

# A Two-Hop Solution to Solving Topology Mismatch

Yunhao Liu, *Senior Member, IEEE*

**Abstract**—The efficiency of Peer-to-Peer (P2P) systems is largely dependent on the overlay constructions. Due to the random selection of logical neighbors, there often exists serious topology mismatch problem between overlay and physical topologies in P2P systems. Such mismatching causes unnecessary query message duplications at both the overlay and IP level, as well as an increase in query response time. In this work, we define the optimal overlay problem and prove its NP-hardness. We then propose a distributed overlay optimization algorithm to address this issue and evaluate its effectiveness through trace-driven simulations. The proposed design has four strengths. First, it does not need any global knowledge. Second, its optimization convergent speed is fast. Third, it is orthogonal to other types of advanced search approaches. Fourth, it reduces both the traffic cost and the search latency.

**Index Terms**—Peer-to-Peer, topology mismatch, optimal overlay, NP-hard, distributed approach.

## 1 INTRODUCTION

PEER-TO-PEER (P2P) networking dictates a fully distributed cooperative network design and has recently gained significant acceptance [8], [17], [19], [24]. Most of the popular P2P applications, such as Gnutella, KaZaA, and Bit Torrent, where peers voluntarily provide data of interest as well as receive it, operate on decentralized and unstructured networks. In such systems, peers connect in an ad hoc fashion, and the placement of documents is not controlled.

In a P2P system, all participating peers form an overlay on top of an underlying physical network. When a new peer joins a P2P network, a bootstrapping node provides IP addresses of a list of existing peers in the system. The new peer then tries to connect with some of those peers. If some attempts succeed, the connected peers will be the new peer's neighbors.

Recent studies [20] have shown that P2Ps contribute the largest portion of Internet traffic, based on measurements on popular P2P systems, such as FastTrack, Gnutella, KaZaA, and DirectConnect [1]. The inefficient P2P overlay topology is one of the major reasons for the heavy P2P traffic [18]. For instance, a peer randomly choosing logical neighbors without any knowledge about the underlying physical topology causes topology mismatch between the overlay network and the underlying network.

Fig. 1 plots an example. For a query message sent along the overlay path  $A \rightarrow C \rightarrow B$ , due to the mismatch problem, the same message may traverse the same physical links, such as  $HE$ ,  $EF$ , and  $FC$  multiple times, causing a large amount of unnecessary traffic. Obviously, the query

search latency is also increased largely, which is undesired for users.

To quantitatively evaluate how serious the topology mismatch is, we simulate 1,000,000 queries on different P2P topologies based on real traces. Detailed simulation methodology will be discussed in Section 5. In the simulation, we track the response of each query message to check if the response comes back along a mismatched path. The results show that more than 70 percent in flat Gnutella-like topologies and around 60 percent in super-peer topologies of the query paths suffer from mismatch.

Existing overlay topology optimization studies, which use different techniques to identify physically closer nodes to connect as overlay neighbors, could significantly limit the number of peers a query reaches, or called search scope. We define *search scope* as the number of peers that a query can reach in an information search process. Hence, most of the previous designs are not feasible in unstructured P2P systems. Aiming at alleviating mismatch, reducing the unnecessary message duplications as well as query response time, we investigate the relationship among query traffic cost, message duplications, average query response time, and P2P overlay topologies. Our study shows that there exists a tradeoff between query traffic cost and its average response time, so that it is impossible to find an overlay with both minimum traffic cost and minimum average response time. We prove that given a physical topology and a subset of all the physical nodes as peering nodes, even if we have global knowledge, finding an optimal overlay is still NP-hard.

To address the limits of the existing solutions, we propose a distributed heuristic, called Two-Hop-Away Neighbor Comparison and Selection (THANCS). The overhead of THANCS is trivial compared with the query cost savings. We show the effectiveness of this design by contrasting the performance of search efficiency on a THANCS optimized overlay and an optimal overlay obtained by an offline brute force algorithm. We also show

• The author is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: liu@cse.ust.hk.

Manuscript received 1 May 2007; revised 1 Oct. 2007; accepted 14 Jan. 2008; published online 1 Feb. 2008.

Recommended for acceptance by C. Shahabi.

For information on obtaining reprints of this article, please send e-mail to: [tpds@computer.org](mailto:tpds@computer.org), and reference IEEECS Log Number TPDS-2007-05-0135. Digital Object Identifier no. 10.1109/TPDS.2008.24.

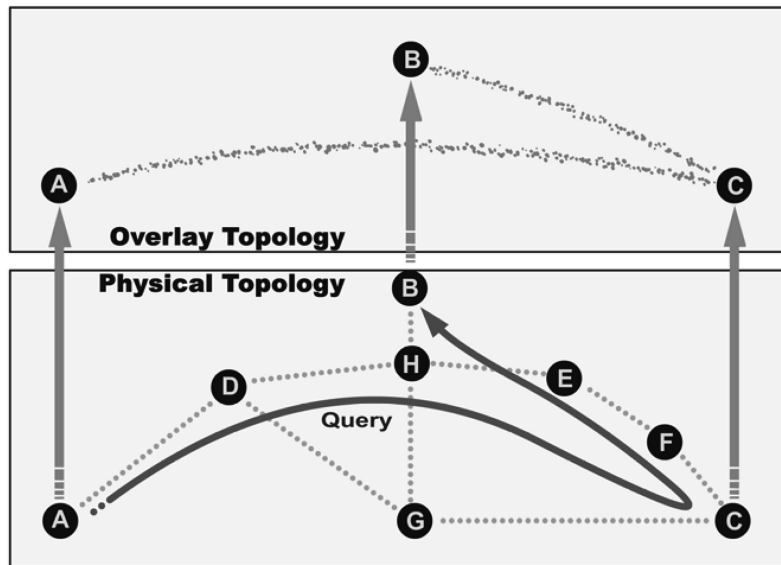


Fig. 1. An example of topology mismatch.

that THANCS is orthogonal to most of the existing search enhancement approaches.

The rest of this paper is organized as follows: Section 2 discusses the related work. Section 3 discusses the optimal overlay problem: Degree-bounded Minimum Average Distance (DMAD) problem. Section 4 presents the THANCS approach. We comprehensively study the performance of THANCS in Section 5 and conclude this work in Section 6.

## 2 RELATED WORK

Different techniques have been employed to build efficient overlay topologies. End system multicast, Narada [6], forms a rich connected graph on which shortest path spanning trees are constructed. Researchers have also considered clustering close by peers based on their IP addresses [9], [16]. There are two limitations in such approaches. First, the mapping accuracy is not guaranteed. Second, they might affect the search scope of a P2P network. In contrast, our technique is measurement based. It dynamically connects physically closer peers and disconnects distant ones without shrinking the search scope. Gia [5] introduces a topology adaptation algorithm to ensure that high capacity nodes are 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.

To attack topology mismatch, minimum spanning tree (MST)-based approaches have been proposed [14], [25], in which peers build an overlay MST among the source node and certain hop neighbors, and then optimizes connections that are not on the tree. An early attempt at alleviating topology mismatch is called LTM [12], in which each peer issues a detector message 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, as well as add closer nodes as direct neighbors. The major drawback of LTM is it needs to synchronize all peering nodes and thus requires the support of NTP [3],

which is critical. We will show that the design of this work does not need NTP and has similar convergent speed as LTM, faster than MST-based approaches.

## 3 OPTIMAL OVERLAY PROBLEM

We model a P2P network based on the following assumptions. First, an overlay connection between a pair of peering nodes consists of a number of physical links, which form a shortest path between the pair of end nodes in the physical topology, and Internet paths are relatively stable [26]. Second, we assume that the same size packets traversing the same physical link in a short period of time will have similar delay.

### 3.1 Modeling P2P Networks

We model a communication network by an undirected graph  $G = (V, E)$ , where the vertex set  $V$  represents units such as hosts and routers, and the edge set  $E$  represents physical links connecting pairs of communicating unit. Given an undirected graph  $G = (V, E)$  modeling an interconnection network, for instance, the whole or part of the Internet, and a subset  $X \subseteq V(G)$  of communicating units (peers), we construct a complete edge weighted graph  $D = (V, E)$ , where  $V(D) = X$ , and the weight of each  $uv \in E(D)$  is equal to the length of a shortest path between peer  $u$  and peer  $v$  in  $G$ . Note that  $D$  is a complete graph, that is, it includes all possible edges, and is referred to as the *distance graph* of  $G$ .

We start with a physical network,  $G$ , and then choose a set of communicating peers,  $X$ . The resulting distance graph,  $D$ , constructed as mentioned earlier, is the basis for constructing a P2P overlay graph  $H = (V, E)$ , which is done as follows: The vertex set  $V(H)$  will be the same as  $V(D)$ , and edge set  $E(H) \subseteq D(G)$ . The key issue here is how to select  $E(H)$ . For the remainder of this paper, we will consistently refer to the physical graph by  $G$ , its distance graph by  $D$ , and an overlay graph corresponding to  $D$  by  $H$ .

From the process of building a P2P network, as we have previously discussed, theoretically,  $E(H)$  could be any subset of  $E(D)$ . However, an optimal overlay network should have the following basic properties. First, the selection of  $E(H)$  should make  $H$  a connected graph. Although in real systems,  $H$  could consist of several isolated components, an optimal overlay should include only one component such that a query can reach all peering nodes if the TTL is large enough.

Second, the overlay network  $H$  should be degree-bounded to balance the load of peers since an extremely large connection degree (the number of peering neighbors) of a peer causes heavy load and query drop. There is no close correlation between the number of connections and the number of incoming queries, but through our implementation on a super-peer Gnutella system, we find that the increase of the number of queries is trivial when one more leaf node is connected compared with having one more super-peer neighbor. This is because a super-peer typically sends out thousands of queries while a leaf peer only sends out one query in several minutes. When there are more than eight super-peer neighbors, a node sometimes witnesses more than  $11k$  incoming queries per minute. When the peer has more than 12 super-peer neighbors, there are always more than  $11k$  incoming queries per minute, and the peer starts dropping queries [13].

In our implementation, the *experimental node* is dedicated to the experiment, while in a real system a peer may have other conventional tasks. Moreover, a peer's local index normally includes many contents; while in our experiment, the local index is almost empty, which will reduce the time for local lookups. Based on such observations, we bound a peer's degree by 8, ignoring the connections with leaf peers in super-peer systems since they are not involved in query forwarding processes.

### 3.2 Metrics

A well-designed search mechanism should seek to optimize both system efficiency and quality of service (QoS) to users. Efficiency focuses on better utilization of resources, such as bandwidth and processing power, while QoS focuses on user-perceived qualities, such as the number of returned results and average response time of queries. If a query reaches more peers, it is more likely that the requested object can be found. Hence, in this study, we consider the query search scope and focus on traffic cost and query response time.

*Traffic cost* is one of the parameters network administrators are seriously concerned with. Heavy network traffic limits the scalability of P2P networks and is also a reason why a network administrator might prohibit P2P applications. We define the traffic cost as a 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, we simplify our assumption as all query messages have the same length, so when messages traverse an overlay connection during the given time period, the traffic cost  $C$  is given by  $C = M \times L$ , where  $M$  is the number of messages that traverse the overlay connection, and  $L$  represents the number of physical links in this overlay connection.

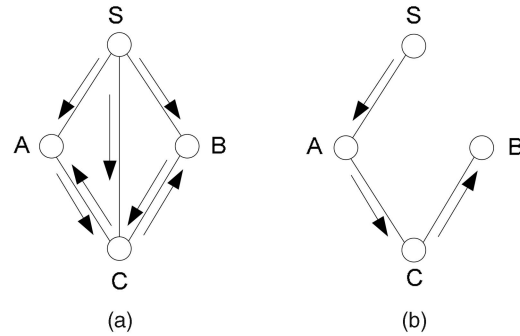


Fig. 2. Examples of P2P overlay topologies.

*Response time* of a query is defined as the time period from when the query is issued until when the source peer receives a result from the first responder. Indeed, there exist several explanations on response time, and we believe this is one of the representative definitions.

### 3.3 Unnecessary Message Duplications

Fig. 2a shows an example of overlay where solid lines denote logical connections among P2P neighbors. Consider the case when node  $S$  issues a query. A solid arrow represents a delivery of the query message along one logical connection. In Gnutella, a peer forwards an incoming query to all its neighbors, except the one that delivered that incoming query. Thus, as shown in Fig. 2a, the query is relayed by nodes  $A$ ,  $B$ , and  $C$ . Clearly, the pairs of transmissions along  $AC$  and  $BC$  are unnecessary. We can easily observe that the overlay shown in Fig. 2b has less unnecessary duplications, while retaining the same search scope for the query.

Generally, as long as loops exist in search paths, message duplications exist in overlay connections. Some peers are visited by the same query message multiple times. If a peer receives a query with the same Message ID (GUID) as the one it has received before, the peer will drop the message. Before discussing the optimal overlay in depth, we discuss the relationship between message duplications in overlay connections and the number of overlay links.

**Theorem 1.** Let  $H(V, E)$  be an overlay graph. An arbitrary query message issued by an arbitrary peer with a sufficiently large TTL value can result in  $d = 2(|E(H)| - |V(H)| + 1)$  duplicate messages in the overlay graph.

**Proof.** We assume that  $H$  is nontrivial and connected, which implies  $H$  will contain at least  $|V(H)| - 1$  edges. The proof is done by induction of  $E(H)$ .

*Basis.*  $|E(H)| = |V(H)| - 1$ .

In this case,  $H$  is a tree, and therefore, there is no loop in the graph to generate duplicate messages, and thus, the assertion holds, as  $d = 2(|E(H)| - |V(H)| + 1) = 0$ .

*Hypothesis.* Assume the assertion is correct for  $|E(H)| < k$ , where  $k \geq |V(H)|$ .

*Step.* We want to show that the assertion is correct when  $|E(H)| = k$ . Let  $H$  be an arbitrary overlay connected graph with  $|E(H)| = k$  edges. Since  $H$  has more than  $|V(H)|$  edges,  $H$  must contain at least one cycle  $C$ . Let  $uv$  be an edge of  $C$ , and now, consider the graph  $H' = H - uv$ . Since  $H$  was assumed to be

connected and  $uv$  belongs to a loop,  $H'$  is also connected. Further, since  $|E(H')| < k$ , by the induction hypothesis, there is an arbitrary query message, called  $Q$ , issued by an arbitrary peer with a sufficiently large TTL value that results in  $2(|E(H')| - |V(H')| + 1)$  duplicate messages in the overlay graph  $H'$ . Now, if we initiate the same query  $Q$  in  $H$ , sticking to the same message propagation as in  $H'$ , at some point, peers  $u$  and  $v$ , the end vertices of the edge  $uv$ , will be informed. These peers in turn can potentially send messages to each other, generating two unnecessary messages on the link  $uv$ . Therefore, the number of duplications in  $H$  for the message query  $Q$  will be  $2(|E(H')| - |V(H')| + 1) + 2$ . Rewriting this expression in terms of  $E(H)$  and  $V(H)$  and remembering that  $V(H) = V(H')$ , we get  $2(|E(H')| - |V(H')| + 1) + 2 = 2(|E(H')| + 1 - |V(H')| + 1) = 2(|E(H)| - |V(H)| + 1)$  and, thus, the theorem.  $\square$

### 3.4 Problem Definition

Having discussed the message duplications in the overlay level, let us look at the example again. If we can change the overlay in Fig. 2a into the one in Fig. 2b, does it mean no message duplications? The example we plot in Fig. 1 clearly gives the answer as "NO." If the peers  $A$ ,  $B$ , and  $C$  are in similar positions as in Fig. 1, when  $A$  forwards the query to  $C$  and  $B$ , although there is no duplication in the overlay level, due to the topology mismatch, there is still unnecessary traffic at the physical level.

Duplications in overlay links increase traffic cost and also reduce response time. We should not reduce the topology into an MST. Duplications from topology mismatch, however, are completely negative, and we should alleviate them as much as we can. Indeed, it is impossible to minimize the two metrics simultaneously, so we seek an optimal overlay when the cardinality of edge set  $E$  is given.

**Definition 1.** Let  $F = (V, E)$  be an edge weighted connected graph. Further, let  $dis(u, v)$  represent the distance (that is, the length of a shortest path) between vertices  $u$  and  $v$ . The average distance, denoted by  $AD(F)$ , of graph  $F$  is defined as follows:

$$AD(F) = \frac{1}{\binom{|V(F)|}{2}} \sum_{\text{unordered pair } u, v \in V(F)} dis(u, v).$$

We now define the DMAD overlay problem.

**Definition 2.** Let graph  $G = (V, E)$  represent a physical network and let graph  $D = (V, E)$  be its distance graph for a set of communicating peers, as defined earlier. Recall that  $D$  is a complete graph. Further, let  $k$  be an integer such that  $\delta(D) \leq k \leq \Delta(D)$ , where  $\delta(D)$  and  $\Delta(D)$  are the minimum and the maximum degree of  $D$ , respectively. A DMAD overlay graph  $H = (V, E)$  is a connected spanning subgraph of  $D$  having the following properties:

1.  $\Delta(H) \leq k$ ;
2.  $AD(H)$  is as small as possible, subject to Property 1.

The DMAD problem is a generalization of the Minimum-Degree Spanning Tree (MDST) problem, a known NP-hard problem [7]. Given an undirected graph  $G$ , the MDST problem is to find a spanning tree of  $G$  whose maximum degree is minimal.

In the following, the words DMAD and MDST are the decision versions of themselves, respectively. The two decision versions can be formulated as follows:

**DMAD:** Given an undirected graph  $G$ , find a connected spanning subgraph of  $G$  whose maximum degree is less than or equal to  $k$  and the average distance of the subgraph is less than or equal to  $d$ .

**MDST:** Given an undirected graph  $G$ , find a spanning tree of  $G$  whose maximum degree is less than or equal to  $k$ .

**Proof.** To prove the DMAD problem is NP-complete, it suffices to show the following two facts:

1. DMAD  $\in$  NP;
2. MDST  $\leq_P$  DMAD.

Property 1 is obvious. To prove Property 2, we design a polynomial transformation from MDST to DMAD as follows:

$f: (G(V, E), k) \rightarrow (G(V, E), k, d)$ , where  $d = \sum_{e \in E} w(e)$ , and  $w(e)$  denotes the weight of edge  $e$ .

Clearly, a YES input of MDST is a YES input of DMAD since the spanning tree  $T$  of  $G$ , whose maximum degree is less than or equal to  $k$ , is also a connected spanning subgraph of  $G$  and the average distance of  $T$  is less than or equal to  $d$  by definition. Conversely, a YES input  $G$  of DMAD has a subgraph  $G'$  satisfying the degree constraints. The spanning tree  $T$  of  $G'$  can be achieved by executing BFS or DFS on  $G'$ . The maximal degree of  $T$  cannot be larger than the maximal degree of  $G'$  since we only remove the edges in the construction of  $T$  from  $G'$ . Therefore,  $T$  satisfies the degree constraints and  $G$  is also a YES input of MDST. So far, we prove DMAD is NP-complete and the corresponding optimization problem is NP-hard.  $\square$

## 4 TWO HOP AWAY NEIGHBOR COMPARISON AND SELECTION (THANCS)

Practically, it is impossible for a peer to collect global knowledge of the overlay topology since the number of online users are typically millions and they are randomly coming and leaving. It is clear, however, that optimizing inefficient overlays can fundamentally improve P2P search efficiency. Thus, we need an overlay optimization algorithm, which has the following properties. First, it is completely distributed and does not need any global knowledge. Second, the traffic overhead incurred by the algorithm and computation overhead is trivial compared with traffic cost savings. Third, the convergent speed of the algorithm must be fast enough so that it is effective in dynamic environments. Finally, the search scope is not shrunk.

In this paper, we present the design of the THANCS scheme that effectively attacks topology mismatch and optimizes the overlay topology. THANCS consists of two main components: piggybacking neighbor distance on queries and neighbor comparison and selection.

### 4.1 Piggybacking Neighbor Distance on Queries

We use network delay as the metric for measuring the distance between peers. In THANCS, each peer probes the

TABLE 1  
PM Body

	Neighbor IP Address		Neighbor Distance	
Byte offset	0	3	4	5

distances with its immediate logical neighbors and stores the information locally. Peers in Gnutella have a limited number of neighbors, so the overhead of this operation is not much. Since Internet paths are relatively stable [26], to keep the neighbor information up to date, a peer only needs to probe the distance to new neighbors and modify the neighbor distance information when neighbors leave.

We add one special query message type, *Piggy Message* (PM), by modifying the Lime Wire implementation of Gnutella 0.6 P2P protocol [6]. A PM has a message body in a format shown in Table 1, including two fields: Neighbor IP Address and Neighbor Distance. It does not need the Gnutella's unified 23-byte header. A peer constructs a PM for each of its neighbors. In order not to increase the number of messages, THANCS is designed to piggyback a PM on a query message. For example, a peer  $P$  constructs a PM for its neighbor  $Q$ , which contains  $Q$ 's IP address and the distance from  $P$  and  $Q$ . When  $P$  receives a query from  $Q$ , this PM will be piggybacked by the query that goes to all the other neighbors of peer  $P$ . The payload type of a query piggybacking a PM can be defined as  $0 \times 82$  to distinguish it from a normal query message ( $0 \times 80$ ). The payload length of such a query message will be increased by 6. Upon receiving such a query message, each of the other neighbors will detach the PM from the query, record the distance of  $PQ$ , and process the query. This PM will not be further forwarded blindly, but it is possible for the query message to piggyback another PM.

Obviously, it is not necessary for all query messages from  $Q$  to carry a PM, especially when  $P$  receives queries from  $Q$  frequently. One critical issue to be examined here is which incoming queries should piggyback a PM. Although a PM is only 6 bytes long, a large amount of piggybacking still inserts nontrivial overhead into the network. In this design, we present two policies for the selection: pure probability-based (PPB) policy and new neighbor triggered (NNT) policy.

The PPB policy employs a predefined probability,  $\alpha$ , for a query to attach a PM. That means in each peer, the probability for a query to piggyback a PM is  $\alpha$ . A smaller  $\alpha$  means fewer queries to piggyback messages. Note that in this scheme, it is not to say that once a query starts to piggyback a PM, the PM is always attached to that query until the query is dropped. Instead, each PM is piggybacked for only one hop. With the probability  $\alpha$ , the query message may piggyback another PM. The major advantage of PPB is simplicity.

In the NNT policy, query messages do not carry PMs until a peer detects a newly arrived neighbor. As shown in Fig. 3, for example, when a peer  $P$  detects a new neighbor  $N$ , it operates as follows: First,  $P$  probes the distance with  $N$  and constructs a PM, then attaches the PM to the first query message coming from  $N$ . This query will be forwarded to all  $P$ 's logical neighbors except  $N$ . Second, the first incoming query from each of  $P$ 's existing neighbors will carry a corresponding PM (with the distance to this

neighbor) when it is forwarded to  $N$ . The query message, however, will not carry a PM when it is forwarded to other neighbors, as illustrated in Fig. 3. Another option is to let the first query message from  $P$ 's previous neighbors piggyback all the PMs (except the one for the new neighbor) to the new neighbor,  $N$ . Compared with the PPB policy, NNT is relatively complicated while it has smaller communication overhead. We will discuss the selection of the two policies in more detail in Section 5.

We use  $N(S)$  to denote the set of direct logical neighbors of node  $S$  and use  $N^2(S)$  to denote the set of peers being two logical hops away from  $S$ . After receiving a number of PMs, an arbitrary peer  $S$  will know the distance between each peer in  $N(S)$  and any of the peers in  $N^2(S)$  that are connected with the peer in  $N(S)$ . This information will be used in the second component, neighbor comparison and selection.

## 4.2 Neighbor Comparison and Selection

The behavior of THANCS peers in neighbor comparison and selection component is illustrated through an example in Fig. 4. A peer,  $S$ , probes the distance to all the known unprobed  $N^2(S)$  peers. The distance of  $SP$  is known to  $S$ . When peer  $S$  receives a PM from peer  $P$  with the distance of  $PQ$ , there are two cases as follows:

*Case 1.* Peer  $Q$  is also a direct neighbor of peer  $S$ , i.e.,  $Q \in (N(S) \cap N^2(S))$ . In this case, peer  $S$  will compare the cost of  $SQ$ ,  $SP$ , and  $PQ$ . If  $SQ$  or  $SP$  is the longest in these three connections,  $S$  will put the longest connection into its *will-cut list* that is a list of connections to be cut later, e.g., 50 seconds later [11]. If  $PQ$  is the longest, peer  $S$  will do nothing because the scheme is fully distributed and peer  $P$  or  $Q$  will disconnect  $PQ$  shortly. 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.

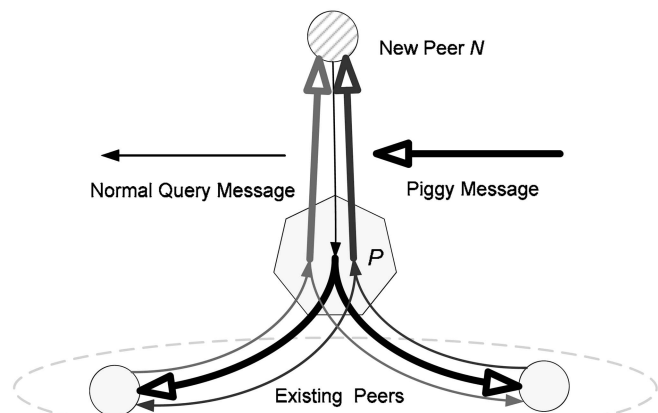


Fig. 3. NNT policy.

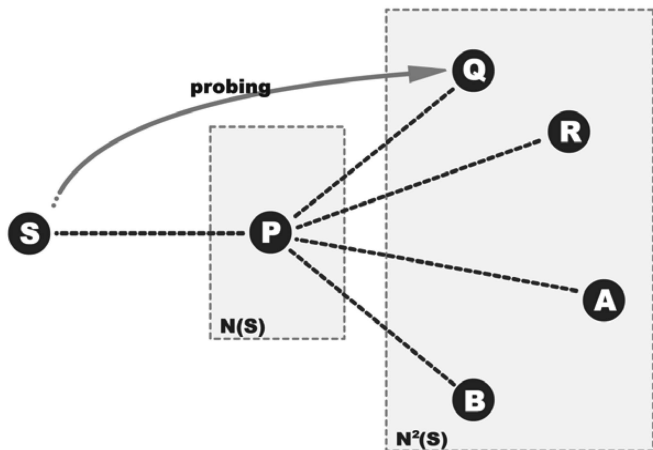


Fig. 4. Probing two-hop-away neighbors.

Case 2.  $Q$  is a two-hop-away neighbor of  $S$  but not a direct logical neighbor of  $S$ , i.e.,  $Q \in (N^2(S) - N(S))$ . Peer  $S$  will first check whether it had probed peer  $Q$  before or used to have peer  $Q$  as a direct neighbor by looking up  $S$ 's distance cache. The distance cache is designed to keep a list of peers that have been probed by peer  $S$ , so as to avoid duplicate probing. If peer  $S$  used to probe the distance to  $Q$ ,  $S$  will do nothing with peer  $Q$  and start probing other peers in  $N^2(S)$ . Otherwise,  $S$  probes the distance to  $Q$  and stores the result in the distance cache. Having the distance of  $SQ$ ,  $S$  compares  $SQ$ ,  $SP$ , and  $PQ$ . If  $SQ$  is the longest,  $S$  will not construct the connection with peer  $Q$ . If  $SP$  is the longest,  $S$  will construct  $SQ$  and put  $SP$  into the will-cut list. If  $PQ$  is the longest in the three connections,  $S$  will keep the connection with both  $P$  and  $Q$ , expecting that  $P$  or  $Q$  will disconnect  $PQ$  later.

The first component, *piggyback neighbor distance on queries*, is straightforward, for which we do not provide the pseudocode. The pseudocode of *neighbor comparison and selection* component for a given source peer  $i$  is shown below. Let  $D_{ij}$  represent the distance from peer  $i$  to  $j$ .

```

for each  $h \in N^2(i)$  in  $j$ 's direct neighbors,
  if ( $h \in N(i)$ )
    {check  $D_{ih}$ ;
     if ( $D_{ih} > D_{ij}$ )  $\cap$  ( $D_{ih} > D_{jh}$ )
       {put connection  $ih$  into  $i$ 's will-cut list;}
     else if ( $D_{ij} > D_{ih}$ )  $\cap$  ( $D_{ij} > D_{jh}$ )
       {put connection  $ij$  into  $i$ 's will-cut list;}
     end if
  else
    if ( $D_{ih}$  is not in  $i$ 's distance cache)
      Probe  $D_{ih}$ ;
      if ( $D_{ih} < D_{ij}$ )  $\cup$  ( $D_{ih} < D_{jh}$ )
        {keep the connection with peer  $h$  and include
          $h$  as a direct logical neighbor of  $i$ ;
         if ( $D_{ij} > D_{ih}$ )  $\cap$  ( $D_{ij} > D_{jh}$ )
           {put connection  $ij$  into  $i$ 's will-cut list;}
         }
        end if
      end if
    end if
  end if
end for

```

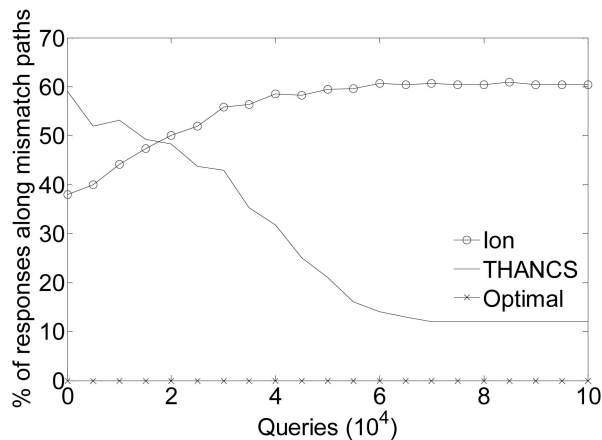


Fig. 5. The percent of mismatched paths of the three schemes.

## 5 PERFORMANCE EVALUATION

In our simulations, we use two types of topologies: logical (P2P overlay topology) and physical (the Internet topology). The logical topology represents the overlay on top of the physical topology, coming from a recent snapshot kit of Ion P2P [22], which logged data from September 2004 to February 2005, including topologies with super peers (i.e., maintaining more than 30 neighbors). All P2P nodes are in a subset of nodes in the physical topology.

The physical topology represents the real topology with Internet characteristics. Previous studies have shown that Internet topologies [23] 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. The study in [19] found that the topologies generated using the AS Model have the properties of the small world and power law. BRITE [2] is a topology generation tool that provides the option to generate topologies based on the AS Model. Using BRITE, we generate three physical topologies, each with around 27,000 nodes. A search operation is simulated by randomly choosing a peer as the sender and a keyword according to Zipf distribution. Totally, 1,000,000 search operations are simulated.

### 5.1 On Attacking Topology Mismatch

We first study the effectiveness of THANCS in a static P2P environment where peers do not join and leave. We contrast the THANCS optimized overlay with an optimal overlay. This set of simulation is conducted in a small scale since finding an optimal overlay is proved to be NP-hard in Section 3. Specifically, 128 nodes are abstracted from the Ion trace as peering nodes.

We simulate 100,000 search operations on 1) the Ion-like overlay, 2) a THANCS optimized overlay, and 3) an optimal overlay obtained by a brute force algorithm. We assume that the underlying physical layer uses shortest path routing with delay as the metric. Each simulation takes 20 runs with different random choices of peers and we report the average. Fig. 5 plots the percentage of query responses along mismatched paths of the three schemes.

Recall that around 60 percent of the queries are back along mismatched paths in Ion-like P2Ps. An optimal overlay effectively replaces all the mismatched paths, as shown in Fig. 5. THANCS is proven to be an effective

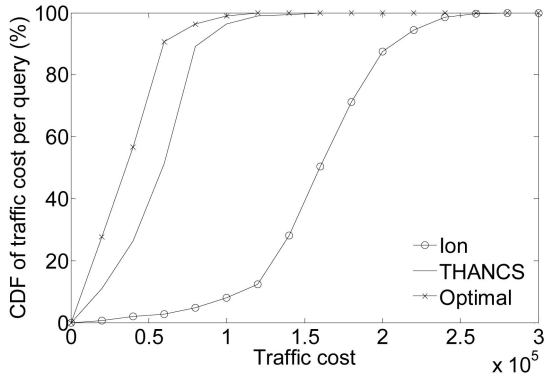


Fig. 6. Average traffic cost per query.

approach, optimizing up to 45 percent out of the 60 percent mismatched paths. As a result, system performance is improved significantly. Fig. 6 shows that the traffic cost,  $c$ , decreases by up to 60 percent when optimization operations of THANCS are conducted. Fig. 7 shows that THANCS shortens the query response time by about 40 percent. We also show the search efficiency on the optimal overlay in those two figures. We can see that the search traffic in an optimal overlay is typically less than 20 percent of the lon-like overlay, indicating roughly 80 percent of the traffic is from the topology mismatch. Due to the lack of global knowledge, however, THANCS can only obtain a suboptimal overlay, but its performance is close to the optimal solution in the static environment.

**5.2 PPB versus NNT**

Another major difficulty of solving topology mismatch is that peers are randomly coming and leaving, especially when real P2P systems have a huge number of online users. We further evaluate THANCS in a dynamic environment. We first discuss how we select a good policy from PPB and NNT for this design.

THANCS is a completely distributed approach, where each single peer determines its operations independently. One key issue is which queries should carry a PM message. We present two options in Section 4.1, PPB and NNT. In this set of simulation, we use a P2P overlay with 5,000 nodes to examine the two designs.

Fig. 8 plots the performance of THANCS on attacking mismatch using NNT policy and PPB policy, respectively.

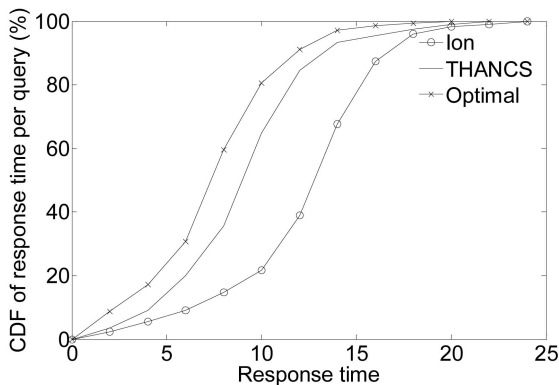


Fig. 7. Average query response time.

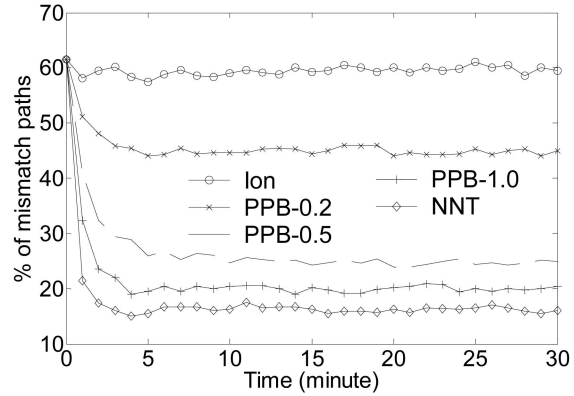


Fig. 8. NNT versus PPB: reducing mismatch.

The curve of “PPB- $\alpha$ ” means that the probability of an incoming query to attach a PM is  $\alpha$ . Fig. 9 plots the traffic overhead of the two policies.

Under PPB policy, each peer blindly selects a portion of the queries using a straightforward rule to carry the PMs without the need of monitoring neighbor behaviors. Indeed, the overhead of PPB is not intolerable, although it is high compared with NNT. In this simulation, the mismatch reduction of PPB-1.0 is close to NNT policy, but PPB-1.0 has the largest traffic overhead. Such overhead, however, is only equivalent to the traffic of a couple of queries per minute, accounting for less than 2 percent of the traffic savings by THANCS. The NNT policy can further reduce the optimization overhead, but it is more complicated in its implementation compared with PPB.

Simulation results in Figs. 8 and 9 show that NNT outperforms PPB in the sense that it has a higher optimization rate and a smaller traffic. Thus, we adopt NNT. The traffic overhead of NNT is less than 0.3 percent of the traffic savings.

**5.3 THANCS in Dynamic Environments**

We simulate joining and leaving behavior of peers via turning on/off logical peers. The peer lifetime is generated according to the distribution observed in [4] and [19]. The lifetime is decreased by one after passing each second and a peer will leave in the next second when its lifetime reaches zero. During each second, a number of peers leave. We

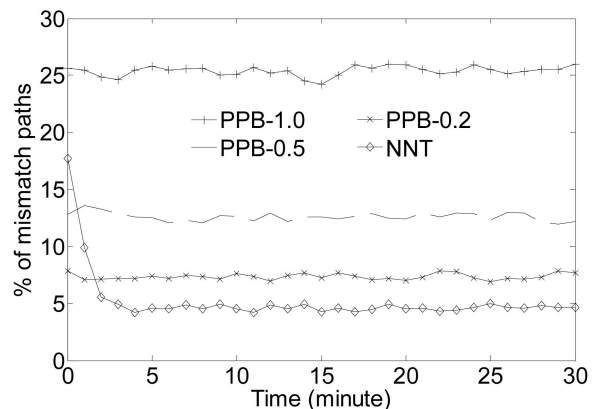


Fig. 9. NNT versus PPB: overhead.

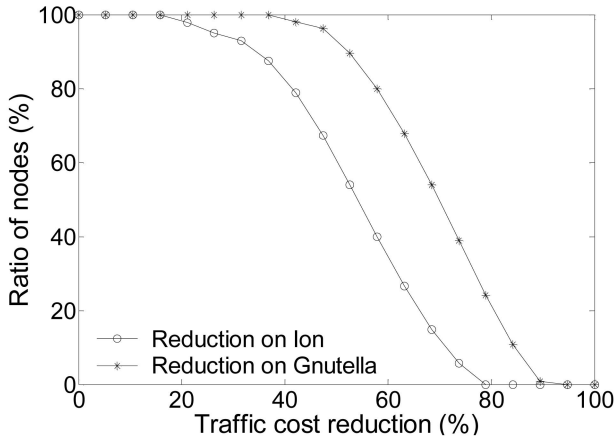


Fig. 10. Traffic cost reduction.

then randomly pick up (turn on) the same number of peers from the overlay. We simulate the search process on both a Gnutella-like topology and an Ion-like trace for 30 minutes and repeat this simulation with different random seeds for 20 times. We use two metrics, traffic reduction rate,  $R_c(*)$ , and response time reduction rate,  $R_t(*)$ .

$R_c(*)$  is defined by  $R_c(*) = \frac{C(\text{Gnutella/Ion-like}) - C(\text{THANCS})}{C(\text{Gnutella/Ion-like})} \times 100\%$ , where  $C(*)$  represents the traffic cost incurred by searching all the peers using the given search technique (\*), such as flooding or random walk, in a Gnutella-like or Ion-like overlay, and  $C(\text{THANCS})$  represents the traffic cost incurred in THANCS-enabled P2P networks.

$R_t(*)$  is given by  $R_t(*) = \frac{T(\text{Gnutella/Ion-like}) - T(\text{THANCS})}{T(\text{Gnutella/Ion-like})} \times 100\%$ , where  $T(\text{Gnutella/Ion-like})$  denotes the average response time of queries in Gnutella-like or Ion-like overlays, and  $T(\text{THANCS})$  is that of THANCS-enabled systems. In this simulation, we study the performance of THANCS based on the blind flooding mechanism, and we will demonstrate the effectiveness of THANCS on other advanced search strategies shortly.

The variance of overlay size in different simulation runs is trivial, so we do not show the confidence intervals in the figures. Fig. 10 shows that the traffic cost saving is up to approximately 70 percent for Gnutella and around 55 percent for Ion topologies. The optimization rate of Ion-like system is lower than that of Gnutella because super-peer architectures are more efficient than flat overlays, and a number of queries are answered by one-hop super peer without inserting into the network. At the same time, average query response time is also decreased by approximately 60 percent for Gnutella and 45 percent for Ion, as shown in Fig. 11.

#### 5.4 Effectiveness with Other Advanced Search Strategies

In the above experiments, we run simulations under flooding search mechanisms based on the following observations. First, although many advanced search techniques are designed, flooding-based search is still the most popular mechanism widely adopted in today's P2P systems. Second, topology optimization is orthogonal to existing search techniques, such as forwarding-based or cache-based optimizations. THANCS can be combined with

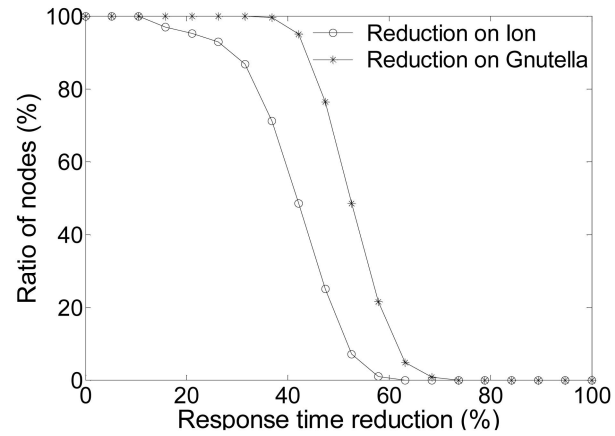


Fig. 11. Response time reduction.

other types of advanced search strategies and further improve efficiency. Therefore, the effectiveness of THANCS on flooding-based search can also reflect the effectiveness of this design on other search mechanisms.

In next set of simulation, we combine THANCS with two popular advanced search mechanisms, *Random Walk* [15] and *Index Caching* [21]. We first simulate search under a simple random walk scheme. For each query, we issue 30 walkers. We then combine THANCS with index caching, in which query responses are cached in passing peers along the returning path [21]. Each peer keeps a local cache and a response index cache. The size of a response index cache is limited by 200 items.

Fig. 12 plots  $R_c(\text{Random Walk})$ ,  $R_r(\text{Random Walk})$ ,  $R_c(\text{Index Cache})$ , and  $R_r(\text{Index Cache})$ . THANCS further reduces traffic cost and response time of the index caching scheme by approximately 80 percent and 39 percent, respectively, and reduces both traffic cost and response time of the random walk by more than 55 percent.

#### 5.5 Comparative Studies

We further contrast this design with three previously proposed approaches, AOTO [10], LTM [12], and SBO [14] in Figs. 13 and 14. The convergent speed of AOTO is the slowest, so its performance in dynamic environments is not as good as the other three approaches. The convergent

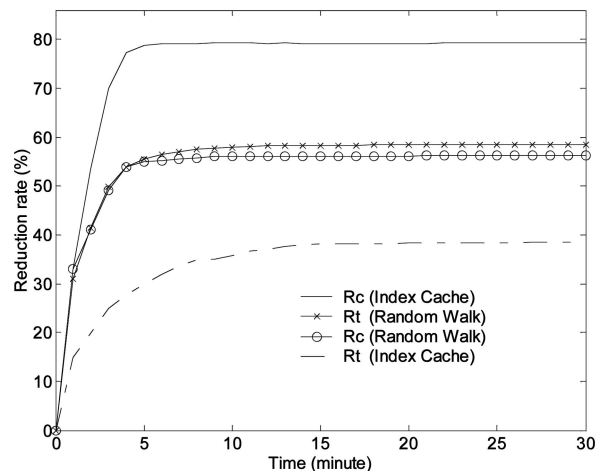


Fig. 12. THANCS combined with random walk and index cache.

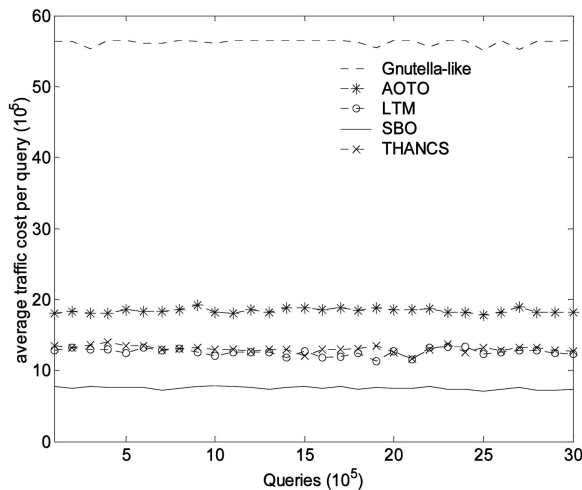


Fig. 13. Traffic cost.

speed of LTM is the fastest, but it needs the support of NTP [3] to synchronize peers. THANCS has similar performance with LTM but does not need synchronization. SBO, incurring half overhead of AOTO, reduces the traffic cost the most [14]. Generally speaking, SBO reduces more query traffic, while THANCS has lower response time, as THANCS converges faster than SBO. Indeed, THANCS performs better in a more dynamic environment. In addition, THANCS is easy to implement and its operation overhead is trivial, compared with the other three approaches. This design, however, has a limitation that it is not easy to extend to non-flooding-based P2Ps, such as CAN and Chord, while the other three designs can be easily extended to any P2P systems.

## 6 CONCLUSIONS

The quality of overlays significantly affects search efficiency of P2P systems. Our observations show that at the least 60 percent of the queries and their responses are delivered to the data requester along mismatched paths, increasing both traffic cost and search response time.

We model unstructured P2P systems based on our observations and implementations of Gnutella peers. We prove that even with global knowledge, computing an optimal overlay is NP-hard. We define the problem as the DMAD problem.

To address topology mismatch, we propose a two-hop approach, THANCS, which is scalable in the sense that it does not need any global knowledge, and each peer performs operations independently. Through trace-driven simulations, we show that the performance of THANCS is close to an optimal solution. We have also shown its other two strengths: its optimization convergent speed is fast even when peers are randomly coming and leaving, and it is orthogonal to other types of advanced search approaches.

## ACKNOWLEDGMENTS

Some preliminary results of this work have been presented in the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. The author would like to thank Professor

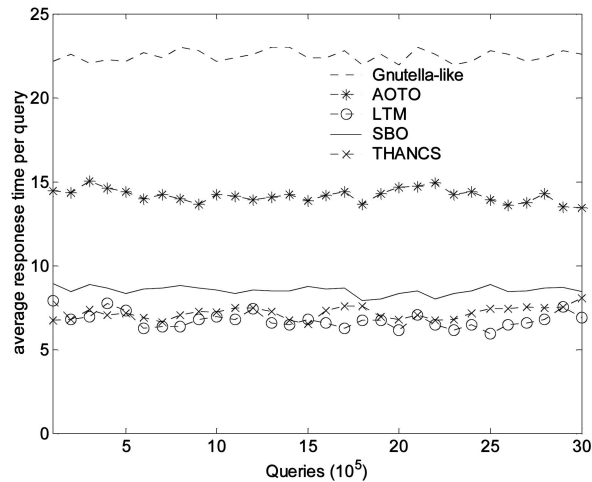


Fig. 14. Response time.

Lionel Ni, Dr. A.-H. Esfahanian, and Dr. Li Xiao for their participation in the early stage of this work and valuable discussions. This research was supported in part by Hong Kong RGC Grants N\_HKUST614/07, HKUST6169/07E, and HKUST6152/06E, HKUST Nansha Research Fund NRC06/07.EG01, NSF China Key Project under Grants 60533110 and 60736016, the National High Technology Research and Development Program of China (863 Program) under Grant 2007AA01Z180, and Nokia APAC research grant.

## REFERENCES

- [1] *Fasttrack*, <http://developer.berlios.de/projects/gift-fasttrack/>, 2007.
- [2] *BRITE*, <http://www.cs.bu.edu/brite/>, 2007.
- [3] *NTP: The Network Time Protocol*, <http://www.ntp.org/>, 2007.
- [4] R. Bhagwan, S. Savage, and G.M. Voelker, "Understanding Availability," *Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS)*, 2003.
- [5] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM*, 2003.
- [6] Y. Chu, S.G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM SIGMETRICS*, 2000.
- [7] M. Furer and B. Raghavachari, "Approximating the Minimum Degree Spanning Tree to within One from the Optimal Degree," *Proc. Third Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 1992.
- [8] D. Guo, J. Wu, H. Chen, and X. Luo, "MOORE: An Extendable Peer-to-Peer Network Based Incomplete Kautz Digraph with Constant Degree," *Proc. IEEE INFOCOM*, 2007.
- [9] B. Krishnamurthy and J. Wang, "Topology Modeling via Cluster Graphs," *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2001.
- [10] Y. Liu, Z. Zhuang, L. Xiao, and L.M. Ni, "AOTO: Adaptive Overlay Topology Optimization in Unstructured P2P Systems," *Proc. IEEE Global Comm. Conf. (GLOBECOM)*, 2003.
- [11] Y. Liu, X. Liu, L. Xiao, L.M. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems," *Proc. IEEE INFOCOM*, 2004.
- [12] Y. Liu, L. Xiao, X. Liu, L.M. Ni, and X. Zhang, "Location Awareness in Unstructured Peer-to-Peer Systems," *IEEE Trans. Parallel and Distributed Systems (TPDS)*, vol. 16, pp. 163-174, 2005.
- [13] Y. Liu, X. Liu, C. Wang, and L. Xiao, "Defending P2Ps from Overlay Flooding-Based DDoS," *Proc. Int'l Conf. Parallel Processing (ICPP)*, 2007.
- [14] Y. Liu, L. Xiao, and L.M. Ni, "Building a Scalable Bipartite P2P Overlay Network," *IEEE Trans. Parallel and Distributed Systems (TPDS)*, vol. 18, no. 9, pp. 1296-1306, Sept. 2007.

- [15] 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 (ICS)*, 2002.
- [16] V.N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," *Proc. ACM SIGCOMM*, 2001.
- [17] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," *Proc. ACM SIGCOMM*, 2004.
- [18] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing*, 2002.
- [19] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking (MMCN)*, 2002.
- [20] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2002.
- [21] K. Sripanidkulchai, *The Popularity of Gnutella Queries and Its Implications on Scalability*, <http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html>, 2007.
- [22] D. Stutzbach and R. Rejaie, "Characterizing the Two-Tier Gnutella Topology," *Proc. ACM SIGMETRICS*, 2005.
- [23] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based versus Structural," *Proc. ACM SIGCOMM*, 2002.
- [24] W.W. Terpstra, J. Kangasharju, C. Leng, and A.P. Buchmann, "BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search," *Proc. ACM SIGCOMM*, 2007.
- [25] L. Xiao, Y. Liu, and L.M. Ni, "Improving Unstructured Peer-to-Peer Systems by Adaptive Connection Establishment," *IEEE Trans. Computers*, 2005.
- [26] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the Constancy of Internet Path Properties," *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2001.



**Yunhao Liu** received the BS degree in automation from Tsinghua University, Beijing, in 1995, the MA degree from Beijing Foreign Studies University, Beijing, in 1997, and the MS and PhD degrees in computer science and engineering from Michigan State University, East Lansing, in 2003 and 2004, respectively. He is currently an assistant professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. He is also an adjunct professor at Xi'an Jiaotong University, Xi'an, China, and Ocean University of China, Qingdao, China. His research interests include peer-to-peer computing, pervasive computing, and sensor networks. He is a senior member of the IEEE and a member of the ACM.

► **For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).**