Building a Scalable Bipartite P2P Overlay Network

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

Abstract—The Peer-to-Peer (P2P) model, being widely adopted in today's Internet computing, suffers from the problem of topology mismatch between the overlay networks and the underlying physical network. Traditional topology optimization techniques identify physically closer nodes to connect as overlay neighbors, but could significantly shrink the search scope. Recent efforts have been made to address the mismatch problem without sacrificing the search scope, but they either need time synchronization among peers or have a low convergent speed. In this paper, we propose a scalable bipartite overlay (SBO) scheme to optimize the overlay topology by identifying and replacing the mismatched connections. In SBO, we employ an efficient strategy for distributing optimization tasks in peers with different colors. We conducted comprehensive simulations to evaluate this design. The results show that SBO achieves approximately 85 percent of reduction on traffic cost and about 60 percent of reduction on query response time. Our comparisons with previous approaches to address the topology mismatch problem have shown that SBO can achieve a fast convergent speed, without the need of time synchronization among peers.

Index Terms—Unstructured peer to peer, topology mismatch, overlay, bipartite, search efficiency.

1 Introduction

The peer-to-peer (P2P) model, such as Gnutella [1], KaZaA [3], and BitTorrent [6], [21], aims at utilizing and managing increasingly large and globally distributed information and computing resources, thus complementing available client-server services. Decentralized and unstructured P2P systems are most commonly deployed in today's Internet. File placement is random in these systems, which has no correlation with the network topology. The most popular search mechanism in use is to blindly "flood" a query to the network among peers (such as in Gnutella) or among superpeers (such as in KaZaA). A query is broadcast and rebroadcast until a certain criterion is satisfied.

Based on their measurements of popular P2P systems such as FastTrack (including KaZaA and Grokster) [2], Gnutella, and DirectConnect, the studies in [21], [23] have shown that P2P traffic contributes the largest portion of the Internet traffic. Much P2P traffic, however, is unnecessary. An attractive feature of P2P is that peers do not need to directly interact with the underlying physical network, providing many new opportunities for user-level development and applications. Nevertheless, the mechanism for a peer to randomly choose logical neighbors, without any knowledge about the physical topology, causes a serious topology mismatch between the P2P overlay networks and the physical network. Our studies, detailed in Section 5, have shown that about 75 percent of the query response paths suffer from the topology mismatch problem. Because of the mismatch problem, a pair of logical neighbors can be far away from each other, causing a message to traverse the same physical link multiple times and wasting a huge amount of network bandwidth.

Traditional overlay topology optimization studies that use different techniques to identify physically closer nodes to connect as overlay neighbors could significantly shrink the search scope, which is not feasible in P2P systems. Recent efforts have been made to address the mismatch problem without sacrificing the search scope, such as Adaptive Connection Establishment (ACE) [18] and Location-aware Topology Matching (LTM) [19]. In ACE, each peer exchanges information with its neighbors so that an overlay multicast tree can be built among each source peer and its neighbors as the base for the overlay optimization. The convergent speed of ACE is slow because of the information exchange among each peer and all its neighbors. In LTM, each peer issues a detector so that the peers receiving the detection can record relative delay information as the optimization basis. In order to record the correct delay information, time synchronization among peers is needed.

In order to address the limits of existing solutions for the overlay mismatch problem, we propose a scalable bipartite overlay (SBO) among peers in Gnutella-like systems or among the superpeers in KaZaA-like systems to optimize the overlay topology by identifying and replacing the mismatched connections. In SBO, we employ an efficient strategy for distributing optimization tasks to peers with different colors, that is, white and red. The color of a peer is assigned at the bootstrapping stage, and a peer will only be connected with peers of a different color. A white peer probes the cost with its neighbors and sends the cost information to its red neighbors. With the cost information of peers within two hops, each red peer will build a minimum spanning tree (MST), which is the base for a white peer to replace or cut mismatched connections. Our

- Y. Liu and L.M. Ni are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: {liu, ni}@cse.ust.hk.
- 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.

Manuscript received 14 Oct. 2005; revised 3 Nov. 2006; accepted 9 Nov. 2006; published online 9 Jan. 2007.
Recommended for acceptance by K. Hwang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0434-1005.
Digital Object Identifier no. 10.1109/TPDS.2007.1059.

simulation studies show that the total traffic and response time of the queries can be significantly reduced by optimized SBO without shrinking the search scope. Our comparisons with ACE and LTM have also shown that SBO can achieve a fast convergent speed, without the need of time synchronization among peers.

The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 analyzes the inefficient P2P overlay topologies. Section 4 presents the design of SBO and its optimization operations. Performance evaluation of SBO is presented in Section 5, and we conclude the work in Section 6.

2 Related Work

Different techniques have been used to build efficient overlay topologies. End-system multicast Narada was proposed in [11], which first forms a rich connected graph on which shortest path spanning trees are constructed. Each tree is rooted at the corresponding source by using well-known routing algorithms. This approach introduces a large overhead in forming the graph and trees in a large scope and does not consider the peers' dynamic characteristics of joining and leaving. The overhead of Narada is proportional to the multicast group size. Researchers have also considered clustering close peers based on their Internet Protocol (IP) addresses (for example, [14], [20]). We believe that there are two limitations in this approach. First, the mapping accuracy is not guaranteed. Second, this approach might affect the search scope of a P2P network. In contrast, our technique is measurement based and can accurately and dynamically connect physically closer peers and disconnect distant ones. Also, our scheme does not shrink the search scope. Researchers in [31] proposed measuring the latency between each peer and 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. In contrast, our measurement is conducted locally with high accuracy, significantly reducing the network traffic. Chawathe et al. [10] introduced a topology adaptation algorithm to ensure that high-capacity nodes are the ones with high degree, and low-capacity nodes are within the 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.

To attack the topology mismatch problem, the LTM scheme [17] is proposed, 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 its direct neighbors. The major drawback of LTM is that it needs to synchronize all the peering nodes and, thus, LTM requires the support of the Network Time Protocol (NTP) [5], which is critical.

One of our early attempts at alleviating topology mismatch is called ACE [18], in which every single peer builds an overlay MST among itself (source node) and the peers within a certain diameter from the source peer and

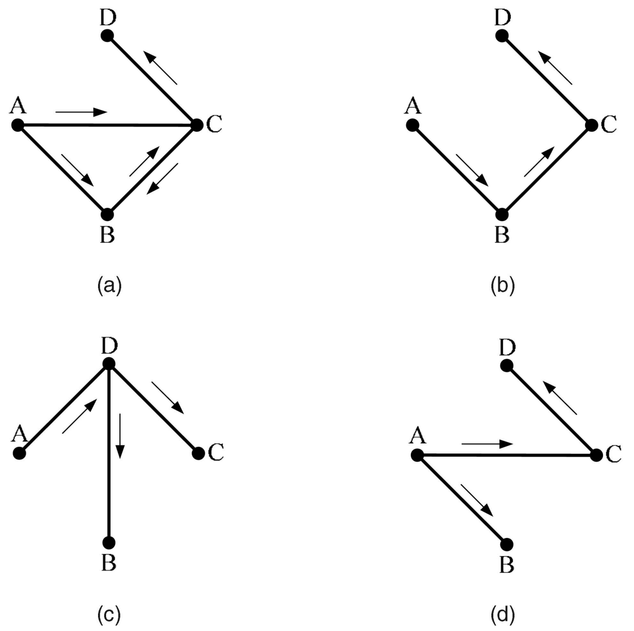


Fig. 1. Examples of P2P overlay topologies.

then optimizes the neighbor connections that are not on the tree. However, ACE can only work with one-hop logical neighbors, and the convergent speed is relatively slow. Later discussions will show that our SBO proposed in this paper has a smaller overhead and a faster convergent speed than ACE.

3 Inefficient Overlay Topologies

In a P2P system, all participating peers form a P2P network on top of an underlying physical network. A P2P network is an abstract logical network called an overlay network. The maintenance and search operations of a Gnutella peer are specifically described in [7]. 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 some of these peers. If some attempts succeed, then the connected peers will be the new peer's neighbors. Once this peer joins 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 rejoin the P2P network, it will try to connect to the peers whose IP addresses have already been cached. The mechanism by which a peer joins a P2P network, with peers randomly joining and leaving, and the nature of a flooding-based search make an inefficient mismatched overlay network and cause large amounts of unnecessary traffic. In this section, we use examples to explain the message duplications and the mismatch problem.

3.1 Unnecessary Message Duplications in Overlay Connections

Fig. 1 illustrates some examples of P2P overlay topologies, where solid lines denote overlay connections among logical P2P neighbors. Consider the case when node A issues a

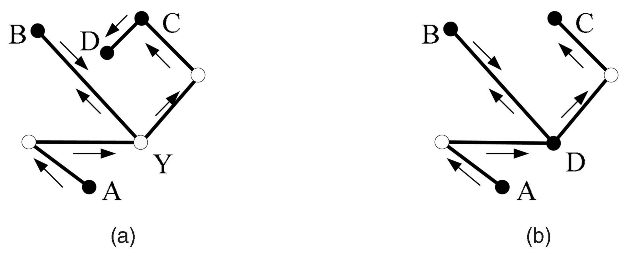


Fig. 2. Example of physical topologies.

query. A solid arrow represents a delivery of the query message along one logical connection. In Gnutella, a peer forwards an incoming query message to all of its directly connected peers, except the one that delivered the incoming query. Thus, as shown in Fig. 1a, A's query is relayed by nodes B and C. Peer B forwards the query to C, whereas C also forwards the query to B. In this case, the pair of transmissions between B and C causes an unnecessary message duplication. We can easily observe that the other three overlays, as shown in Figs. 1b, 1c, and 1d, have fewer message duplications while retaining the same search scope for this query.

However, we cannot draw the conclusion that the overlays in Figs. 1b, 1c, and 1d are better than the one in Fig. 1a because the above discussion only takes message and traffic cost into consideration. In fact, compared with Fig. 1a, the overlays in Figs. 1b, 1c, and 1d have fewer overlay connections, but may cause longer average query response times/query latencies. For example, when A issues a large number of queries, and D has most of the desired data, the query response time/query latency in the overlay in Fig. 1b will be much longer than that in the other three overlays.

Generally, as long as cycles exist in search paths, there will be message duplications in overlay connections. Some peers such as B and C are visited by the same query message multiple times. If a peer receives a query message with the same Message ID (GUID) as the one that it has received before, then the peer will discard the message. Since a peer is aware of this kind of revisit, we call it a Revisit Known (RK) problem. In the example shown in Fig. 1, although C eventually knows that the messages from A and B are duplicate ones, it cannot avoid such duplication on the link between B and C, unless we remove some logical links such as BC or AC, as shown in Figs. 1b, 1c, and 1d. On the other hand, reducing RK duplications might lead to an increase in the query latency. Hence, the first design objective of SBO is to reduce message duplications and attack RK problems with minimal increment in the query response time while retaining the same search scope of queries.

3.2 Message Duplications in Physical Links and Topology Mismatch Problem

We have discussed message duplications in overlay connections. However, for an overlay without RK problems, the same message still can traverse the same *physical* link multiple times, causing a large amount of unnecessary traffic and increasing the query response time. Here is an example. Suppose Fig. 2a illustrates the underlying physical network of the overlay, as shown in Fig. 1b, where A, B, C, and D are peering nodes, and node Y is not a peering node. We can see that the query message along the overlay path $A \rightarrow B \rightarrow C \rightarrow D$ traverses physical link YB twice. Node Y is visited twice.

Since node Y is not a peering node, the message duplication (revisit to Y) cannot be avoided. We may reduce the duplication between link YB by creating a direct connection between A and C and disconnecting the logical link BC, resulting in an overlay, as shown in Fig. 1d, but new duplications indeed occur in other links between Y and A. Later discussions in this paper will show that SBO will carefully choose the most efficient overlay connections under this situation by comparing the delay of logical connections and observing the behavior of each peering node.

Fig. 2b illustrates another underlying topology of the overlay, as shown in Fig. 1b. For a query message along the overlay path $A \rightarrow B \rightarrow C \rightarrow D$, D is visited three times. Node D is a peering node, but in the first two visits, D is visited as a nonpeering node. These first two visits are not known by the P2P application. We define this kind of revisit as a *Revisit Not known* (RN) problem. In this case, three physical links have been traversed twice, as shown in Fig. 2b; thus, a topology mismatch problem occurs.

It is more effective to solve RN problems than RK problems, since RN problems not only increase message duplications/traffic cost as RK problems do, but also significantly increase the query response time. In Fig. 2b, node D has been visited by the same query message twice before it "formally" receives the query as a peering node. If we can replace the overlay in Fig. 1b with the one in Fig. 1c for the physical topology shown in Fig. 2b, then there will be no message duplication at all, and the response time from D to A will decrease significantly or substantially. Hence, the second objective of SBO is to improve search performance by alleviating RN problems.

Indeed, the stochastic peer connection and peers randomly joining and leaving a P2P network cause large amounts of topology mismatch between the P2P overlay network and the physical underlying network. Studies in [22] have shown that only 2 percent to 5 percent of Gnutella connections link peers within a single autonomous system (AS). However, 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, increasing the topology mismatch costs. Our simulation results in Section 5 show that 744,734 out of 1,000,000 query responses traverse along mismatched paths, in each of which at least one of the peering nodes is visited as a nonpeering node more than once.

4 SBO

Optimizing inefficient overlay topologies can fundamentally improve P2P search efficiency. All the existing approaches such as forwarding-based and cache-based improvement strategies could be employed based upon an efficient overlay. In this paper, we propose a design of SBO that effectively avoids RK and RN problems to improve search efficiency in unstructured P2P networks.

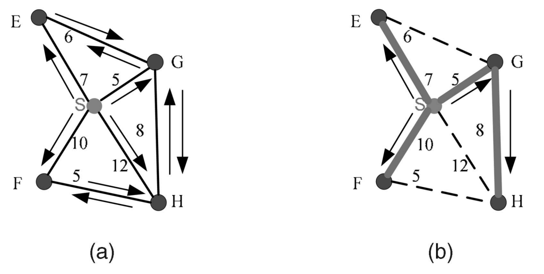


Fig. 3. ACE.

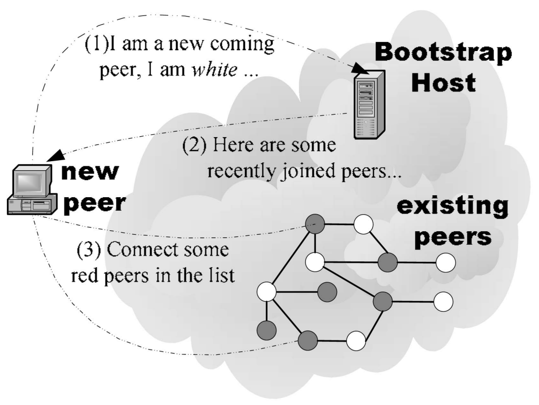

(1)I am a new coming peer, I am *white* ...
Bootstrap Host
(2) Here are some recently joined peers...
new peer
existing peers
(3) Connect some red peers in the list

Fig. 4. Bootstrapping a new peer.

4.1 Our First Attempt

Before going to SBO, we briefly introduce our early attempt, ACE, so that we can clearly see why SBO outperforms ACE in the traffic cost and response time reduction.

ACE uses the network delay between two peering nodes as a metric for measuring the cost between peers. 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 neighbors is known to the source peer. If we use $\mathrm{N}(S)$ to denote the set of direct logical neighbors of peer S, then each peer S has the information to obtain the overlay topology including S itself and $\mathrm{N}(S)$, as illustrated in Fig. 3a.

Based on the obtained neighbor cost tables, an MST among each peer S and its immediate logical neighbors $(S \cup \mathrm{N}(S))$ is built, as shown in Fig. 3b. Then, the message routing strategy of a peer is to select the peers that are the direct neighbors in the MST to send its queries. We can see that, after ACE operations, the traffic cost is reduced from $7 + 5 + 10 + 12 + 6 \times 2 + 8 \times 2 + 5 \times 2 = 72$ to $7 + 5 + 10 + 8 = 30$. After recognizing the so-called nonflooding neighbors such as peer H, as shown in Fig. 3, ACE selects closer peers as its new neighbors [29].

It is also shown that a larger diameter ACE leads to a better topology optimization rate. However, working within a larger diameter often means more information exchange and a higher overhead. Can we enable the peers to compute MSTs more than one hop away without an additional overhead? This is one direct motivation for the design of SBO.

4.2 Design of SBO

SBO employs an efficient strategy to select query-forwarding paths and logical neighbors. In the SBO design, we divide all the nodes into two groups: white ones and red ones. The advantage of SBO is twofold: 1) instead of letting every node probe the neighbor distances and compute the MST, SBO assigns one group of nodes to *probe* only and lets the other group *compute* only, reducing the average overhead incurred by each node, and 2) due to the bipartite property, SBO nodes are able to compute MST in a two-hop depth without an extra overhead, thus increasing the convergent speed of the algorithm.

In the first phase of SBO, each joining peer is randomly assigned a color: white or red. Each peer is only connected with peers in a different color. In the second phase, each white peer probes its distances with all its red neighbors and reports the information to the red neighbors. In the third phase, each red peer computes efficient forwarding paths so that the same search scope can be retained without the need to flood a query to all neighbors. In the fourth phase, a white peer that is not on the forwarding path tries to find a more efficient red peer to replace its current neighbor. Thus, the topology construction and optimization of SBO consist of four phases: bootstrapping a new peer, neighbor distance probing and reporting, forwarding connections (FCs) computing, and direct neighbor replacement. Details are explained as follows:

Phase 1: Bootstrapping a new peer. A typical unstructured P2P system provides several permanent well-known bootstrap hosts to maintain a list of recently joined peers so that a new incoming peer can find an initial host to start its first connection by contacting the bootstrap hosts. In the design of SBO, when a new peer is joining the P2P system, it will randomly take an initial color: red or white. A peer should keep its color until it leaves and again randomly select a color when it rejoins the system. Thus, each peer has a color associated with it, and all peers are separated into two groups: red and white. In SBO, a bootstrap host will provide the joining peer a list of active peers with color information. The joining peer then tries to construct connections to the different color peers in the list. Fig. 4 illustrates a new peer's joining process. In this way, all the peers form a bipartite overlay, in which a red peer will only have white peers as its direct neighbors, and vice versa.

Once a peer has joined the P2P system, it will periodically ping the network connections and obtain the IP addresses of other peers in the network, which will be used to create new connections for the peer's rejoining or in case the peer loses some of its connections with its neighbors due to the neighbors' departure or failure, or faults in the underlying networks.

Phase 2: Neighbor distance probing and reporting by white peers. We use the network delay between two peers as a metric for measuring the traffic cost between peers. We modify the Limewire implementation of the Gnutella V0.6 P2P protocol [7] by adding one routing message type for a peer to measure the transmission cost with its neighbors. Each white peer probes the costs with its immediate logical neighbors, forms a neighbor cost table, and sends this table to all its neighbors that are all red peers. The impact of the

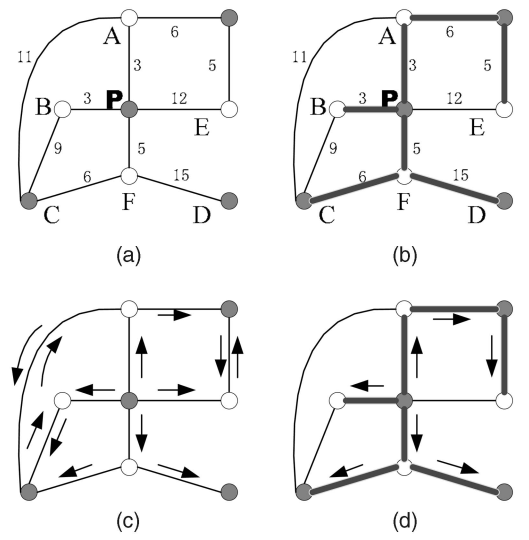


Fig. 5. A red peer P has a small overlay topology of $\mathrm{N}(P)$ and $\mathrm{N}^2(P)$ and computes the efficient forwarding paths.

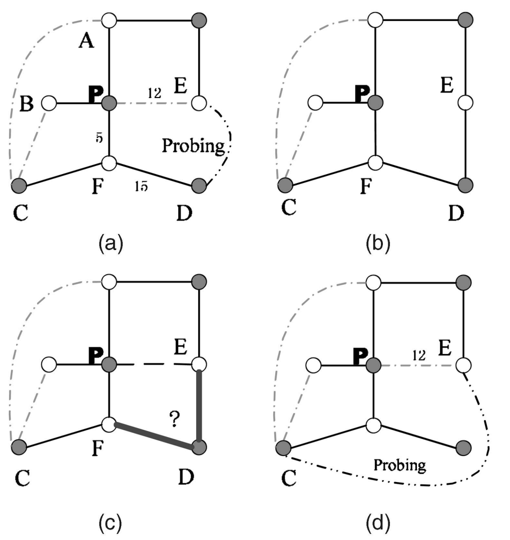


Fig. 6. An example of neighbor replacement.

frequency of the white peers' probing and cost table reporting operation will be discussed in more detail in Section 5.

We use $\mathrm{N}^2(S)$ to denote the set of peers being two hops away from S. Since each red peer P receives the cost table from its white neighbors about its all red neighbors, the red peer P has the information to obtain the overlay topology including P itself, $\mathrm{N}(P)$, and $\mathrm{N}^2(P)$, as illustrated in Fig. 5a. Note that in SBO, the overlay forms a bipartite topology, so there are no connections between any pairs of peers in $\mathrm{N}^2(P)$. Thus, we only require all the white peers to probe the costs to their neighbors and send out the cost tables. There is no need for the red peers to probe the distance.

Phase 3: FC computing by red peers. Based on cost tables obtained from neighbors, an MST can be built by each red peer such as P, as shown in Fig. 5b. Since a red peer builds an MST in a two-hop diameter, a white peer does not need to build an MST. The thick lines in the MST are selected as FCs, whereas the remaining lines are non-FCs (NFCs). Queries are only forwarded along FCs. For example, in Fig. 5b, P will send/forward queries to A, B, and F, but not E. Peer P also informs E that E is a nonforwarding neighbor. This information will be used by E in phase 4, that is, direct neighbor replacement.

Fig. 5c illustrates how the query message from P is flooded along the connections based on Fig. 5a. We can see many message duplications, that is, the RK problem. The total traffic cost incurred by the query is $3 + 6 + 5 + 5 + 12 + 3 + 5 + 6 + 9 + 9 + 15 + 11 + 11 = 100$. After FC computing, as shown in Fig. 5b, the traffic cost incurred by this query becomes $3 + 6 + 5 + 3 + 5 + 6 + 15 = 43$, as shown in Fig. 5d.

Although FC computing can reduce a lot of traffic while retaining the same search scope, as we described earlier, the price is to sacrifice the query response time or the query latency. For instance, P issues a query, and E has the desired data. The response time in Fig. 5c is $2 \times 12 = 24$. After FC computing, the response time becomes $2 \times (3 + 6 + 5) = 28$. Based on this observation, we will further improve our FC selecting algorithm later in this section.

Phase 4: Direct neighbor replacement by white peers. This operation is only conducted by white peers. The goal of neighbor replacement is to alleviate the topology mismatch problem or RN problems. As we have explained, solving the RN problem is essential, since it will not only reduce message duplications and the traffic cost, but also shorten the response times.

After computing an MST among the peers within two hops, a red peer P is able to send its queries to all the peers within two hops. Some white peers become nonforwarding neighbors, such as E in Fig. 5. In this case, for peer E, P is no longer its neighbor. In the phase of direct neighbor replacement, a nonforwarding neighbor E will try to find another red peer being two hops away from P to replace P as its new neighbor.

Peer P will send the neighbor cost tables that it collected from A, B, and F to the nonforwarding neighbor E so that E has enough information to find another neighbor to form a more efficient topology. Having received the cost tables, E can obtain the overlay topology among P and the peers $\mathrm{N}(P)$ and $\mathrm{N}^2(P)$. In the design of SBO, E will probe the round-trip times (RTTs) to all the red peers in $\mathrm{N}^2(P)$ and sort the red peers according to their RTTs. Peer E then selects the one with the smallest RTT, for example, peer D in Fig. 6a. There are three cases for peer E that finds D as its nearest red peer.

Case 1. The delay of ED is smaller than that of EP. The connection of ED will be created, and D becomes E's direct logical neighbor. The connection EP will be put into E's *will-cut list*, which is a list of connections to be cut later. A connection in a will-cut list will be disconnected when it has been in the list for a certain period of time. A peer will not send or forward any queries to the connections in its will-cut list.

The reason for E not disconnecting EP immediately is that some query responses might be sent back along the overlay path EP for some earlier queries. Disconnecting NFCs such as EP immediately may cause a serious response loss problem. Fig. 6b is the topology after E connects with D and disconnects with P after a time-out period.

Case 2. The delay of ED is larger than that of EP but is smaller than the larger delays of PF and FD. For example, if $\text{ED} = 13$, as shown in Fig. 6c, then $12 < \text{ED} < 15$. In this case, E will create the connection of ED and treat D as its direct neighbor. Peer E will not put connection EP into its will-cut list until it sends its neighbor cost table to D so that D still thinks that the connection of EP exists. Note that the algorithm is fully distributed. Thus, when red peer D conducts the FC computing, F will become D's nonforwarding neighbor. The white peer F will conduct the same operations as what peer E has done and may try to find a better red peer to replace node D as its neighbor.

Case 3. If ED has the largest delay among EP, PF, and FD, then peer E will pick the second nearest peer in $\text{N}^2(P)$ such as C, as shown in Fig. 6d, and repeat the above process until it finds a better node to replace P as its neighbor or until it has tried all the peers in $\text{N}^2(P)$.

The first three operations are relatively straightforward, so we do not provide the detailed pseudocode, and the pseudocode of the *direct neighbor replacement* operation is given as follows:

```
For a white peer i
For each peer $j$ in white peer $i$'s nonforwarding neighbor list
    Replaced = false;
    List = all the two hop away red neighbors of $j$, $\text{N}^2(\text{j})$;
    Peer $i$ pings all the peers in the list;
    Add peers' RTT information to the list;
    While the list is not empty and Replaced = false
        remove the peer $h$ with the smallest RTT from the list;
        if $\text{RTT}_{ih} < \text{RTT}_{ij}$ {replace $j$ by $h$ in $i$'s neighbor list;
            Put $j$ into the will-cut list;
            Replaced = true;}
        else
            Common_list = all common neighbors of peer $j$ and $h$;
            While Common_list is not empty and Replace = false
                Randomly remove a peer $k$ from the Common_list;
                $\text{RTT}_\text{k} = \max\{\text{RTT}_\text{kj}, \text{RTT}_\text{kh}\}$;
                if $\text{RTT}_{ih} < \text{RTT}_\text{k}$
                    {add $h$ to $i$'s neighbor list;
                    remove $j$ from $i$'s neighbor list right after
                    $i$ finds out $jk$ or $kh$ is disconnected;
                    Replaced = true;};
            End While;
    End While;
End For;
```

Based on the above discussions, we can see the major advantages of using a bipartite overlay. First, the operation overhead is reduced. In ACE, every peer is required to probe the neighbor distance and compute the MST, whereas in SBO, white peers do the probing and red peers do the computing. Second, SBO extends ACE into two-hop neighbors, without additional information exchange. For a red peer such as peer P in Fig. 5, it is sure that the two-hop topology that it gets is complete, as the bipartite property guarantees that there are no connections between P and A, C, or D, or among A, C, and D, since they are all red peers. As a result, our experimental results show that SBO outperforms ACE significantly.

4.3 Further Improvements

Previous studies have shown that queries and queried data have significant locality [25], [30]. A small number of peers issue a large portion of the queries, and only 5 percent of the files account for 50 percent of all transfers. Peers' behaviors are different in query frequency and response frequency. We define a *query-heavy* peer as a peer that issues queries frequently, and a *response-heavy* peer as a peer that often responds to queries. We have discussed in the previous section that reducing RK duplication may lead to an increase in the query response time. To avoid disconnecting a path from/to a query- or response-heavy peer, we further improve SBO by keeping some *single-direction connections* (SDCs).

Query-heavy peer. If the number of queries that a peer has issued or forwarded is five times more than the average number of queries in the last minute (the number of times, that is, five, is selected based on our simulation results), then it is defined as a query-heavy peer. In our simulation, with an average number of neighbors being 6, the initial time to live $(\text{TTL}) = 7$, an average peer lifetime of 10 minutes, and a query frequency of 0.3 query issued per peer per minute, we measured that the average number of queries processed (issued and forwarded) by each peer is about 15 to 25 per second. Thus, a peer is identified as a query-heavy peer if it processed more than 75 queries per second.

Response-heavy peer. In the Gnutella protocol V0.6, QueryHit (response) messages are sent along the same path that carried the incoming query message. In our simulation, a peer delivers or forwards three responses per minute on the average. In SBO, a peer that processed more than 20 responses in the last minute is defined as a response-heavy peer (the number of responses, that is, 20, is selected based on our simulation).

SDCs. Every peer in SBO will monitor its own status. If a peer finds itself a query or response-heavy peer, then it will report its status to all its neighbors. Thus, when a red peer computes FCs to form the forwarding paths, a white neighbor that is not a forwarding peer may be a query or response-heavy peer. The connection between the red peer and the white peer will be set as an SDC. For example, if peer E in Fig. 5b is a response-heavy peer, instead of setting PE as an NFC, then it will set PE as an SDC $\text{P} \rightarrow \text{E}$, where P will send/forward query messages to E, whereas E will not send/forward any query messages to P. In this case, E will still do its neighbor replacement operation. The SDC $\text{P} \rightarrow \text{E}$ will be disabled when E is no longer a response-heavy peer. If E is a query-heavy peer, then connection PE will be set as an SDC $\text{E} \rightarrow \text{P}$, where E will send/forward query messages to P, whereas P will not send/forward any query messages to E.

The simulation results in Section 5 will show that the design of SDC further improves the system search performance.

4.4 Traffic Overhead of SBO Optimizations

The simplicity of blind flooding makes it very popular in practice. This mechanism relays a query message to all its logical neighbors except the incoming peer. For each query, each peer records the neighbors that relay the query to it. A peer will discard a query message that has been received before. Therefore, in the worst case, the same query message can be sent on each logical link at most twice. For an overlay network with n peers, we use c_n to denote the average number of neighbors and use c_e to denote the average number of physical links in each logical link. The total traffic caused by a query is less than or equal to $n\ c_n\ c_e$. In a typical P2P system, the value of n (more than millions) is much larger than c_n (less than tens) [26], so we can view both c_n and c_e as constant numbers. Thus, in the flooding-based search, the traffic incurred by one query from an arbitrary peer in a P2P network is $O(n)$. As observed in [27], each peer issues 0.3 query per minute on the average. Thus, the per-minute traffic incurred by a P2P network with n peers is $O(n^2)$.

One optimization step of SBO includes all white peers' neighbor distance probing/reporting and neighbor replacement. In the worst case, each white peer P needs to probe every peer in $N^2(P)$. It is reasonable to assume that the traffic overhead of peer A probing peer B is equal to a query message traversing the connection AB twice. If each peer conducts the SBO optimization operation k times per minute, then the total traffic overhead per minute is

$$k \times \left(\frac{n}{2} \times \left(2c_nc_e + c_nc_e + 2c_n^2c_e\right)\right) = \frac{kc_nc_e(3+2c_n)}{2}n.$$

Our simulation results will show that the optimal value for k is less than 1, so the per-minute traffic overhead incurred by SBO to the P2P network is $O(n)$. Compared with the query traffic savings, the traffic overhead incurred by the SBO optimization is relatively trivial.

4.5 Property of SBO Operations

The strength of the SBO optimization operation is that it reduces the search traffic cost and query response time without shrinking the search scope of queries. In most previous 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. These kinds of approaches, however, often destroy the connectivity of overlays and, thus, create many isolated islands in P2P systems. In this section, we will prove that the SBO operations will not increase the number of components of a graph.

Theorem 1. *Given a bipartite graph $G = (V, E)$, the SBO 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 SBO operations. Suppose C is disconnected into two parts, X and Y, after the SBO operations, as shown in Fig. 7.

Before the SBO optimization, there must be one or more edges between X and Y, since C is connected. Let M denote the set of the edges between X and Y. 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. Since G is a bipartite graph, peers u and v must have different colors. Without loss of generality, we can assume that u is red and v is white. C being disconnected after the SBO operations means that none of the edges in M, including uv, is selected as u's forwarding path. We know that the red peer u employs an MST algorithm such as the Kruskal algorithm in the FC computing 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. If the edge uv is not selected by the MST, then it 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 X and Y, at least one of the edges in P, say, edge e, belongs to M. We then have $e < uv$, which is a contradiction to our choice that uv is the shortest edge in M and, thus, the theorem. □

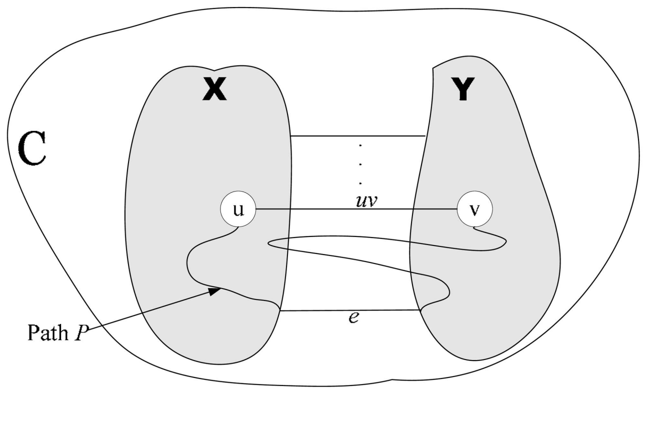


Fig. 7. Proof of the property of SBO operations.

5 Performance Evaluation

To evaluate the effectiveness of SBO, we conducted comprehensive simulations. Based on the real P2P overlay topology and the generated networks, we simulate P2P flooding search, host joining/leaving behavior, SBO operations, and other previous approaches.

5.1 Simulation Methodology

We use two types of topologies, physical topology and logical topology, in our simulation. The physical topology should represent the real topology with Internet characteristics. The logical topology represents the overlay topology built on top of the physical topology. All P2P nodes are in a subset of nodes in the physical topology. Previous studies have shown that both large-scale Internet physical topologies [28] and P2P overlay topologies [24] follow the small world and power law properties. Power law describes the node degree, whereas the small world describes characteristics of path length and clustering coefficients [9]. The study in [24] found that the topologies generated using the AS model have the properties of the small world and power law. The Boston University Representative Internet Topology gEnerator (BRITE) [4] 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. We also generate logical topologies, with the number of peers ranging from 3,000 to 8,000. The average number of neighbors of each node ranges from four to 10. We simulate SBO 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 a decision support system (DSS) clip-2 trace). We obtained consistent results on the real-world topology and the generated topologies.

In this evaluation, we simulate the flooding search used in the Gnutella network by conducting the Breadth-First

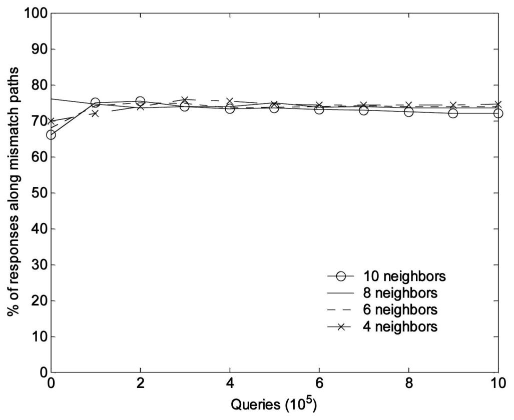


Fig. 8. The percent of query responses along mismatching paths.

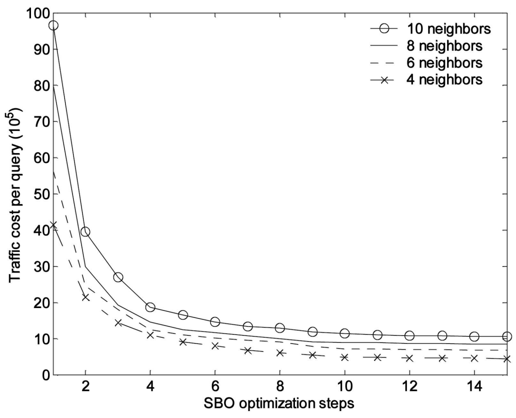


Fig. 9. Traffic reduction versus optimization steps.

Search (BFS) algorithm from a specific node. A search operation is simulated by randomly choosing a peer as the sender and a keyword according to Zipf distribution. In our simulation, 1,000,000 search operations are simulated. A well-designed search mechanism should seek to optimize both efficiency and quality of service (QoS). Hence, we mainly focus on two performance metrics: the average traffic cost and the query response time. Traffic cost is one of the parameters that most seriously concern network administrators. Heavy network traffic limits the scalability of P2P networks [23] and is also a reason why a network administrator may prohibit P2P applications. We define the traffic cost as the network resource used in an information search process of P2P systems, which is mainly a function of consumed network bandwidth and other related expenses. The response time of a query is one of the parameters concerned by P2P users. 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.

5.2 The Amount of Mismatch

We first quantitatively evaluate how serious the topology mismatch problem is in Gnutella-like networks. We insert 1,000,000 queries into different topologies and track the response of each query message to see if the response path is a mismatching path. We count a path as a mismatching path if a peering node in the path has been visited more than once, that is, the RN problem. The results in Fig. 8 show that about 75 percent of the paths have the topology mismatch problem. Although the mismatch problem is slightly less serious when the average number of neighbors increases, we can see that the mismatching degree is not very sensitive to the average number of neighbors when the number is changed from four to 10. We expect that the mismatching degree can be greatly reduced if the average number of neighbors increases significantly. However, an extremely large average number of neighbors is not realistic in existing P2P networks.

5.3 Effectiveness of SBO in Static Environments

We study the effectiveness of SBO in a static P2P environment where the peers do not join and leave frequently. This will show, without changing the overlay topology, how many SBO optimization steps are required to reach a better topology matching. Here, one step means that each red peer collects the neighbor cost tables from its neighboring white peers and computes the efficient FCs, and its neighbors finish neighbor replacement operations if needed. Note that in the design of SBO, if the reported information from all neighbors, including neighbor status and cost tables, are not changed, then the red peer will not compute FCs. Consequently, the neighboring white peers will not do the neighbor replacement operations.

One goal of SBO is to reduce the traffic cost as much as possible while retaining the same search scope. We generate 500,000 queries and simulate flooding search for different topologies, with the average neighbor number as four, six, eight, and 10 after each SBO optimization step. Fig. 9 shows that the traffic cost decreases when the optimization operations of SBO are conducted multiple times, where the search scope is all 7,000 peers. To cover the same search scope, SBO reduces the traffic cost significantly in the first two optimization steps. We can see that the traffic cost reduction reaches a threshold after eight to 10 steps of the SBO optimization.

Short query response time is always preferred in P2P systems. The simulation results in Fig. 10 show that SBO can effectively shorten the query response time by about 60 percent in the first 10 optimization steps. The trade-off between the query traffic cost and response time has been investigated in the recent years. P2P systems with a large number of average connections offer a faster search speed while increasing traffic. One of the strengths of SBO is that it reduces both the query traffic cost and response time without decreasing the query success rate. Our other simulation results also show that different densities of

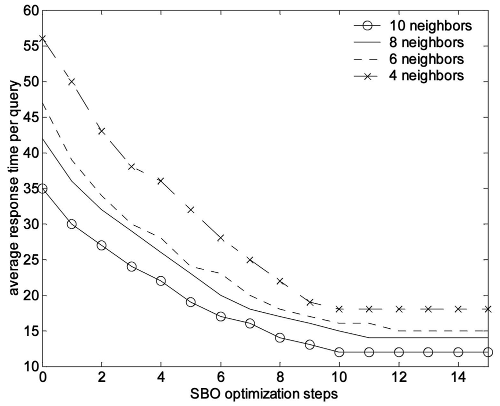


Fig. 10. Response time versus optimization steps.

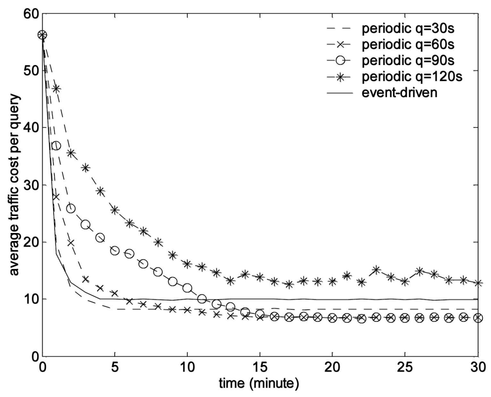


Fig. 11. Traffic cost reduction in a dynamic P2P environment.

logical peers or physical nodes will not degrade the effectiveness of SBO. The average traffic cost is only proportional to the average number of neighbors and the average cost of logical links, which is consistent with previous analysis.

5.4 Effectiveness and Frequency of SBO Optimizations

We further evaluate the effectiveness of SBO in dynamic P2P systems and explore the best frequency for each peer to conduct the SBO optimization operations. There are two ways for a white peer to decide when to conduct neighbor probing and reporting, namely, periodic and event driven. In the periodic approach, each white peer conducts neighbor distance probing at a predetermined interval q. After probing the distances to all the neighbors, a white peer sends the cost table to its neighboring red peers. In the event-driven approach, a white peer produces and sends an updated cost table to its neighboring red 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 the periodic approach. We have investigated the impact of different values of q ranging from 20 to 600 seconds. Figs. 11 and 12 show the results on some representative samples of q at 30, 60, 90, and 120 seconds, 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, for example, $q = 30$, then peers will conduct the optimization operations too often, resulting

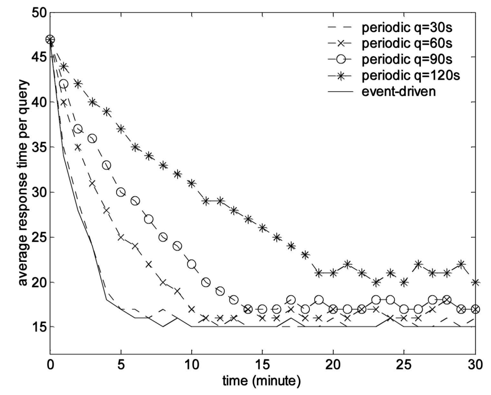


Fig. 12. Response time reduction in a dynamic P2P environment.

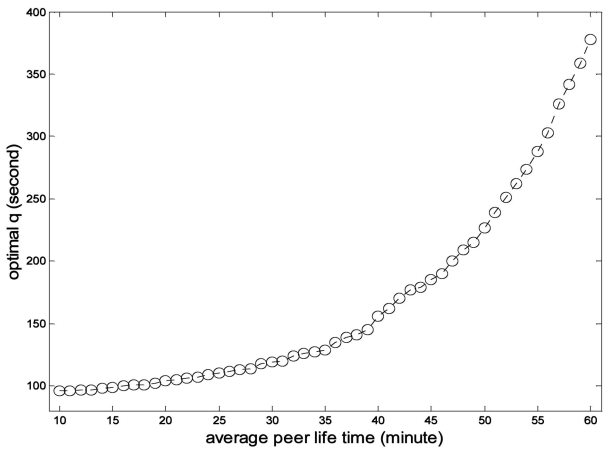


Fig. 13. Optimal SBO operation interval.

in the overhead still growing although the reduction of the traffic cost and response time have already reached a threshold. On the other hand, if q is too large, for example, $q = 120$, then the frequency of the optimization operations are not frequent enough to catch the changes from the random joining and leaving of peers. Thus, the convergent speed is slow, and the reduction of the traffic cost and the response time are limited. We define the value of q to be optimal if it incurs the smallest traffic cost, which is the sum of the query flooding traffic and the optimization operation overhead traffic, and the difference between its average response time and the response time threshold is not more than 5 percent. Figs. 11 and 12 suggest that $q = 90$ seconds is the best, with about 85 percent of reduction on the traffic cost and 60 percent of reduction on the response time. Our results for different setups show that the optimal q ranges from 60 to 90 seconds, as long as the average peer lifetime remains at 10 minutes. The value of q should be able to adapt to the average peer lifetime in order to achieve optimal performance. Figs. 11 and 12 also show that the *periodic* approach, with $q = 90$ sec, outperforms the *event-driven* approach on traffic reduction.

Different values of the average peer lifetime have been presented by previous studies [8], [24]. We further tune the average peer lifetime in our simulation. Fig. 13 shows that the SBO operations can be conducted less frequently if the average peer lifetime is longer. From the simulation results, we also see that, if the average peer lifetime is longer than 37 minutes, then the event-driven policy outperforms the periodic policy. In a hierarchical P2P system such as KaZaA, the flooding-based search is only employed among superpeers. The mechanism to select superpeers makes the superpeers more stable than leaf peers. Thus, an *event-driven* policy is highly recommended when SBO is implemented among superpeers.

5.5 SBO with SDC and Index Caching

We have discussed the design of SDC in Section 4.3 to further improve SBO. In this section, we evaluate SDC on top of SBO and a strategy of combining SBO with the response index caching scheme. We plot the traffic cost and response time in a Gnutella-like system without any optimization, with query index caching only, with the SBO optimization only, with SDC-enabled SBO, and with SDC-enabled SBO plus response index caching, as shown in

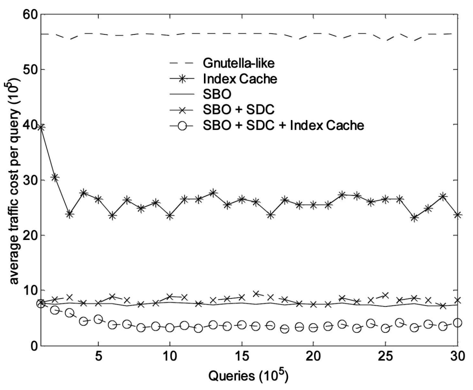


Fig. 14. Average traffic cost of five schemes.

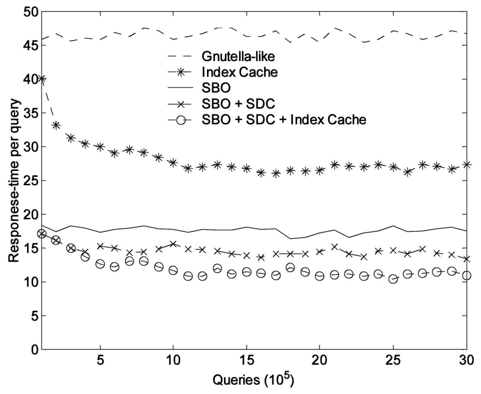


Fig. 15. Average response time of five schemes.

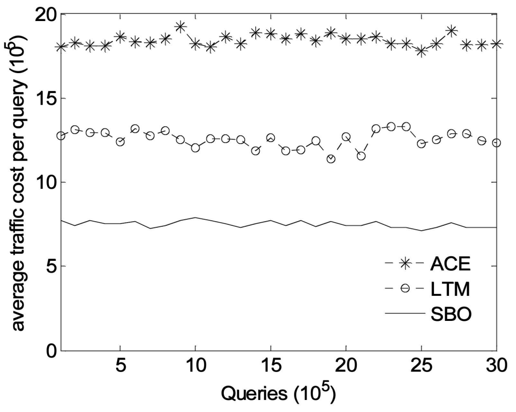


Fig. 16. Comparison on traffic cost.

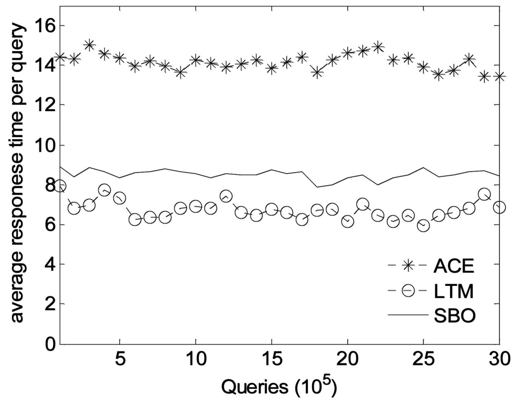


Fig. 17. Comparison on response time.

Figs. 14 and 15. The design of SDC can further improve the average response time of SBO by about 25 percent, with a very trivial traffic cost increment. Also, compared with SBO, by combining SDC-enabled SBO with response index caching, the traffic cost is reduced by about 50 percent without shrinking the search scope, and the average query response time is reduced by about 42 percent.

5.6 Comparison with Previous Approaches

We compare the performance of the three approaches, the previous ACE [29], LTM [19], and the proposed SBO in dynamic P2P environments, in Figs. 16 and 17. In this simulation, the size of the overlay topology is 5,000, and the physical topology has 27,000 nodes.

In the figures, we can see that the convergent speed of ACE is the slowest, so its overall performance in dynamic environments is not as good as SBO and LTM. SBO, incurring only half of the overhead of ACE, reduces the traffic cost the most. Although the convergent speed of LTM is fast and it reduces the response time a little bit more than SBO, it needs the support of NTP [5] to synchronize the peering nodes, which means a lot of additional overhead. Overall, SBO outperforms ACE and LTM.

6 Conclusion and Future Work

We propose an SBO to attack the topology mismatch problem in P2P systems. SBO is scalable and completely distributed in the sense that it does not require any global knowledge when each node is optimizing its logical neighborhood. We show that the significant performance benefit of SBO is consistent with various network sizes and average numbers of neighbors. SBO achieves about 85 percent of reduction on the traffic cost and about 60 percent of reduction in the query response time. Our experimental results also demonstrate that SBO significantly outperforms existing approaches to address the overlay mismatch problem.

Indeed, there are two fundamental issues in P2P systems: efficiency and security. Over the past years, many efforts have been made to improve the efficiency of P2P systems, and SBO is one of them. Current P2P systems, however, are facing more challenges from security problems. Our future work will consider how we can provide a healthier P2P environment. We will pay more attention on P2P reliability [16], privacy [12], incentive [15], and trustworthiness [13].

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. This work was supported in part by Hong Kong RGC Grants HKUST6152/06E and HKUST6264/04E, the China National Natural Science Foundation (NSFC) Grants 60573129 and 60673166, the US National Science Foundation (NSF) Grants CCF 0514078, CNS 0549006, and CNS 0551464, and Microsoft Research Asia. Some preliminary results of this work were presented in the Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004).

References

[1] Gnutella Network Size, http://www.limewire.com/index.jsp/size, 2007.

[2] FastTrack, http://developer.berlios.de/projects/gift-fasttrack/, 2007.

[3] KaZaA, http://www.kazaa.com, 2007.

[4] BRITE, http://www.cs.bu.edu/brite/, 2007.

[5] NTP: The Network Time Protocol, http://www.ntp.org/, 2007.

[6] BitTorrent, http://www.bittorrent.com/, 2007.

[7] The Gnutella Protocol Specification 0.6, http://rfc-gnutella.sourceforge.net, 2007.

[8] R. Bhagwan, S. Savage, and G.M. Voelker, "Understanding Availability," *Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '03),* 2003.

[9] T. Bu and D. Towsley, "On Distinguishing between Internet Power Law Topology Generators," *Proc. IEEE INFOCOM,* 2002.

[10] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM,* 2003.

[11] Y. Chu, S.G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM SIGMETRICS,* 2000.

[12] J. Han and Y. Liu, "Rumor Riding: Anonymizing Unstructured Peer-to-Peer Systems," *Proc. 14th IEEE Int'l Conf. Network Protocols (ICNP '06),* 2006.

[13] J. Han and Y. Liu, "Dubious Feedback: Fair or Not," *Proc. Int'l Workshop Peer-to-Peer Information Management (P2PIM '06),* 2006.

[14] B. Krishnamurthy and J. Wang, "Topology Modeling via Cluster Graphs," *Proc. ACM SIGCOMM Internet Measurement Workshop '01,* 2001.

[15] X. Liao, H. Jin, Y. Liu, L.M. Ni, and D. Deng, "AnySee: Peer-to-Peer Live Streaming," *Proc. IEEE INFOCOM,* 2006.

[16] X. Liu, L. Xiao, A. Kreling, and Y. Liu, "Optimizing Overlay Topology by Reducing Cut Vertices," *Proc. 16th ACM Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '06),* 2006.

[17] Y. Liu, X. Liu, L. Xiao, L.M. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems," *Proc. IEEE INFOCOM,* 2004.

[18] 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 '04),* 2004.

[19] 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,* vol. 16, pp. 163-174, 2005.

[20] V.N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," *Proc. ACM SIGCOMM,* 2001.

[21] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," *Proc. ACM SIGCOMM,* 2004.

[22] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing,* vol. 6, 2002.

[23] J. Ritter, "Why Gnutella Can't Scale. No, Really," http://www.tch.org/gnutella.html, 2001.

[24] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. ACM/SPIE Conf. Multimedia Computing and Networking (MMCN '02),* 2002.

[25] M.T. Schlosser and S.D. Kamvar, "Availability and Locality Measurements of Peer-to-Peer File Systems," *Proc. SPIE ITCom Conf. Scalability and Traffic Control in IP Networks,* 2002.

[26] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. ACM SIGCOMM Internet Measurement Workshop '02,* 2002.

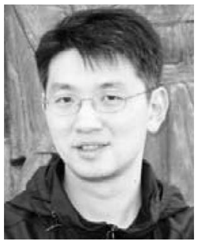
[27] K. Sripanidkulchai, "The Popularity of Gnutella Queries and Its Implications on Scalability," http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html, 2007.

[28] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based versus Structural," *Proc. ACM SIGCOMM,* 2002.

[29] L. Xiao, Y. Liu, and L.M. Ni, "Improving Unstructured Peer-to-Peer Systems by Adaptive Connection Establishment," *IEEE Trans. Computers,* 2005.

[30] Y. Xie and D. O'Hallaron, "Locality in Search Engine Queries and Its Implications for Caching," *Proc. IEEE INFOCOM,* 2002.

[31] Z. Xu, C. Tang, and Z. Zhang, "Building Topology-Aware Overlays Using Global Soft-State," *Proc. 23rd Int'l Conf. Distributed Computing Systems (ICDCS '03),* 2003.



Yunhao Liu received the BS degree from the Department of Automation, Tsinghua University in 1995, the MA degree from Beijing Foreign Studies University in 1997, and the MS and PhD degrees in computer science and engineering from Michigan State University in 2003 and 2004, respectively. He is now an assistant professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST). His research interests include peer-to-peer and grid computing, sensor networks, pervasive computing, network security, and high-speed networking. He is a senior member of the IEEE and the IEEE Computer Society.

Li Xiao received the BS and MS degrees in computer science from Northwestern Polytechnic University, China, and the PhD degree in computer science from the College of William and Mary in 2002. She is an assistant professor of computer science and engineering at Michigan State University. Her research interests are in the areas of distributed and Internet systems, overlay systems and applications, sensor networks, 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.

Lionel M. Ni received the PhD degree in electrical and computer engineering from Purdue University, West Lafayette, Indiana, in 1980. He is a chair professor at the Hong Kong University of Science and Technology (HKUST) and heads the Department of Computer Science and Engineering, HKUST. He is also the director of the HKUST China Ministry of Education/Microsoft Research Asia IT Key Laboratory. He has chaired many professional conferences and has received a number of awards for authoring outstanding papers. He is a coauthor of the book *Interconnection Networks: An Engineering Approach* (with Jose Duato and Sudhakar Yalamanchili, Morgan Kaufmann, 2002) and the book *Smart Phone and Next Generation Mobile Computing* (with Pei Zheng, Morgan Kaufmann, 2006). He is a fellow of the IEEE and a member of the IEEE Computer Society.