Improving Unstructured Peer-to-Peer Systems by Adaptive Connection Establishment

Li Xiao, *Member*, *IEEE*, Yunhao Liu, *Member*, *IEEE*, and Lionel M. Ni, *Fellow*, *IEEE*

Abstract—In unstructured peer-to-peer (P2P) systems, the mechanism of a peer randomly joining and leaving a P2P network causes a topology mismatch between the P2P logical overlay network and the physical underlying network, incurring a large volume of redundant traffic in the Internet. In order to alleviate the topology mismatch problem, we propose Adaptive Connection Establishment (ACE), an algorithm for building an overlay multicast tree among each source node and the peers within a certain diameter from the source peer and further optimizing the neighbor connections that are not on the tree while retaining the search scope. Our simulation study shows that this approach can effectively solve the mismatch problem and significantly reduce P2P traffic. We further study the trade-offs between the topology optimization rate and the information exchange overhead by changing the diameter used to build the tree.

Index Terms—Peer-to-peer systems, overlay, topology mismatch problem, distributed approach, Adaptive Connection Establishment.

1 Introduction

As an emerging model of communication and computation, peer-to-peer systems are under intensive study. In unstructured P2P systems, queries are flooded among peers (such as in Gnutella [2]) or among super-nodes (such as in KaZaA [3]). In such systems, all participating peers form a P2P network over a physical network. A P2P network is an abstract, logical network called an *overlay network*. When a new peer wants to join a P2P network, a bootstrapping 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. This mechanism of a peer joining a P2P network and the fact of a peer randomly joining and leaving causes an interesting matching problem between a P2P overlay network topology and the underlying physical network topology.

Studies in [23] show 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 ASs. This means that most Gnutella-generated traffic crosses AS borders so as to increase topology mismatch costs. The same message can traverse the same physical link multiple times, causing a large amount of unnecessary traffic.

The objective of this paper is to minimize the effect due to topology mismatch. We propose the *Adaptive Connection Establishment* (ACE) that builds an overlay multicast tree among each source node and the peers within a certain diameter from the source peer and further optimizes the neighbor connections that are not on the tree while retaining the search scope. ACE is scalable and completely distributed in the sense that it does not require global knowledge of the whole overlay network when each node is optimizing the organization of its logical neighbors. Our simulations show that ACE can significantly improve the performance. We also show that a larger diameter leads to a better topology optimization rate and a higher overhead due to extra information exchanging. Our experiments and discussions provide a guide on how to achieve a good performance by considering the trade-offs between the topology optimization rate and the information exchange overhead in selecting the diameter to determine the peers to form the multicast tree for a source peer.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents the adaptive connection establishment (ACE) scheme. Section 4 describes our simulation methodology. Performance evaluation of the ACE is presented in Section 5 and we conclude the work in Section 6.

2 Related Work

In order to reduce unnecessary flooding traffic and improve search performance, two approaches have typically been proposed to improve from the flooding-based search mechanism in unstructured P2P systems. Rather than flooding a query to all neighbors, the first approach routes queries to peers that are likely to have the requested items by some heuristics based on maintained statistic information [17], [33], [34]. In the second approach, a peer keeps indices of other

- *L. Xiao is with the Department of Computer Science and Engineering, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824. E-mail: lxiao@cse.msu.edu.*
- *Y. Liu and L.M. Ni are with the Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: {liu, ni}@cs.ust.hk.*

Manuscript received 14 Mar. 2004; revised 28 Nov. 2004; accepted 30 Mar. 2005; published online 15 July 2005.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0089-0304.

peers' sharing information or caches query responses in hoping that subsequent queries can be satisfied quickly by the cached indices or responses [11], [18], [19], [22], [26], [29], [33]. The performance gains of both approaches are seriously limited by the topology mismatch problem.

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, was proposed in [10], 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 a large overhead of 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 not feasible for large-scale P2P systems.

Researchers have also considered clustering close peers based on their IP addresses (e.g., [13], [21]) or probed distances [20]. We believe there are two limitations to this approach. First, the mapping accuracy is not guaranteed by this approach. Second, this approach may affect the search 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 peers. Furthermore, our scheme does not shrink the search scope.

Researchers in [32] have proposed measuring 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.

Gia [8] 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. Apocrypha [12] optimized the overlay topology by swapping a number of selected pairs of peers.

A preliminary design of ACE [16], which is called AOTO, has been discussed in [15]. We have also proposed a location-aware topology match scheme [14] in which 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. However, this approach creates slightly more overhead and requires that the clocks in all peers be synchronized.

3 Adaptive Connection Establishment

In unstructured P2P systems, the most popular search mechanism in use is to blindly "flood" queries to the network among peers or among super-nodes. 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. 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. In this section, we first use examples to explain the unnecessary traffic incurred by flooding-based search and the topology mismatch problem. We then introduce the design of proposed approach, ACE.

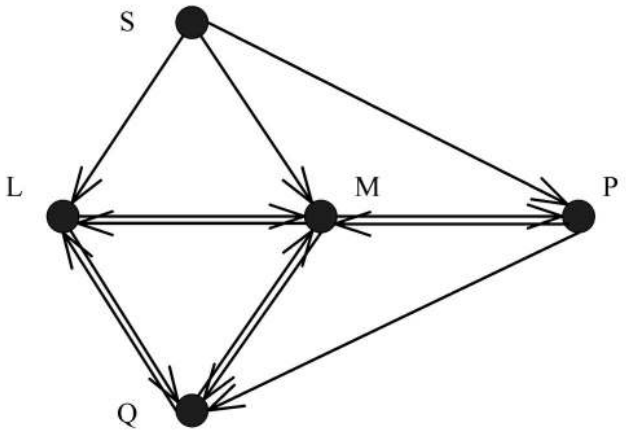

Fig. 1. An example of P2P overlay.

3.1 Unnecessary Message Duplications

3.1.1 Unnecessary Traffic by Flooding

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. For example, after node S sends the query to L and M, since it is possible that none of L or M knows the other one will receive the same query from S, they will forward the query to each other. The pair of transmissions on the logical link LM is unnecessary. In such a simple overlay, node M will receive the same query message for up to 4 times. In this case, it is clear that the search scope of the query from node S will not shrink without logical connections of LM, MQ, LQ, and MP.

3.1.2 Topology Mismatch Problem

As we have discussed, the stochastic peer connection and peers' randomly joining and leaving a P2P network can cause a topology mismatch between the P2P logical overlay network and the physical underlying network. For example, Fig. 2a and Fig. 2b are two overlay topologies on top of the underlying physical topology shown in Fig. 2c. Suppose nodes S and B are in the same autonomous system (AS) at Michigan State University in the USA, while nodes A and C are in another AS at Tsinghua University in China. We can assume that the physical link delay between nodes S and C

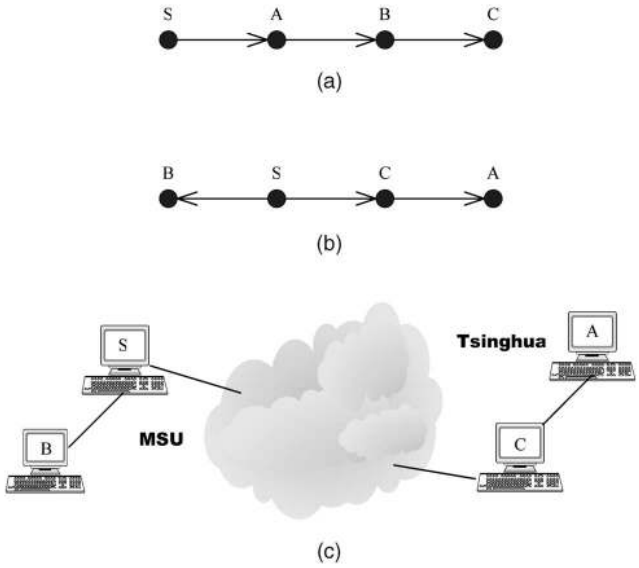

Fig. 2. Topology mismatch problem. (a) Mismatching overlay. (b) Matching overlay. (c) Underlying physical topology.

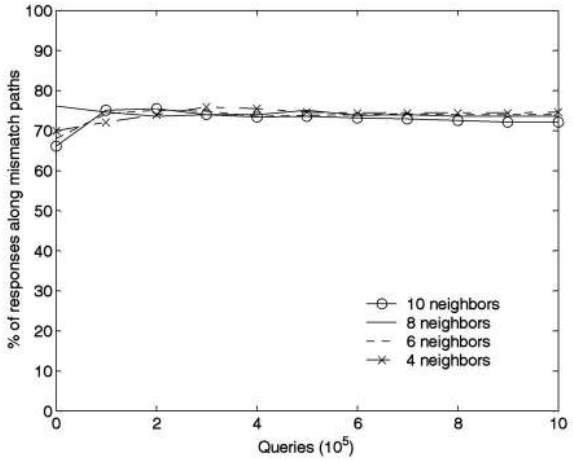

Fig. 3. The percent of query responses along mismatched paths.

is much longer than the link between nodes S and B or the link between nodes A and C in Fig. 2c. Clearly, in the inefficient mismatched overlay of Fig. 2a, the query message from source S will traverse the longest link SC three times to reach all other nodes (A, B, and C). This is a scenario of topology mismatch problem. If we can construct an efficient overlay, shown in Fig. 2b, the message needs to traverse all the physical links in Fig. 2c only once.

To quantitatively evaluate how serious the topology mismatch problem is in Gnutella-like networks, we simulate 1,000,000 queries on different Gnutella-like topologies with an average number of neighbors being 4, 6, 8, and 10. Detailed simulation methodology will be discussed in Section 4. In this simulation, we track the response of each query message to check if the response comes back along a mismatched path. We count a path as a mismatched path if a peering node in the path has been visited more than once. We plot the results in Fig. 3, which shows more than 70 percent of the paths suffer from the topology mismatch problem.

3.2 The Trade-Off between Traffic Cost and Query Response Time

To measure the quality of a P2P overlay topology in terms of search process, we introduce some popular 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. In unstructured P2P systems, the QoS of a search mechanism generally depends on the number of peers being explored (queried), response time, and traffic cost overhead. If a query reaches more peers, it is more likely that the requested object can be found. So, we use two performance metrics: average 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 [24] 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, 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.

The *response time* of a query is one of the parameters P2P users are concerned with. We define the response time of a query as the time period from when the query is issued until when the source peer received a response result from the first responder.

The trade-off between query traffic cost and response time has been discussed in [34]. P2P systems with a large number of average connections offer a faster search speed while increasing query traffic. Therefore, it is meaningless to merely optimize one metric without considering the other. Our goal is to design a distributed approach which reduces both traffic cost and response time.

3.3 Design of ACE

The proposed approach, Adaptive Connection Establishment (ACE), includes three phases, which are *neighbor cost table construction and exchanging, selective flooding, and overlay optimization*.

3.3.1 Phase 1: Neighbor Cost Table Construction and Exchanging

We use network delay between two peering nodes as a metric for measuring the cost between peers. We modify the Limewire implementation of Gnutella 0.6 P2P protocol by adding one routing message type. Each peer probes the costs with its immediate logical neighbors and forms a *neighbor cost table*. Two neighboring peers exchange their neighbor cost tables so that a peer can obtain the cost between any pair of its logical neighbors. Thus, a small overlay topology of a source peer and all its logical

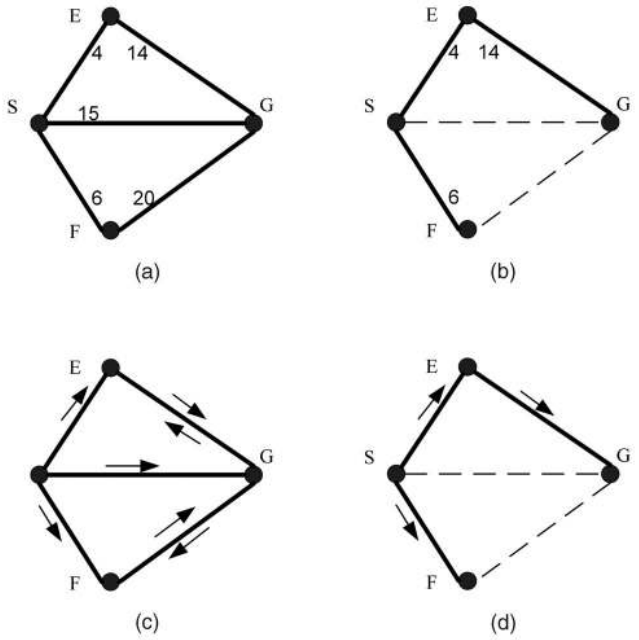

Fig. 4. Selective flooding.

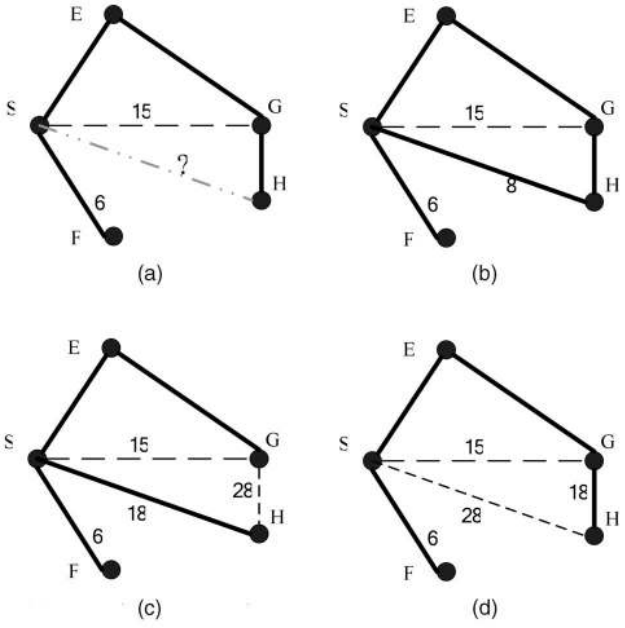

Fig. 5. Overlay optimization. (a) S probes G's neighbor H. (b) SH < SG, replace G by H. (c) SH > SG, but SH < GH, S keeps H as a direct neighbor. (d) SH > SG and SH > GH, S starts probing G's next neighbor.

neighbors is known to the source peer. If we use $\mathrm{N}(S)$ to denote the set of direct logical neighbors of peer S, each peer S has the information to obtain the overlay topology including S itself and $\mathrm{N}(S)$ as illustrated in Fig. 4a.

A critical issue to be examined in this phase is how often peers exchange their neighbors' cost table. In fact, there are two basic operations in this phase: Peers first probe the cost to their neighbors and construct the cost table; they then exchange the table with direct neighbors. In this design, there are two ways for each single peer to decide when to conduct neighbor probing and reporting, namely, periodic and event-driven. In the periodic approach, each peer conducts neighbor distance probing at every certain period of time, q. After probing the distances to all the neighbors, a peer sends the cost table to its neighboring peers. The value q is a critical factor for the performance of the periodic approach. In the event-driven approach, a peer produces and sends an updated cost table to its neighboring peers when there is a change on its logical connections with its neighbors, such as on a neighbor's leaving or on a peer's joining as its new neighbor. We have investigated the impact of the selection of the policies in ACE.

3.3.2 Phase 2: Selective Flooding (SF)

Based on the obtained neighbor cost tables, a minimum spanning tree (MST) among each peer S and its immediate logical neighbors ($S \cup \mathrm{N}(S)$) can be built by simply using an algorithm like PRIM, which has a computation complexity of $\mathrm{O}(m^2)$, where m is the number of logical neighbors of the source node. A computed MST is shown in Fig. 4b. Now, the message routing strategy of a peer is to select the peers that are the direct neighbors in the MST to send its queries, instead of flooding queries to all neighbors. We thus call peer S's direct neighbors in its MST *flooding-neighbors* of S and call those who are not direct neighbors in S's MST *nonflooding neighbors*. The connections between S and its flooding neighbors are defined as *forwarding connections*. See the example shown in Fig. 4. In Fig. 4c, the traffic incurred by node S's flooding of messages to its direct neighbor E, F, and G is: $4 + 14 + 14 + 15 + 6 + 20 + 20 = 93$. After SF, the *forwarding connections* are changed, as shown in Fig. 4d, and the total traffic cost becomes: $6 + 4 + 14 = 24$.

In Fig. 4d, peer S sends a message only to peers E and F and expects that peer E will forward the message to peer G. Note that, in this phase, even peer S does not flood its query message to peer G any more, S still retains the connections with G and keeps exchanging the neighbor cost tables with G. In this example, peer G is a *nonflooding neighbor* of peer S, which is the direct neighbor potentially to be replaced in phase 3. Peers E and F are *flooding neighbors* of S.

3.3.3 Phase 3: Overlay Optimization

This phase reorganizes the overlay topology. Note that each peer has a neighbor list which is further divided into *flooding neighbors* and *nonflooding neighbors* in Phase 2. Each peer also has the neighbor cost tables of all its neighbors. In this phase, a peer tries to replace those physically far away neighbors by physically close by neighbors, thus minimizing the topology mismatch traffic. An efficient method to identify such a candidate peer to replace a far away neighbor is critical to the system performance. Many methods may be proposed. In ACE, a nonflooding neighbor may be replaced by one of the nonflooding neighbor's neighbors.

The basic concept of phase 3 is illustrated in Fig. 5. In Fig. 5a, peer S is probing the distance to one of its *nonflooding neighbor* G's neighbors, for example, H. If SH is smaller than SG, as shown in Fig. 5b, connection SG will be cut. If SG is smaller than SH, but S finds that the cost between nodes G and H is even larger than the cost between peers S and H, as shown in Fig. 5c, S will keep H

as a new neighbor. Since the algorithm is executed in each peer independently, S cannot let G remove H from its neighbor list. However, as long as S keeps both G and H as its logical neighbors, we may expect that peer H will become a *nonflooding neighbor* to peer G after G's Phase 2 since peer G expects S to forward messages to H to reduce unnecessary traffic. Then, G will try to find another peer to replace H as its direct neighbor. After knowing that H is no longer a neighbor to G from periodically exchanged neighbor cost tables from node G (or from node H), S will cut connection SG, although S has already stopped sending query messages to G for a period of time, ever since the spanning tree was built for S. Obviously, if SH is larger than SG and GH, as shown in Fig. 5d, this connection will not be built and S will keep probing others of G's direct neighbors.

Let C_{ij} represent the cost from peer i to peer j. The following pseudocode describes the algorithm of Phase 3: overlay optimization for a given source peer i.

For each j in i's nonflooding neighbors
  Replaced = false;
    List = all j's neighbors excluding i;
  While List is not empty and Replaced = false
    randomly remove a peer h from List;
    measure C_{ih};
    if $C_{ih} < C_{ij}$ {replace j by h in i's neighbor list;
        Replaced = true;}
    else if $C_{ih} < C_{jh}$ {add h to i's neighbor list;
        remove j from i's neighbor list right after i
          finds out jh is disconnected;
        Replaced = true;}
  End While;
End For;

Recall that there are two different problems with Gnutella, which have been discussed in Section 3.1. The first one is the message duplications on overlay connections due to the flooding search. The second one is message duplications on physical links because of the topology mismatch problem. In fact, the second phase of ACE, selective flooding, focuses on improving inefficient flooding by reducing message duplications on the overlay level since the spanning tree can avoid short circles. The third phase of ACE, overlay optimization, tackles the second problem by replacing nonflooding neighbors (which are overlay neighbors but are far apart on the physical topology) by nearby neighbors to reduce the message duplications on physical links.

3.4 Property Analysis of ACE Operations

The strength of ACE optimization operation is that it reduces the total search traffic cost and query response time without shrinking the search scope of the queries. In other topology optimization approaches, the topology mismatch problem is attacked by simply letting all the peers keep replacing their direct neighbors with physically closer peers without considering the search scope issue. However, these types of approach may destroy the connectivity of the overlay and thus create many isolated islands in the P2P system.

Fig. 6 shows one such example in which peers A, B, C, D locate in the same AS, peers E, F, and H, G, K belong to other ASs, respectively. It is safe to assume that the physical distance between A and B or E and F is much smaller than that of K and A or C and F, as illustrated in Fig. 6. If the optimization policy for each node is to connect the closest peers while retaining the original number of logical neighbors, a connected graph may be broken into three components. As a result, all queries can only visit a small group of live peers in the system and the search scope of queries is significantly reduced, as shown in Fig. 6.

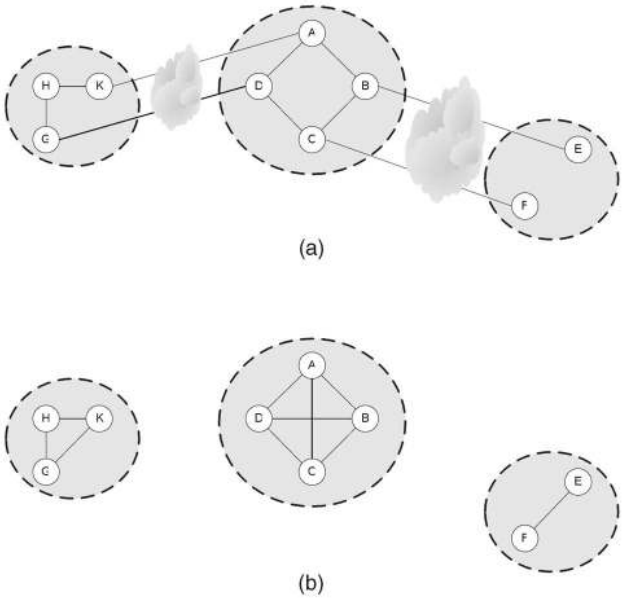

Fig. 6. Some naive approaches may disconnect the overlay topology.

We prove that ACE operations will not increase the number of components of a graph.

Theorem. *Given a graph $G = (V, E)$, the ACE optimization operations will not increase G's component number.*

Proof. We prove by contradiction. Suppose our claim is false. Then, there exists at least one component C, where C is a subgraph of G, which could be disconnected by the ACE operations. Suppose C is disconnected into two parts, D and H, after ACE operations, as shown in Fig. 7. Before the ACE optimization, there must be one or more edges between D and H since C is connected. Let M denote the set of the edges between D and H. Among all these edges in M, we choose the shortest one, $uv \in M$. Here, we assume that there are no exact equal length edges in the system, so uv is the only shortest edge in M. The graph C is disconnected after ACE operations means that none of the edges in M, including uv, is selected as a forwarding connection, either at peer u or peer v. Without loss of generality, let us assume that uv is disconnected by peer u. We know that peer u employs an MST algorithm, such as the Kruskal algorithm, in SF operation. Because v is u's one hop neighbor, v must be included in u's MST. In the Kruskal algorithm, edges are sorted from shortest to longest. That edge uv is not selected by MST means that there is already another path P ($uv \notin P$) between u and v and the length of each edge in P is shorter than uv. As P is between D and H, at least one of the edges in P, say edge e, belongs to M. We

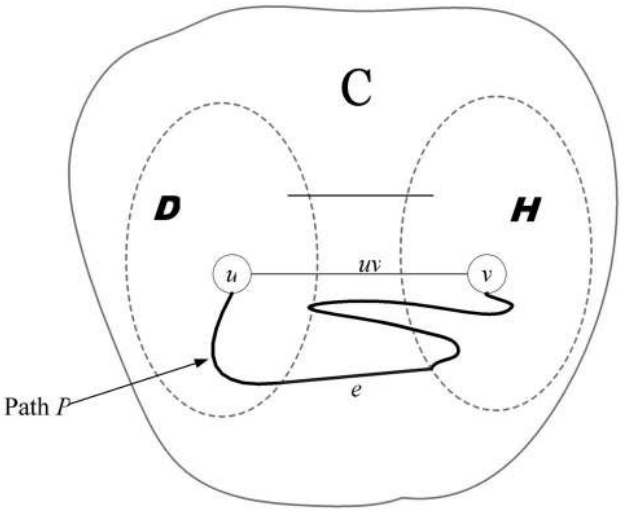

Fig. 7. Proof of the property of ACE operations.

then have $e < uv$, which is a contradiction to our choice that uv is the shortest edge in M and, thus, the theorem is proven. □

3.5 Depth of Optimization

In the ACE described in Section 3.3, the optimization is conducted among each source peer and all its direct logical neighbors. We expect better performance if the optimization can be done in a larger scope with more peers. We define the *h-neighbor closure* of a source peer as the set of peers within h hops from the source peer. For example, a 2-neighbor closure includes the source peer, all its direct neighbors, and all the neighbors of the direct neighbors. The optimization in the initial ACE (Section 3.3) is only conducted within 1-neighbor closure. We can enlarge the optimization scope by increasing the value of h, which is also called as the *depth of optimization*. A larger value of h leads to a better topology matching improvement, but a higher overhead due to the extra information exchanging. We will conduct further studies in this direction with the aim of reaching a good performance level by considering the trade-offs between the topology optimization improvement and the information exchange overhead.

The optimization procedure in h-neighbor closure is similar to that in 1-neighbor closure described in Section 3.3 except for the neighbor cost table exchanging. Each peer builds a neighbor cost table. The difference is that each cost table associates with a TTL value initialized as h. Each peer sends out its own neighbor cost table and the cost tables it received from other peers with $\mathrm{TTL} > 0$. When a peer receives one or more cost tables, the TTLs of the tables will be decremented by 1. In such a way, each peer will have the knowledge of the topology around it up to h hops. A minimum spanning tree can be built among this source peer and its *h-neighbor closure*. Phase 3 described in Section 3.3 can be used to replace nonflooding neighbors.

Fig. 8 illustrates the overlay trees constructed for each peer within 1-neighbor closures. In Fig. 8, peer A initiates a query. The bold links denote the links on the tree and the arrows indicate the query directions. The query is sent from peer A to

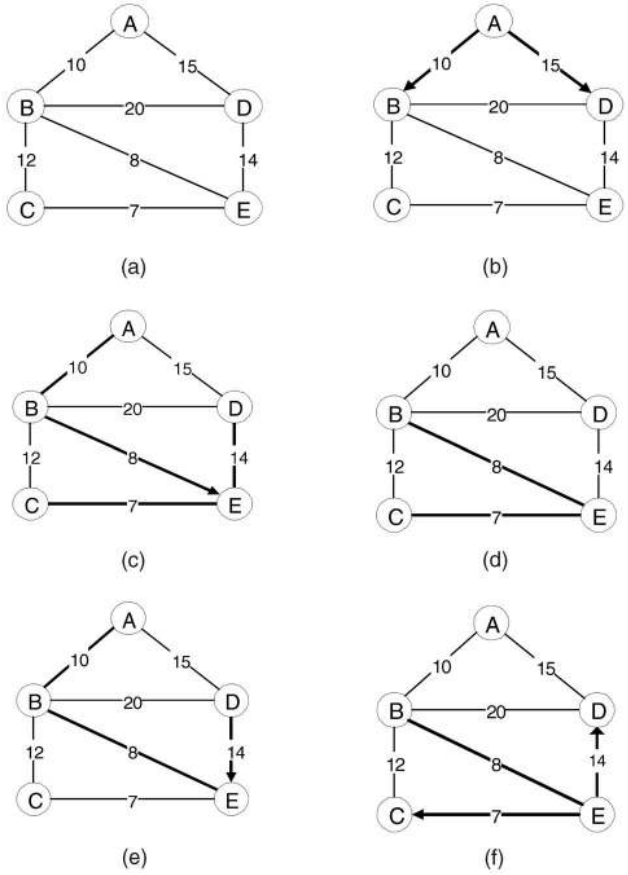

Fig. 8. Overlay trees built in 1-neighbor closure. (a) Original topology. (b) Overlay tree rooted at A. (c) Overlay tree rooted at B. (d) Overlay tree rooted at C. (e) Overlay tree rooted at D. (f) Overlay tree rooted at E.

B and D since both B and D are direct logical neighbors of A on the overlay tree. Peer B then forwards the query to E and D forwards the query to E. Peer E finally forwards the query to D and C. Peer C will not forward the query because only E is its direct neighbor, but E is the peer which forwards the query to C. So, the query process terminates. The query paths and corresponding costs for this query are listed in Table 1. The total cost for this query from peer A to be forwarded to all other peers through the overlay trees built in 1-neighbor closures is 68. The number of unnecessary messages is reduced from three to one compared with blind flooding in this example. In 1-neighbor closure, the query message traverses one path twice, which is E-D. In blind flooding, the

TABLE 1
Query Paths and Costs on Overlay Trees Built in 1-Neighbor Closure

Query Path		
From	To	Corresponding Cost
A	B, D	10+15=25
B	E	8
D	E	14
E	C, D	7+14=21
Total Cost		68

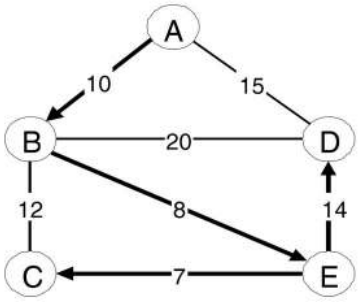

Fig. 9. Overlay tree built in 2-neighbor closure.

same query message traverses 3 paths twice, which is B-D, D-E, and C-E.

Fig. 9 and Table 2 illustrate the overlay tree built in 2-neighbor closure and the corresponding query direction and cost. The total cost to forward a query from peer A to all other peers is 39. No path is traversed twice by ACE with $h = 2$ in this example. We can see that the number of unnecessary messages and the total traffic is decreased as the value of h is increased.

TABLE 2
Query Paths and Costs on the Overlay Tree Built in 2-Neighbor Closure

Query Path		
From	To	Corresponding Cost
A	B,	10
B	E	8
E	C, D	7+14=21
Total Cost		39

4 Simulation Methodology

We describe the topology generation, performance metrics used in our simulations, our simulation setup, and parameter settings in this section.

4.1 Simulation Setup

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 [30] and P2P overlay topologies [25] follow the small world and power law properties. Power law describes the node degree, while small world describes characteristics of path length and clustering coefficient [7]. The study in [25] 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 to generate topologies based on the AS Model. Using BRITE, we generate three physical topologies, each with 27,000 nodes. The logical topologies are generated with the number of peers (nodes) ranging from 2,000 to 9,000. The average number of neighbors of each node ranges from 4 to 10. We simulate ACE for all the generated logical topologies on top of each of the three generated physical topologies. We also simulate this approach in a real-world P2P topology (based on DSS Clip2 trace). We obtained consistent results on the real-world topology and the generated topologies.

The content popularity of a publisher follows a Zipf-like distribution (aka Power Law) [4], [6], where the relative probability of a request for the ith most popular page is proportional to $1/i^{\alpha}$, with α 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 follows a more general Zipf-like distribution. Query word frequency does not follow a Zipf distribution [31]. The user's query lexicon size does not follow a Zipf distribution [31] but with a heavy tail. Both the overall traffic and the traffic from the 10 percent most popular nodes are heavy-tailed in terms of the host connectivity, traffic volume, and average bandwidth of the hosts [28]. Studies in [9] 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 [9], which could be approximated by two Zipf distributions.

In our simulation, we simulate the flooding search used in a 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. TTLs are used to propagate the queries.

4.2 Performance Metrics

As we discussed in Section 3.2, we should consider traffic cost as well as query response time in P2P systems. To evaluate the search efficiency of the systems, besides traffic cost and response time, we also use the following metrics:

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.

Optimization rate is defined as gain/penalty ratio, i.e., the ratio of query traffic reduction and overhead traffic increment in order to study the trade-offs between query traffic and overhead traffic by changing the value of optimization depth of h. One major factor to impact the traffic overhead is the frequency of exchanging cost information. We define *frequency ratio*, R, as the ratio of query frequency to use the overlay trees to the frequency of cost information changes. For a given P2P network topology, if the frequency of the topology and cost changes and query frequency can be measured so that R is determined, we should be able to adjust the value of h to

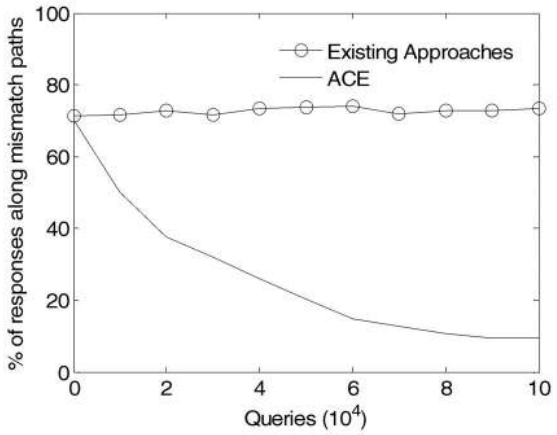

Fig. 10. The percentage of query responses along mismatched paths of existing schemes and ACE.

achieve optimal gain/penalty ratio. ACE is worth to use only if the gain/penalty ratio is larger than 1.

4.3 A Dynamic P2P Environment

P2P networks are highly dynamic with peers joining and leaving frequently. The observations in [28] 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 [25] indicated that the median up-time for a node in Gnutella and Napster is 60 minutes. Studies in [5] 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 over 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 [29], 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 [25]. The mean of the distribution is chosen to be 10 minutes [28]. 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.

5 Performance Evaluation

We have simulated ACE for all the generated logical topologies on top of each of the 10 generated physical topologies with 27,000 nodes. We representatively present the results in this section.

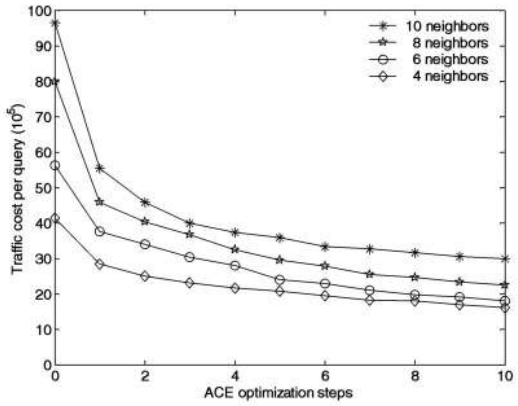

Fig. 11. Traffic reduction versus the optimization step.

5.1 ACE in Static Environments

In our first simulation, we study the effectiveness of ACE in a static P2P environment where the peers do not join and leave frequently.

In Fig. 10, we show the percentage of query responses along mismatched paths of the other existing approaches and ACE scheme. Recall the results in Fig. 3 show that 70 percent of the queries are back along mismatched paths in Gnutella like P2Ps. As most of the existing approaches did not deal with the topology mismatch problem, the mismatch degree is not changed under those advanced approaches. ACE is proven an effective approach optimizing up to 60 percent out of the 70 percent mismatched paths. As a result, system performance is improved significantly.

The first goal of ACE schemes is to reduce traffic cost as much as possible while retaining the same search scope. Fig. 11 shows the traffic cost reduction of ACE, where the curve of "c_n *neighbors*" means the average traffic cost incurred by a query to cover the search scope in the x-axis and, in the system, the average number of logical neighbors is c_n. We can see that the traffic cost decreases when ACE is conducted multiple times, where the search scope is all of the 9,000 peers. ACE may reduce traffic cost by around 65 percent and it converges in around 10 steps. One step means one round of ACE optimization including the three phases, i.e., neighbor cost table exchanging, selective flooding, overlay optimization.

The simulation results in Fig. 12 show that ACE can shorten the query response time by about 35 percent after 10 steps. One of the strengths of ACE schemes is that they reduce both query traffic cost and response time without decreasing the query success rate. We vary the size of the overlay topology ranging from 1,000 to 9,000, and the results are consistent. It is safe to say that ACE is completely distributed and scalable and the size of P2P overlays has no impact on the performance of ACE.

We also plot the node-degree's pdf distribution before and after ACE optimization in a 1,000 peer overlay in Fig. 13. We can see that optimized logical topology keeps the similar branching factor (i.e., the average number of logical neighbors) property as the original logical topology and the query coverage range can be guaranteed.

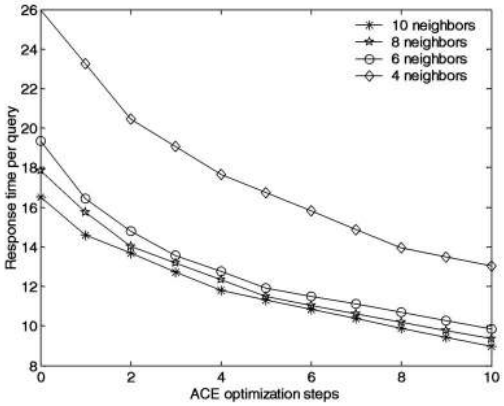

Fig. 12. Average response time versus the optimization step.

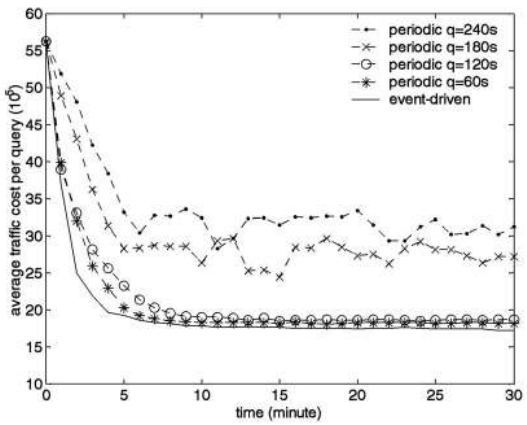

Fig. 14. Traffic cost reduction of ACE in a dynamic P2P environment.

5.2 ACE in Dynamic Environments

We further evaluate the effectiveness of ACE in dynamic P2P systems. In this simulation, we assume that peer average lifetime in a P2P system is 10 minutes; 0.3 queries are issued by each peer per minute. The overlay we are using has 9,000 peers with each peer having six neighbors on average.

The first key issue we examined in this simulation is how each peer decides when to conduct the neighbor cost probing and reporting operation. In our design, there are two ways, namely, periodic and event-driven, as we mentioned in Section 3.3.1. In the periodic approach, each peer conducts neighbor distance probing at every certain period of time, q. After probing the distances to all the neighbors, a peer sends the cost table to its neighboring peers. The major advantage of this policy is its simplicity. In the event-driven approach, a peer produces and sends an updated cost table to its neighboring peers only if there is a change in its logical connections with its neighbors, such as on a neighbor's leaving or on a peer's joining as its new neighbor.

The value q is a critical factor for the performance of periodic approach. We have investigated the impact of different values of q ranging from 30s to 300s. Figs. 14 and 15 show the results on some representative samples of q at 60s, 120s, 180s, and 240s, respectively, where the x-axis indicates the time elapsed since the first probing or event occurred. A small q leads to a fast convergent speed. However, if q is too small, peers will conduct the optimization operations too often, making the overhead keep growing when the reduction of the traffic cost and response time have already reached a threshold. On the other hand, if q is too large, e.g., $q = 240$, the frequency of the optimization operations will not be enough to catch the changes of peers' frequent joining and leaving. Thus, the convergent speed is slow and the reduction of traffic cost and response time is limited. Figs. 14 and 15 suggest that $q = 60\text{s}$ and $q = 120\text{s}$ make ACE have very close performances, while, when $q = 180\text{s}$ and $q = 240\text{s}$, ACE does not work well. Considering there are other overheads of ACE except traffic cost, we select $q = 120\text{s}$, which means each single peer will probe its neighbors and report the cost table every two minutes. ACE has about 60 percent reduction on traffic cost and 40 percent reduction on response time for this simulation setup with the given physical topology, average peer lifetime (10 minutes), and query frequency (0.3/minute).

Figs. 14 and 15 also show that the event-driven policy outperforms periodic policy in ACE. We did not select event-driven policy as the first priority in ACE based on the following observations: First, the overall performances of these two policies are very close if we carefully choose the q value for periodic policy. Second, compared with periodic

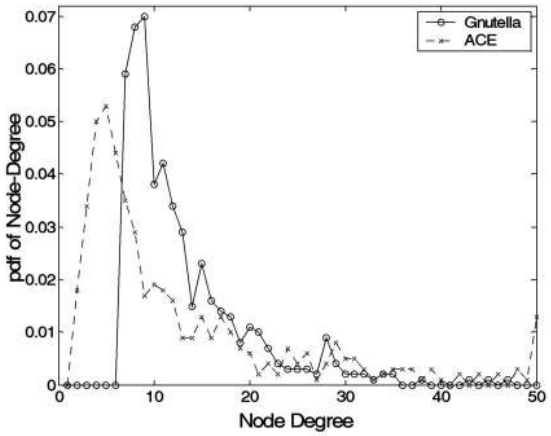

Fig. 13. Comparison of topology properties.

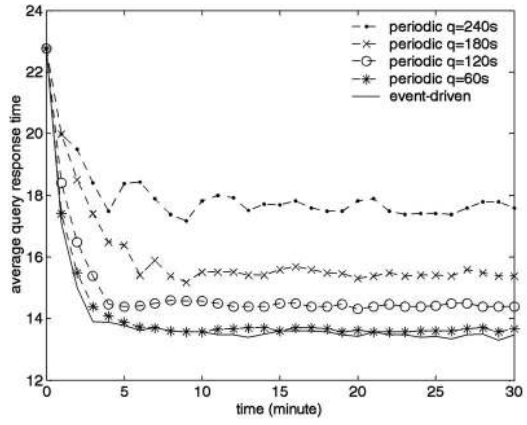

Fig. 15. Response time reduction in a dynamic P2P environment.

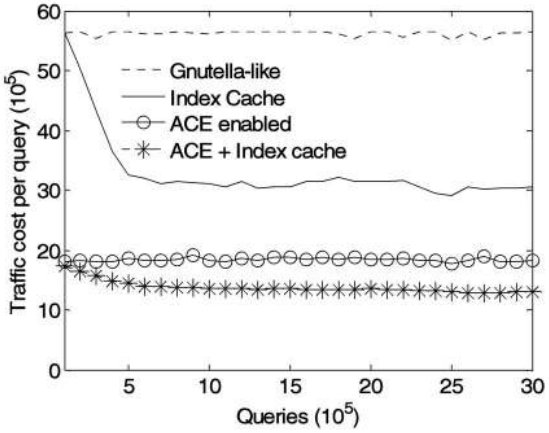

Fig. 16. The effectiveness of ACE plus the Response Index Caching Scheme on reduction of traffic cost.

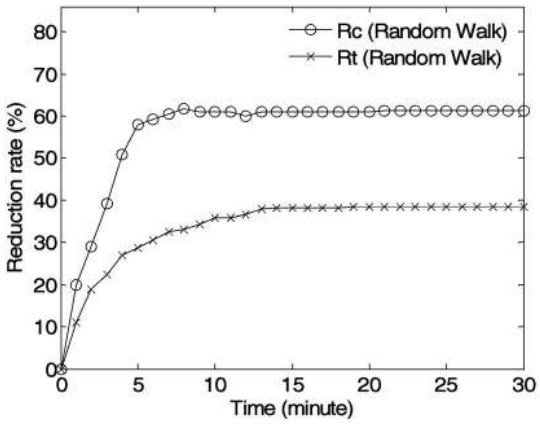

Fig. 18. Traffic cost and average response time reduction on the Random Walk scheme when employing ACE.

policy, to employ event-driven policy needs more operations. For example, peers must monitor all the neighboring peers' behaviors, which means extra overhead, especially when some peers have a very short lifetime in the P2P network. In contrast, to employ a periodic policy at a peer is simple and straightforward: Do the probing and reporting in a given frequency. However, if ACE is employed among super-peers only, the event-driven policy could be a good option because super-peers tend to have a longer lifetime.

As we have done in static environments, we vary the size of overlays from 2,000 to 9,000 peers and the optimization rate of ACE remains consistent.

We claim the strength of ACE is that ACE is orthogonal with existing advance search approaches. In a dynamic P2P environment, we simulate strategies of combining ACE with other approaches, such as response index caching scheme [29] and Random Walk scheme [17].

In Figs. 16 and 17, we show a strategy of combining ACE with the response index caching scheme [29] in which query responses are cached in passing peers along the returning path. 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. We compare the traffic cost and response time in a Gnutella-like system without any optimization (Gnutella-like), with response index caching only (Index Cache), with ACE optimization only (ACE enabled), and with ACE optimization plus response index caching (ACE + Index Cache). The results in Figs. 16 and 17 show that, by combining ACE and response index caching, the traffic cost is reduced by about 75 percent without shrinking the search scope, and the average query response time is reduced by about 70 percent.

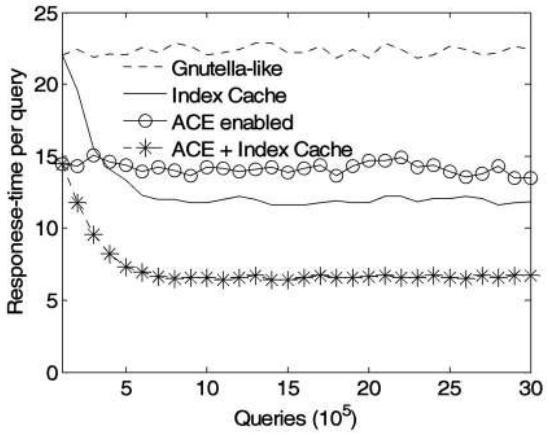

Fig. 17. The effectiveness of ACE plus the Response Index Caching Scheme on reduction of query response time.

We then use the Random Walk scheme instead of blind flooding during the search in ACE enabled P2Ps. In the Random Walk scheme [17], 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. In our simulation, 30 walks are issued for each query. We repeat this simulation with different random seeds for 20 times and plot the average. Here, we use two metrics, traffic cost reduction rate $(R_c(*))$ and response time reduction rate $(R_t(*))$. $R_c(*)$ is defined by

$$R_c(*) = \frac{C(RandomWalk) - C(ACE)}{C(RandomWalk)} \times 100\%,$$

where C(RandomWalk) represents the traffic cost incurred by searching all the peers using the Random Walk approach in a Gnutella-like overlay. C(ACE) represents the traffic cost incurred in ACE enabled P2P networks. $R_t(*)$ is given by

$$R_t(*) = \frac{T(RandomWalk) - T(ACE)}{T(RandomWalk)} \times 100\%,$$

where T(RandomWalk) denotes the average response time of queries using Random Walk approach in Gnutella-like overlays and T(ACE) is that of ACE enabled systems.

We plot $\mathrm{R_c(RandomWalk)}$ and $\mathrm{R_t(RandomWalk)}$ in Fig. 18. As expected, ACE reduces the traffic cost and response time of the Random Walk scheme by approximately 60 percent and 38 percent, respectively.

5.3 The Impact of Optimization Depth

Fig. 19 shows the query traffic reduction rate over blind flooding versus the depths of neighbor closure to construct overlay trees. Different curves correspond to the performance

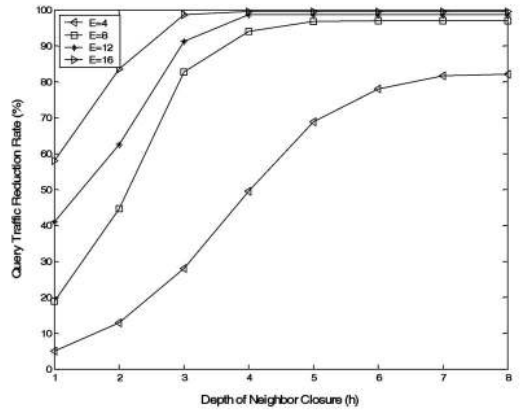

Fig. 19. Query traffic reduction rate.

on different topologies with different values of E, where E is the average number of neighbors. For a given depth of neighbor closure, the reduction rate increases with increased average number of neighbors. For a given average number of neighbors, the reduction rate also increases as the depths of neighbor closure increases. There is a threshold of depth for each E, from which it is hard to further reduce the query traffic.

Fig. 20 shows the overhead traffic versus the depth of neighbor closure. The overhead traffic increases as the depths of neighbor closure increases or as the average number of neighbors increases.

Figs. 21 and 22 show the optimization rate versus the depth of neighbor closure with $E = 16$ and $E = 4$, respectively. Different curves in each figure correspond to different values of R. Based on this figure, we can determine, for a given value of R, the minimal value of h to achieve performance gain in ACE. The minimal value of h is defined as the value of h that leads to an optimization rate of 1.

To achieve performance gain, we should choose the depth values that can lead to optimization rates that are greater than 1. When we increase R, the optimization rate increases for a given depth value (h) and the minimal value of h to achieve performance gain decreases. As h increases, the optimization rate also increases. However, there is a threshold of h from which the optimization rate is hard to

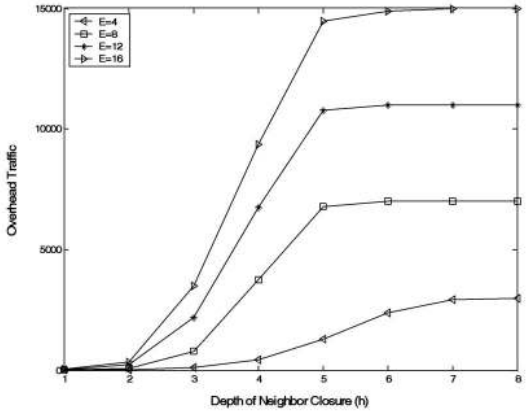

Fig. 20. Overhead traffic.

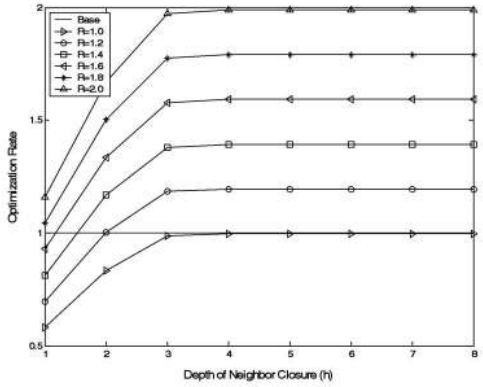

Fig. 21. Optimization rate versus h (E = 16).

increase anymore. Figs. 21 and 22 also show that, for a large value of E, a small minimal value of h is needed to achieve performance gain for a given value of R.

Figs. 23 and 24 show the optimization rate versus frequency ratio with $E = 16$ and $E = 4$, respectively. When the value R increases, the optimization rate significantly increases. A large value of R means that the query frequency is high and the tree reconstruction frequency is low. For a given network after a period of time, if we can find a relatively stable value of R, we will be able to find a minimal value of h to construct overlay trees and achieve performance gain in ACE. We can see from Fig. 23 that, for $R = 1$, the optimization rate is always less than 1. Thus, using ACE under an environment with $R = 1$, the given topology will not improve any performance. From Fig. 23, the minimal value of h is 2 for $R = 1.5$ and is 1 for $R = 2$. Comparing Fig. 23 with Fig. 24, for the same value of R, the minimal value of h is small for a large value of E. For example, for $R = 2$, the minimal value of h is 1 for $E = 16$, while the minimal value of h is 5 for $E = 4$. Thus, ACE is more effective in a topology with high connectivity density.

6 Conclusion

In this paper, we propose a distributed approach to solving overlay topology mismatch problem. Our simulation shows that the average cost of each query to reach the same scope of nodes is reduced by about 65 percent when using our

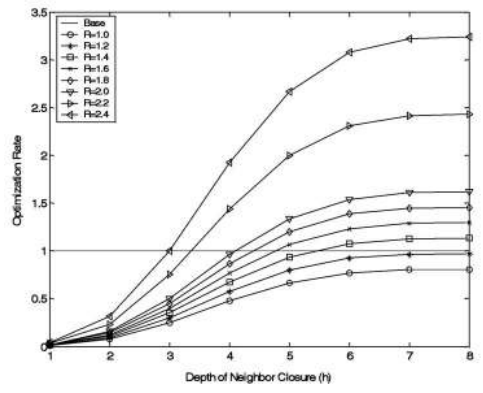

Fig. 22. Optimization rate versus h (E = 4).

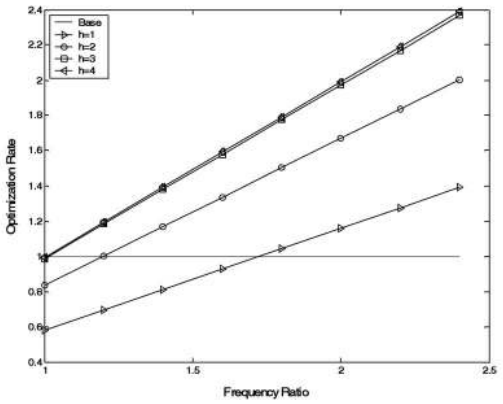

Fig. 23. Optimization rate versus frequency rate (E = 16).

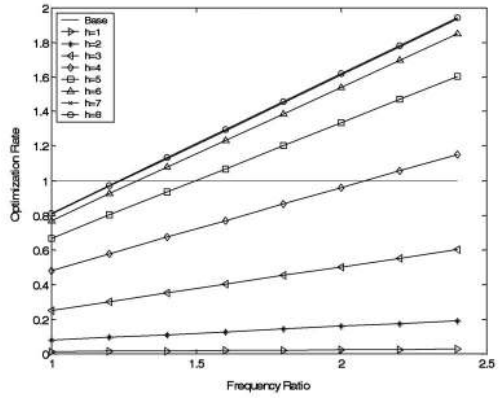

Fig. 24. Optimization rate versus frequency rate (E = 4).

proposed ACE in a Gnutella-like P2P network, without losing any autonomy feature, and the average response time of each query can be reduced by about 35 percent. The impacts of the frequency of exchanging neighbor cost tables on ACE performance have also been studied. ACE's ability to complement other advanced search approaches has been shown by two combination strategies of ACE with query response index caching and random walk. The proposed ACE technique is fully distributed, easy to implement, and adaptive to the dynamic nature of P2P systems. Furthermore, a larger diameter leads to a better topology optimization rate and a higher overhead due to extra information exchanging. ACE is more effective in a topology with high connectivity density. It will make the decentralized flooding-based P2P file sharing systems more scalable and efficient.

It is very important for ACE to quickly identify the best candidate from a nonflooding neighbor's neighbor list to minimize replacement overhead. In our simulations, we only use random policy to replace a nonflooding neighbor by a random selected candidate. We are studying several alternatives to choose the candidate. For example, the naive policy simply disconnects the source node's most expensive neighbor. The source node will probe the costs to some other nodes and try to find a less expansive node as a replacement of the disconnected neighbor. The second one is the closest policy in which the source will probe the costs to all of the nonflooding neighbor's neighbors and select the closest one.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grant CCF-0325760, by Microsoft Research Asia, and by Hong Kong RGC Grants DAG 04/ 05.EG01, and HKUST6264/04E. Zhenyun Zhuang participated in the early stage of the work.

REFERENCES

[1] BRITE, http://www.cs.bu.edu/brite/, 2003.
[2] Gnutella, http://gnutella.wego.com/, 2003.
[3] KaZaA, http://www.kazaa.com, 2003.
[4] 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.
[5] R. Bhagwan, S. Savage, and G.M. Voelker, "Understanding Availability," *Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '03),* 2003.
[6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM,* 1999.
[7] T. Bu and D. Towsley, "On Distinguishing between Internet Power Law Topology Generators," *Proc. IEEE INFOCOM,* 2002.
[8] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM,* 2003.
[9] J. Chu, K. Labonte, and B. Levine, "Availability and Locality Measurements of Peer-to-Peer File Systems," *Proc. ITCom: Scalability and Traffic Control in IP Networks II Conf.,* 2002.
[10] Y. Chu, S.G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM SIGMETRICS,* 2000.
[11] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," *Proc. ACM SIGCOMM,* 2002.
[12] P. Ganesan, Q. Sun, and H. Garcia-Molina, "Apocrypha: Making P2P Overlays Network-Aware," technical report, Stanford Univ., 2004.
[13] B. Krishnamurthy and J. Wang, "Topology Modeling via Cluster Graphs," *Proc. SIGCOMM Internet Measurement Workshop,* 2001.
[14] Y. Liu, X. Liu, L. Xiao, L.M. Ni, and X. Zhang, "Location-Aware Topology Matching in Unstructured P2P Systems," *Proc. IEEE INFOCOM,* 2004.
[15] Y. Liu, Z. Zhuang, L. Xiao, and L.M. Ni, "AOTO: Adaptive Overlay Topology Optimization in Unstructured P2P Systems," *Proc. IEEE GLOBECOM,* 2003.
[16] Y. Liu, Z. Zhuang, L. Xiao, and L.M. Ni, "A Distributed Approach to Solving Overlay Mismatch Problem," *Proc. 24th Int'l Conf. Distributed Computing Systems (ICDCS),* 2004.
[17] 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.
[18] 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.
[19] D.A. Menasce and L. Kanchanapalli, "Probabilistic Scalable P2P Resource Location Services," *ACM SIGMETRICS Performance Evaluation Rev.,* vol. 30, no. 2, pp. 48-58, 2002.
[20] A. Nakao, L. Peterson, and A. Bavier, "A Routing Underlay for Overlay Networks," *Proc. ACM SIGCOMM,* 2003.
[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] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing,* 2002.
[24] J. Ritter, "Why Gnutella Can't Scale. No, Really," http://www.tch.org/gnutella.html, 2001.
[25] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking (MMCN),* 2002.
[26] 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.

[27] 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.
[28] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. ACM SIGCOMM Internet Measurement Workshop,* 2002.
[29] K. Sripanidkulchai, "The Popularity of Gnutella Queries and Its Implications on Scalability," http://www2.cs.cmu.edu/kunwadee/research/p2p/gnutella.html, 2001.
[30] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topol-ogy Generators: Degree-Based vs. Structural," *Proc. SIGCOMM '02,* 2002.
[31] Y. Xie and D. O'Hallaron, "Locality in Search Engine Queries and Its Implications for Caching," *Proc. IEEE INFOCOM,* 2002.
[32] 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.
[33] B. Yang and H. Garcia-Molina, "Efficient Search in Peer-to-Peer Networks," *Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS),* 2002.
[34] 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 (ICPP),* 2003.

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, overlay systems and applications, system resource management, and design and implementation of experimental algorithms. She is a member of the ACM, the IEEE, the IEEE Computer Society, and IEEE Women in Engineering.

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, grid computing, and high-speed networking. He is a member of the IEEE and the IEEE Computer Society.

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, Dr. Ni has chaired many professional conferences and has received a number of awards for authoring outstanding papers.