc. ACM SIGCOMM,* 2001.
[22] S. Patro and Y.C. Hu, "Transparent Query Caching in Peer-to-Peer Overlay Networks," *Proc. 17th Int'l Parallel and Distributed Processing Symp. (IPDPS),* 2003.
[23] L. Peterson, D. Culler, T. Anderson, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," *Proc. HOTNETS,* 2002.
[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM,* 2001.
[25] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing,* 2002.
[26] J. Ritter, "Why Gnutella Can't Scale. No, Really," http://www.tch.org/gnutella.html, 2001.
[27] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. Int'l Conf. Distributed Systems Platforms,* 2001.
[28] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking (MMCN),* 2002.
[29] S. Saroiu, K.P. Gummadi, R.J. Dunn, S.D. Gribble, and H.M. Levy, "An Analysis of Internet Content Delivery Systems," *Proc. Fifth Symp. Operating Systems Design and Implementation,* 2002.
[30] M.T. Schlosser and S.D. Kamvar, "Availability and Locality Measurements of Peer-to-Peer File Systems," *Proc. ITCom: Scalability and Traffic Control in IP Networks,* 2002.
[31] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. ACM SIGCOMM Internet Measurement Workshop,* 2002.
[32] K. Sripanidkulchai, "The Popularity of Gnutella Queries and Its Implications on Scalability," http://www2.cs.cmu.edu/kunwadee/research/p2p/gnutella. html, 2001.
[33] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM,* 2001.
[34] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based vs. Structural," *Proc. SIGCOMM '02,* 2002.
[35] Y. Xie and D. O'Hallaron, "Locality in Search Engine Queries and Its Implications for Caching," *Proc. IEEE INFOCOM,* 2002.
[36] Z. Xu, C. Tang, and Z. Zhang, "Building Topology-Aware Overlays Using Global Soft-state," *Proc. 23rd Int'l Conf. Distributed Computing Systems (ICDCS),* 2003.
[37] B. Yang and H. Garcia-Molina, "Efficient Search in Peer-to-Peer Networks," *Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS),* 2002.
[38] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing," Technical Report UCB//CSD-01-1141, Univ. of California, Berkeley, 2001.
[39] Z. Zhuang, Y. Liu, L. Xiao, and L.M. Ni, "Hybrid Periodical Flooding in Unstructured Peer-to-Peer Networks," *Proc. Int'l Conf. Parallel Processing,* 2003.

**Yunhao Liu** received the BS degree in automation from Tsinghua University, China, in 1995, the MA degree from Beijing Foreign Studies University, China, in 1997, and the PhD degree in computer science from Michigan State University in 2004. He is now an assistant professor in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests are in the areas of peer-to-peer computing, pervasive computing, distributed systems, network security, embedded systems, and high-speed networking. He is a member of the IEEE and the IEEE Computer Society.

**Li Xiao** received the BS and MS degrees in computer science from Northwestern Polytechnic University, China, and the PhD degree in computer science from the College of William and Mary in 2002. She is an assistant professor of computer science and engineering at Michigan State University. Her research interests are in the areas of distributed and Internet systems, system resource management, and design and implementation of experimental algorithms. She is a member of the ACM, the IEEE, and the IEEE Computer Society.

**Xiaomei Liu** received the BS degree in electronics and information system technology from East China Normal University in 1996 and the MS degree in computer engineering from University of Toledo in 2000. She is currently a PhD student at Michigan State University. Her research interests include distributed operating systems and computer network.

**Lionel M. Ni** received the PhD degree in electrical and computer engineering from Purdue University, West Lafayette, Indiana, in 1980. He is a professor and head of the Computer Science Department of the Hong Kong University of Science and Technology. He was a professor of computer science and engineering at Michigan State University from 1981 to 2003, where he received the Distinguished Faculty Award in 1994. His research interests include parallel architectures, distributed systems, high-speed networks, and pervasive computing. A fellow of the IEEE and the IEEE Computer Society, he has chaired many professional conferences and has received a number of awards for authoring outstanding papers.

**Xiaodong Zhang** received the BS degree in electrical engineering from Beijing Polytechnic University in 1982 and the MS and PhD degrees in computer science from the University of Colorado at Boulder in 1985 and 1989, respectively. He is the Lettie Pate Evans Professor of Computer Science and the Department Chair at the College of William and Mary. He was the Program Director of Advanced Computational Research at the US National Science Foundation from 2001 to 2003. He is a past member of the Editorial Board of the *IEEE Transactions on Parallel and Distributed Systems* and currently serves as an associate editor of *IEEE Micro*. His research interests are in the areas of parallel and distributed computing and systems and computer architecture. He is a senior member of the IEEE and the IEEE Computer Society.