A Distributed Approach to Solving Overlay Mismatching Problem

Yunhao Liu[1], Zhenyun Zhuang[1], Li Xiao[1] and Lionel M. Ni[2]

[1]*Department of Computer Science & Engineering*
Michigan State University
East Lansing, Michigan, USA
{liuyunha, zhuangz1,lxiao}@cse.msu.edu

[2]*Department of Computer Science*
Hong Kong University of Science and Technology
Kowloon, Hong Kong, China
ni@cs.ust.hk

Abstract

In unstructured peer-to-peer (P2P) systems, the mechanism of a peer randomly joining and leaving a P2P network causes topology mismatching between the P2P logical overlay network and the physical underlying network, causing a large volume of redundant traffic in the Internet. In order to alleviate the mismatching problem, we propose Adaptive Connection Establishment (ACE), an algorithm of 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 mismatching problem and significantly reduce P2P traffic. We further study the tradeoffs between the topology optimization rate and the information exchange overhead by changing the diameter used to build the tree.

1. Introduction

In unstructured P2P systems, queries are flooded among peers (such as in Gnutella [2]) or among supernodes (such as in KaZaA [1]). 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 [15] 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 ASes. This means that most Gnutella-generated traffic crosses AS borders so as to increase topology mismatching costs. The same message can traverse the same physical link multiple times, causing large amount of unnecessary traffic.

The objective of this paper is to minimize the effect due to topology mismatching. 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 significant 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 tradeoffs 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 III presents the adaptive connection establishment (ACE) scheme. Section IV describes our simulation methodology. Performance evaluation of the ACE is presented in Section V, and we conclude the work in Section VI.

2. Related Work

In order to reduce unnecessary flooding traffic and improve search performance, two approaches have typically been used 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 informa-

tion [10, 22, 23]. In the second approach, a peer keeps indices of other peers' sharing information or caches query responses in hoping that subsequent queries can be satisfied quickly by the cached indices or responses [6, 11, 12, 14, 18, 20, 22]. The performance gains of both approaches are seriously limited by the topology mismatching 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 [5], which first constructs a rich connected graph on which to further construct shortest path spanning trees. Each tree rooted at the corresponding source using well-known routing algorithms. This approach introduces large overhead 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 infeasible to large-scale P2P systems.

Researchers have also considered to cluster close peers based on their IP addresses (e.g., [7, 13]). We believe there are two limitations for this approach. First, the mapping accuracy is not guaranteed by this approach. Second, this approach may affect the 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.

Recently, researchers in [21] have proposed to measure the latency between each peer to multiple stable Internet servers called ``landmarks''. The measured latency is used to determine the distance between peers. This measurement is conducted in a global P2P domain and needs the support of additional landmarks. Similarly, this approach also affects the search scope in P2P systems. In contrast, our measurement is conducted in many small regions, significantly reducing the network traffic.

Gia [4] 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 mismatching problem between the overlay and physical networks.

A preliminary design of ACE, which is called AOTO, has been discussed in [8]. We have also proposed a location-aware topology matching scheme [9], 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" a query to the network among peers or among supernodes. 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 use examples to explain the unnecessary traffic incurred by flooding based search and the topology mismatching problem. Then we introduce the design of proposed approach, ACE.

3.1. Unnecessary Traffic by Flooding

Figure 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 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 transmission on the logical link LM is unnecessary. In such a simple overlay, node M will receive the same query message for 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.
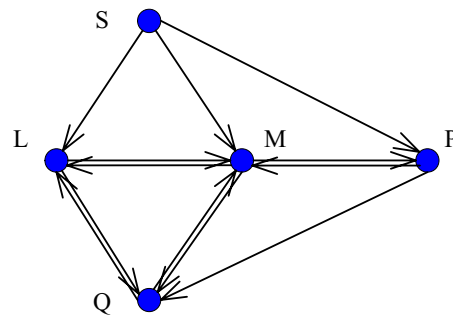

Figure 1: An example of P2P overlay

3.2. Topology Mismatching

As we have discussed the stochastic peer connection and peers randomly joining and leaving a P2P network can cause topology mismatching between the P2P logical overlay network and the physical underlying network. For example, Figures 2(a) and 2(b) are two overlay topologies on top of the underlying physical topology shown in Figure 2(c). Suppose nodes S and B are in the same autonomous system (AS) at Michigan State University (MSU) in USA, while nodes A and C are in another AS at Tsinghua Uni-

versity in China. So we can assume that the physical link delay between nodes S and C is much longer than the link between nodes S and B, or nodes A and C in Figure 2(c). Clearly, in the inefficient mismatching overlay of Figure 2(a), the query message from source S will traverse the longest link SC three times, which is a scenario of topology mismatching. If we can construct an efficient overlay shown in Figure 2(b), the message needs to traverse all the physical links in Figure 2(c) only once.
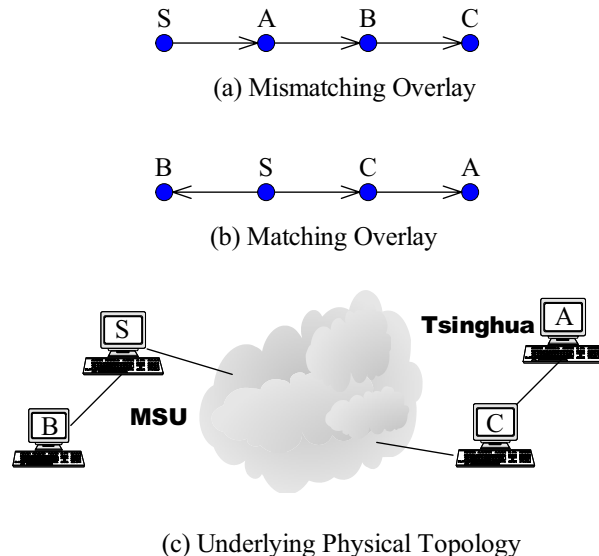

(a) Mismatching Overlay

(b) Matching Overlay

(c) Underlying Physical Topology

Figure 2: Topology mismatching problem

3.3. Design of ACE

Optimizing inefficient overlay topologies can fundamentally improve P2P search efficiency. The proposed approach, Adaptive Connection Establishment (ACE), includes three phases.

Phase 1: we use network delay between two nodes as a metric for measuring the cost between nodes. 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 neighbors is known to the source peer.

Phase 2: based on obtained neighbor cost tables, a minimum spanning tree among each peer and its immediate logical neighbors then can be built by simply using an algorithm like PRIM which has a computation complexity of $O(m^2)$, where m is the number of logical neighbors of the source node. Now the message routing strategy of a peer is to select the peers that are the direct neighbors in the multicast tree to send its queries, instead of flooding queries to all neighbors. An example is shown in Figure 3. In Figure

3(a), 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 phase 2, we can see the forwarding connections are changed as shown in Figure 3(b), and the total traffic cost becomes: 6+4+14=24.

In Figure 3(b), node S sends a message only to nodes E and F and expects that node E will forward the message to node G. Note that in this phase, even node S does not flood its query message to node G any more. S still retains the connections with G and keeps exchanging the neighbor cost tables. We call node G *non-flooding neighbor of node S*, which is the direct neighbor potentially to be replaced in the next phase.
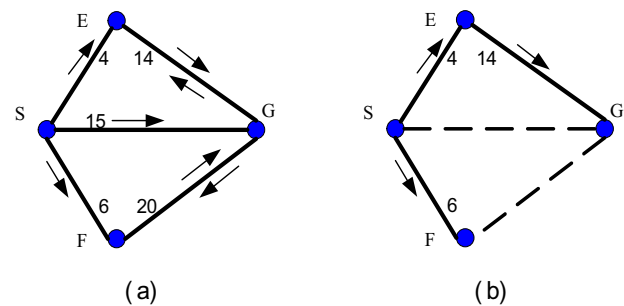

(a) (b)

Figure 3: Second phase in ACE

Phase 3: this phase reorganizes the overlay topology. Note that each peer has a neighbor list which is further divided into flooding neighbors and non-flooding neighbors in Phase 2. Each peer also has the neighbor cost tables of all its neighbors. In this phase, it tries to replace those physically far away neighbors by physically close by neighbors, thus minimizing the topology mismatching 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 non-flooding neighbor may be replaced by one of the non-flooding neighbor's neighbor.

The basic concept of phase 3 is illustrated in Figure 4. In Figure 4(a), node S is probing the distance to one of its non-flooding neighbor G's neighbors, for example, H. If SH is smaller than SG, as shown in Figure 4(b), 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 nodes S and H, as shown in Figure 4(c), S will keep H as a new neighbor. Since the algorithm is executed in each peer independently, S cannot let G to remove H from its neighbor list. However, as long as S keeps both G and H as its logical neighbors, we may expect that node H will become a non-flooding neighbor to node G after node G's Phase 2 since node 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 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 since the spanning tree has been built for S. Obviously if SH is larger than SG and GH, as shown in Figure 4 (d), this connection will not be built and S will keep probing other G's director neighbors.
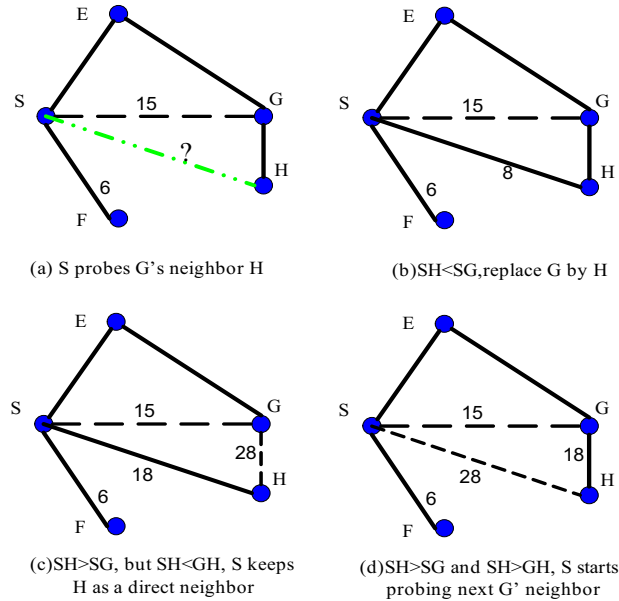

(a) S probes G's neighbor H

(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 next G' neighbor

Figure 4: Third phase in ACE

3.4. Depth of Optimization

We define *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. In the initial ACE described in Section 3.3, the optimization is only conducted within 1-neighbor closure (among each

source peer and all its direct logical neighbors). We can enlarge the optimization scope by increasing the value of *h*. A larger value of *h* leads to a better topology matching improvement, but a higher overhead due to the extra information exchanging. We will further study in this direction with the aim of reaching a good performance level by considering the tradeoffs between the topology optimization improvement and the information exchange overhead.

Figure 5 illustrates the overlay trees constructed for each peer within 1-neighbor closures. 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 *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 who 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 3 to 1 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 same query message traverses 3 paths twice, which is *B–D, D–E,* and *C–E*.

Figure 6 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.
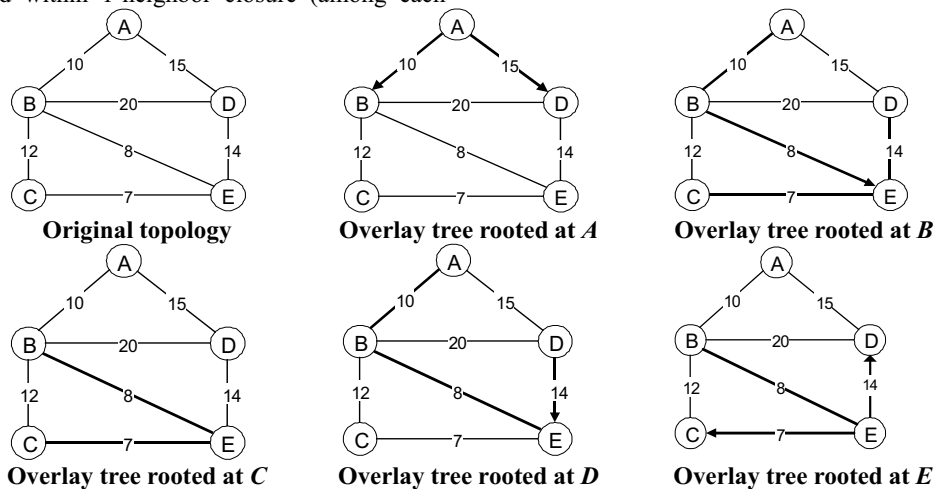

Original topology

Overlay tree rooted at A

Overlay tree rooted at B

Overlay tree rooted at C

Overlay tree rooted at D

Overlay tree rooted at E

Figure 5: Overlay trees built in 1-neighbor closure

Table 1. Query paths and costs on overlay trees built in 1-neighbor closure

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

Table 2. Query paths and costs on the overlay tree built in 2-neighbor closure

Query Path		Corresponding Cost
From	To	
A	B	10
B	E	8
E	C, D	7+14=21
Total Cost		39
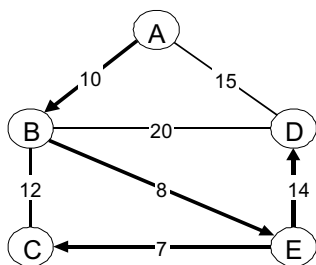
Figure 6: Overlay tree built in 2-neighbor closure

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. Topology Generation

Both physical topologies and logical overlay topologies which can accurately reflect the topological properties of real networks in each layer are needed in the simulation study. Previous studies have shown that both large scale Internet physical topologies [6] and P2P overlay topologies [7] follow small world and power law properties. Power law describes the node degree while small world describes characteristics of path length and clustering coefficient [9]. The study in [6] found that the topologies generated using the AS Model have the properties of small world and power law. BRITE is a topology generation tool that provides the option to generate topologies based on the AS Model. We generate 10 physical topologies each with 20,000 nodes. The logical topologies are generated with the number of peers (nodes) ranging from 1000 to 8000. For each given number of nodes, we generate logical topologies with average edge connections between 1 and 20. Note that an edge

connection of value m indicates that there are $2m$ logical neighbors for each peer.

4.2. Performance Metrics

A well-designed search mechanism should seek to optimize both efficiency and Quality of Service (QoS). Efficiency focuses on better utilizing resources, such as bandwidth and processing power, while QoS focuses on user-perceived qualities, such as number of returned results and response time. In unstructured P2P systems, the QoS of a search mechanism generally depends on the number of peers being explored (queried), response time, and traffic overhead. If more peers can be queried by a certain query, it is more likely that the requested object can be found. So we use several performance metrics as follows.

Traffic cost is one of the parameters seriously concerned by network administrators. Heavy network traffic limits the scalability of P2P networks [16] 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 a function of consumed network bandwidth and other related expenses.

S*earch 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.

Response time of a query is one of the parameters concerned by P2P users. We define 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.

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 tradeoffs 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 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 [19] have shown that over 20% of the logical connections in a P2P last 1 minute or less, and around 60% 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 [17] indicated that the median up-time for a node in Gnutella and
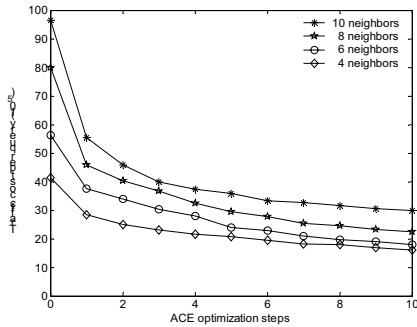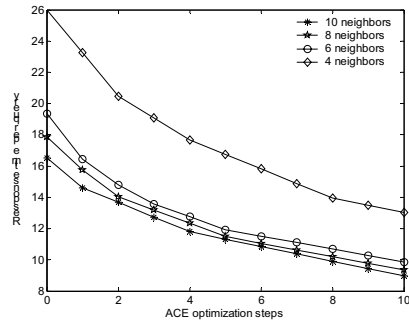
Figure 7 : Traffic reduction vs. optimization step
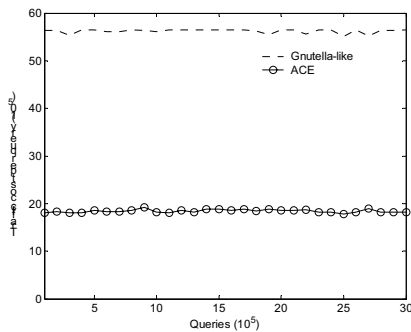

Figure 8: Average Response time vs. opt. step

Figure 9: Average traffic cost reduction in a dynamic P2P environment

Figure 10: Average response time reduction in a dynamic P2P environment

Napster is 60 minutes. Studies in [3] 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% 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 [20], 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 [17]. The mean of the distribution is chosen to be 10 minutes [19]. 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 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 20,000 nodes. We have also simulated ACE
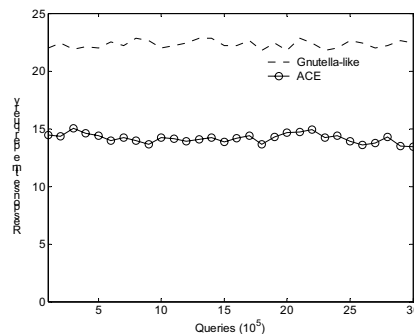
in a real-world P2P topology (based on DSS Clip2 trace). We obtained consistent results on the real-world topology and the generated topologies. We representatively present the results based on 8,000 peers only.

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. This will show that without changing the overlay topology, how many optimization steps are required to reach a better topology matching.

As we have discussed, the first goal of ACE schemes is to reduce traffic cost as much as possible while retaining the same search scope. Figure 7 shows the traffic cost reduction of ACE, where the curve of 'c_n *neighors*' means the average traffic cost caused by a query to cover the search scope in 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 8000 peers. ACE may reduce traffic cost by around 65% and it converges in around 10 steps.

The simulation results in Figure 8 show that ACE can shorten the query response time by about 35% after 10 steps. The tradeoff between query traffic cost and response time has been discussed in [23]. P2P systems with a large number of average connections offer a faster search speed while increasing traffic. One of the strengths of ACE schemes is that it reduces both query traffic cost and response time without decreasing the query success rate.
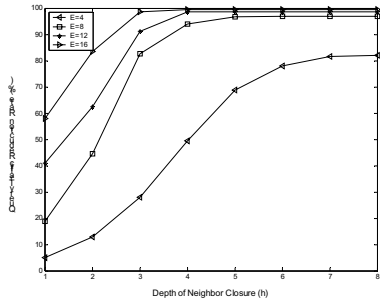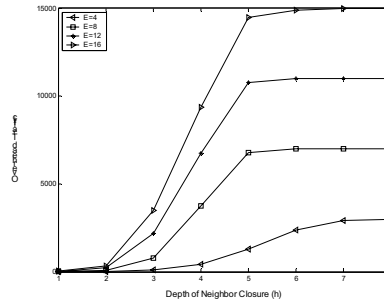
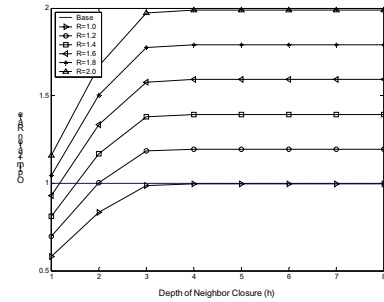**Figure 11: Query traffic reduction rate**

Figure 12: Overhead traffic

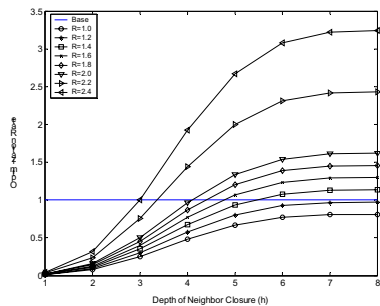
Figure 13: Optimization rate (E=16)

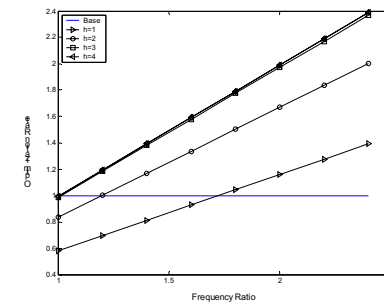**Figure 14: Optimization rate (E=4)**

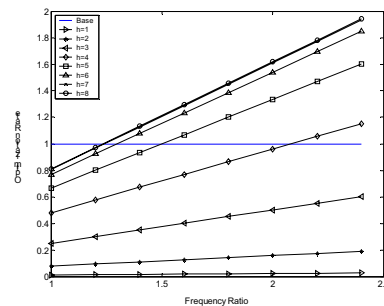**Figure 15: Optimization rate (E=16)**

Figure 16: Optimization rate (E=4)

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; and the frequency for ACE at every peer to conduct optimization operations is twice per minute. Figure 9 shows the average traffic cost per query of Gnutella-like P2P systems, ACE enabled Gnutella. Note that here the traffic cost includes the overhead needed by each operation in the optimization steps. ACE could significantly reduce the traffic cost while retaining the same search scope. Figure 10 shows that with reduction of the traffic, the queries' average response times of ACE are also reduced in a dynamic environment.

In a dynamic P2P environment, we simulate ACE employed together with other approaches, such as response index caching scheme or some forwarding based strategies. We obtained very good results. For example, using a 200-item size cache at each peer, ACE with index cache will reduce 75% of the traffic cost and 70% of the response time. Due to the page limitation, we do not show the detailed curves here.

5.3. The Impact of Optimization Depth

Figure 11 illustrates the query traffic reduction rate over blind flooding versus the depths of neighbor closure to construct overlay trees. Different curves correspond to the performance 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 the query traffic is hard to be further reduced.

Figure 12 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.

Figures 13 and 14 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 increase anymore. Figures 13 and 14 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.

Figures 15 and 16 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 relative 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 Figure 15 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 Figure 15, the minimal value of h is 2 for $R=1.5$, and is 1 for $R=2$. Comparing Figure15 with Figure 16, 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 mismatching problem. Our simulation shows that the average cost of each query to reach the same scope of nodes is reduced by about 65% when using our 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 35%. 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 non-flooding neighbor's neighbor list to minimize replacement overhead. In our simulations, we only use random policy to replace a non-flooding neighbor by a random selected candidate. We are studying several alternatives to choose the candidate. For example, the naïve 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 closest policy in which the source will probe the costs to all of the non-flooding neighbor's neighbors, and select the closest one.

References

[1]   KaZaA, http://www.kazaa.com
[2]   Gnutella, http://gnutella.wego.com/
[3]   R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding Availability," Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS), 2003.
[4]   Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," Proceedings of ACM SIGCOMM, 2003.
[5]   Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," Proceedings of ACM SIGMETRICS, 2000.
[6]   E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," Proceedings of SIGCOMM, 2002.
[7]   B. Krishnamurthy and J. Wang, "Topology modeling via cluster graphs," Proceedings of SIGCOMM Internet Measurement Workshop, 2001.
[8]   Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "AOTO: Adaptive Overlay Topology Optimization in Unstructured P2P Systems," Proceedings of GLOBECOM, San Francisco, USA, 2003.
[9]   Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-Aware Topology Matching in Unstructured P2P Systems," Proceedings of INFOCOM, 2004.
[10] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," Proceedings of the 16th ACM International Conference on Supercomputing, 2002.
[11] E. P. Markatos, "Tracing a large-scale peer to peer system: an hour in the life of gnutella," Proceedings of the 2nd IEEE/ACM International Symp. on Cluster Computing and the Grid, 2002.
[12] D. A. Menasce and L. Kanchanapalli, "Probabilistic Scalable P2P Resource Location Services," ACM SIGMETRICS Performance Evaluation Review, vol. 30, pp. 48-58, 2002.
[13] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for Internet hosts," Proceedings of ACM SIGCOMM, 2001.
[14] S. Patro and Y. C. Hu, "Transparent Query Caching in Peer-to-Peer Overlay Networks," Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS), 2003.
[15] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," IEEE Internet Computing, 2002.
[16] Ritter,Why Gnutella can't scale. No, really, http://www.tch.org/gnutella.html
[17] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," Proceedings of Multimedia Computing and Networking (MMCN), 2002.
[18] S. Saroiu, K. P.Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002.
[19] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," Proceedings of ACM SIGCOMM Internet Measurement Workshop, 2002.
[20] K. Sripanidkulchai,The popularity of Gnutella queries and its implications on scalability, http://www2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html
[21] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," Proceedings of ICDCS, 2003.
[22] B. Yang and H. Garcia-Molina, "Efficient search in peer-to-peer networks," Proceedings of ICDCS, 2002.
[23] Z. Zhuang, Y. Liu, L. Xiao, and L. M. Ni, "Hybrid Periodical Flooding in Unstructured Peer-to-Peer Networks," Proceedings of International Conference on Parallel Processing, 2003.