

Approaching Optimal Peer-to-Peer Overlays

Yunhao Liu¹, Li Xiao², Abdol-Hossein Esfahanian², Lionel M. Ni¹

¹*Dept. of Computer Science, Hong Kong University of Science and Technology, Hong Kong*

²*Dept. of Computer Science and Engineering, Michigan State University, USA*

Abstract

In unstructured peer-to-peer (P2P) systems, there exists a serious topology mismatch problem between physical and logical network. We first analyze the relationship between the property of the overlay and the corresponding message duplications incurred by queries in a given overlay, and prove that computing an optimal overlay with global knowledge is an NP-hard problem. Motivated by the analysis results, we design a distributed overlay optimization algorithm, THANCS, to attack topology mismatch. We demonstrate its performance by comprehensive simulations in dynamic environments. The proposed THANCS has three major strengths. First, it does not need any global knowledge. Second, its optimization convergent speed is fast. Third, it is orthogonal with other types of advanced search approaches.

1. INTRODUCTION

As an emerging model of communication and computation, peer-to-peer networking dictates a fully distributed, cooperative network design, and has recently gained significant acceptance. Most of the current popular P2P applications, such as Gnutella and KaZaA, where peers voluntarily provide data of interest as well as receive it, operate on decentralized unstructured networks. In these systems, peers connect in an ad-hoc fashion, and the placement of documents is not controlled.

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. Maintaining and searching operations of a Gnutella peer are specifically described in [5]. The most popular search method used in these systems is blind flooding, where each peer can issue queries and a query is broadcast and re-broadcast until the desired data is found or the query has been forwarded to peers within a given number of hops, which is 7 in Gnutella. 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.

Studies in [22] have shown that P2P traffic contributes the largest portion of Internet traffic, based on measurements on some popular P2P systems, such as FastTrack (including KaZaA and Grokster) [1], Gnutella, and DirectConnect. The inefficient P2P overlay topology is a major reason for the heavy P2P traffic [12, 14, 15, 21]. For instance, a peer randomly choosing logical neighbors without any knowledge about the underlying physical topology causes topology mismatch between the P2P logical overlay network and physical underlying network. Figure 1 gives an example. For a query message sent along the overlay path $A \rightarrow C \rightarrow B$, node B is visited twice. Although B is a peering node, B is visited as a non-peering node the first time. Because of the mismatch problem, the same message may traverse the same physical links, such as BE, EF and FC in Figure 1, multiple times, causing a large amount of unnecessary traffic, and increasing the P2P users' query search latency as well.

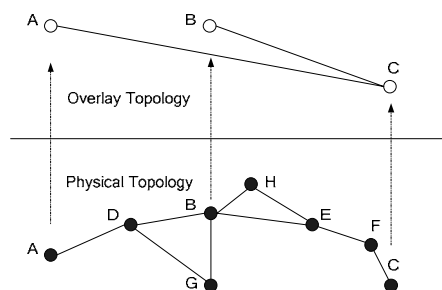


Figure 1: An example of topology mismatch problem

In this paper, we first investigate the relationship among query traffic cost, message duplications, average query response time, and P2P overlay topologies. We prove that given a physical topology and a subset of all the physical nodes as peering nodes, even if we have global knowledge of these two topologies, which is hardly possible in real P2P systems, finding the optimal overlay is still NP-hard.

To address topology mismatch problem and approach an optimal overlay in P2Ps, we propose a completely distributed heuristic algorithm, called Two Hop Away Neighbor Comparison and Selection (THANCS), among peers in Gnutella-like systems or among the

super-peers in KaZaA-like systems. The overhead of THANCS is trivial compared with the query cost savings. We show the effectiveness of THANCS by contrasting the performance of search efficiency on a THANCS optimized overlay and an optimal overlay obtained by an offline brute force algorithm. Our simulation studies also show that the total traffic and response time of the queries can be significantly reduced by THANCS without shrinking the search scope. We also show that THANCS is orthogonal to most of the existing search enhancement approaches, such as Random Walk[16] and Response Index Caching[17, 23].

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

2. RELATED WORK

Many efforts have been made to avoid the large volume of unnecessary traffic in unstructured P2P systems based on overlay topology optimization [8, 13, 18-20, 25].

End system multicast, Narada, was proposed in [8], 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 in forming the graph and trees in a large scope, and does not consider the dynamic joining and leaving characteristics of peers. Our proposed THANCS is easy to implement and adaptive to the dynamics of P2P systems.

Researchers have also considered clustering close peers based on their IP addresses (e.g., [13, 19]). Recently, researchers in [25] 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. Gia [7] introduced a topology adaptation algorithm to ensure that high capacity nodes are indeed the ones with high degree and low capacity nodes are within short reach of high capacity nodes. It addresses a different matching problem in overlay networks, but does not address the topology mismatch problem between the overlay and physical networks.

3. OPTIMAL OVERLAY PROBLEM

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. We model a P2P network based on the following assumptions. First, an overlay connection

between a pair of peering nodes consists of a number of physical links which form a shortest path between the pair of end nodes in the physical topology, and Internet paths are relatively stable [26]. Second, we assume that the same size packets traversing the same physical link in a short period of time will have similar delay, as assumed by many other measurement applications [9, 24].

3.1 Modeling P2P Networks

Graph theoretic terms not defined here can be found in [6]. We model a communication network by an undirected graph $G = (V, E)$ where the vertex set V represents units such as hosts and routers, and the edge set E represents physical links connecting pairs of communicating unit. For instance, G could model the whole or part of the Internet.

Given an undirected graph $G=(V, E)$ modeling an interconnection network, and a subset $X \subseteq V(G)$ of communicating units (peers), we construct a corresponding complete edge weighted graph $D=(V, E)$, where $V(D) = X$, and the weight of each $uv \in E(D)$ is equal to the length of a shortest path between peer u and peer v in G . Note that D is a complete graph, that is, it includes all possible edges, and is referred to as the *distance graph* of G .

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

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

Second, the overly network H should be degree-bounded to balance the load of peers since an extremely large connection degree (the number of peering neighbors) of a peer causes heavy loading and query dropping, and an extremely small degree of a peer causes very long response times. We identify that the

upper bound of a peer's degree for the peer to not become a query bottleneck is 8 by using a modified LimeWire [2] client with logging functionality. Due to the page limit, we do not provide the details of this experiment.

3.2 Metrics

To measure the quality of a P2P overlay topology in terms of search process, we mainly focus on the following two metrics.

Traffic cost is one of the parameters network administrators are seriously concerned with. We define the traffic cost as network resource used in an information search process of P2P systems, which is mainly a function of consumed network bandwidth and other related expenses. Specifically, we assume all the messages have the same length, so when messages traverse an overlay connection during the given time period, the traffic cost (C) is given by: $C=M \times L$, where M is the number of messages that traverse the overlay connection, and L represents the number of physical links in this overlay connection.

Response time of a query is defined as the time collapsed from when the query is issued until when the source peer receives a response from the first responder.

3.3 Unnecessary Message Duplications in Overlay Connections

There is a tradeoff between traffic cost and query response time. It is meaningless to purely optimize one metric without considering the other. Thus, before discussing the optimal overlay in depth, we need to establish the relationship between message duplications in overlay connections and the number of overlay links.

Generally, as long as loops exist in search paths, there must be message duplications in overlay connections. Some peers 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 it has received before, the peer will discard the message.

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

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

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

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

Step: We want to show that the assertion is correct when $|E(H)| = k$. Let H be an arbitrary overlay connected graph with $|E(H)| = k$ edges. Since H has more than $|V(H)|$ edges, then H must contain at least one cycle C . Let uv be an edge of C , and now consider the graph $H' = H - uv$. Since H was assumed to be connected, and uv belongs to a loop, H' is also connected. Further, since $|E(H')| < k$, then by induction hypothesis, there is an arbitrary query message, call it Q , issued by an arbitrary peer with a sufficiently large TTL value that results in $2(|E(H')| - |V(H')| + 1)$ duplicate messages in the overlay graph H' . Now if we initiate the same query Q in H , sticking to the same message propagation as in H' , at some point peers u and v , the end vertices of the edge uv , will be informed. These peers in turn can potentially send messages to each other, generating two unnecessary messages on the link uv . Therefore, the number of duplications in H for the message query Q will be $2(|E(H')| - |V(H')| + 1) + 2$. Rewriting this expression in terms of $E(H)$ and $V(H)$, and remembering that $V(H) = V(H')$, we get $2(|E(H')| - |V(H')| + 1) + 2 = 2(|E(H')| + 1 - |V(H')| + 1) = 2(|E(H)| - |V(H)| + 1)$, and thus the theorem. ■

More message duplications in overlay links means more traffic cost. In an overlay $H=(V, E)$ with p nodes, concerning the traffic cost only, the best case is when $H=(V, E)$ is a spanning tree reduced from the complete distance graph $D=(V, E)$. In this case, a query incurs $n-1$ messages in overlay connections to reach all peers. However, the average query response time will be long compared with a graph with more overlay connections. Indeed, the number of overlay connections balances the traffic cost and average query response time. As it is impossible to minimize these two metrics simultaneously, we seek an optimal overlay when the cardinality of E is given.

3.4 Optimal Overlay Problem

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

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

We now define the Degree-bounded Minimum Average Distance (DMAD) overlay problem.

Definition 2: Let graph $G=(V, E)$ represent a physical network and let graph $D=(V, E)$ be its distance graph for a set of communicating peers, as defined earlier. Recall that D is a complete graph. Furthermore, let k be an integer such that $\delta(D) \leq k \leq \Delta(D)$, where $\delta(D)$ and $\Delta(D)$ are the minimum and the maximum degree of

D , respectively. A DMAD overlay graph $H=(V, E)$ is a connected spanning subgraph of D having the following properties: (a) $\Delta(H) \leq k$, (b) $AD(H)$ is as small as possible, subject to (a).

The DMAD problem is a generalization of the degree-bounded connected subgraph problem (DBCS) which asks the following question [10]:

Instance: Graph $G=(V, E)$, non-negative integer $d \leq |V(G)|$, positive integer $k \leq |E(G)|$.

Question: Is there a subset $E' \subseteq E(G)$ with $|E'| \geq k$ such that the subgraph $G'=(V, E')$ is connected and has no vertex with degree exceeding d ?

Clearly there is a straightforward polynomial transformation from the DBCS problem to the decision version of the DMAD problem, and thus DMAD problem is NP-hard. ■

4. TWO HOP AWAY NEIGHBOR COMPARISON AND SELECTION (THANCS) ALGORITHM

Knowing that DMAD problem is NP-hard leaves little hope for finding an optimal overlay network even when we have complete information about the overlay network. Worse yet, it is practically impossible for a peer to collect global knowledge of the overlay topology since the number of online users could be millions and these P2P users are randomly coming and leaving. However, it is clear that optimizing inefficient overlay topologies can fundamentally improve P2P search efficiency. In this paper, we propose the design of the Two Hop Away Neighbor Comparison and Selection (THANCS) scheme that effectively attacks the topology mismatch problem and optimizes the overlay topology to approach the optimal solution. THANCS consists of two main components: piggybacking neighbor cost on queries, and neighbor comparison and selection.

4.1 Piggyback Neighbor Distance on Queries

We use network delay between two peers as a metric for measuring the distance between the peers. In THANCS, each peer probes the distances with its immediate logical neighbors and stores the information in its local storage. Peers in Gnutella have a limited number of neighbors, 4 to 6 on average, so the overhead of this operation is trivial for each peer. Since Internet paths are relatively stable [26], to keep the neighbor information up to date, a peer only needs to probe the distance to its new neighbor and modify the neighbor distance information when any of its neighbors is leaving.

Table 1: Piggy Message body

	Neighbor's IP	Neighbor Distance
Byte offset	0 3	4 5

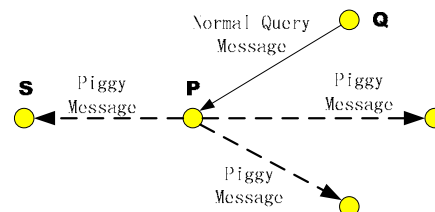


Figure 2: Forwarding piggy messages

For each peer to keep the distance information of its neighbors, we add one special query message type, Piggy Message, by modifying the LimeWire implementation of Gnutella 0.6 P2P protocol [6]. A piggy message has a message body in the format as shown in Table 1. It includes two fields: Neighbor's IP Address and Neighbor's Distance. Since a piggy message will be piggybacked by a query message, it does not need the Gnutella's unified 23-byte header.

A peer constructs a piggy message for each of its neighbors. The idea here is to send the piggy message of one neighbor to all the other neighbors. To avoid increasing the number of messages, THANCS is designed to piggyback a piggy message on a query message. For example, peer P in Figure 2 constructs a piggy message for its neighbor Q , which contains Q 's IP address and the distance to Q . When P receives a query from Q , this piggy message will be piggybacked by the query message that will be forwarded to all the other neighbors of peer P . The payload type of a query message piggybacking a piggy message can be defined as 0x82 to distinguish it from a normal query message (0x80). The payload length of such a query message will be increased by 6. After receiving such a query message, each of the other neighbors will detach the piggy message from the query message, record the distance information of P to Q , and further process the query and forward the query message if necessary. The piggybacked piggy message will not be forwarded, but it's possible for this query message to piggyback another piggy message.

However, peer P may receive queries from Q very frequently. It is obviously not necessary for all query messages from Q to piggyback the piggy message. One critical issue to be examined here is which incoming queries should piggyback the piggy message. Although a piggy message is only 6-byte long, a large amount of duplicated piggy messages still inserts a lot of unnecessary traffic into the network. In this study we present two policies for this selection: pure probability-based (PPB) policy and new neighbor triggered (NNT) policy.

The PPB policy is simply providing a pre-defined probability, α , for a query to piggyback a piggy message. A smaller α means there will be fewer queries piggybacking piggy messages. Note that in this design, it is not to say that once a query starts to piggyback a piggy

message, the piggy message will be forwarded with the query message until the query message is dropped. Instead, each piggy message will be piggybacked for only one single hop. The piggy message will be detached from the query message from the previous peer. With the probability α , this query message may piggyback another piggy message in the current peer. The major advantage of this policy is its simplicity.

In the NNT policy, queries will not piggyback piggy messages until a peer finds a neighbor who just joined the P2P network. As shown in Figure 3, all peers monitor new neighbors' coming. For example, when a peer P gets a new neighbor N , P does the following two operations. First, peer P probes the distance with N and constructs a piggy message. Peer P lets the first query message coming from N piggyback this piggy message. This query message with piggybacked piggy message will be forwarded to all P 's existing logical neighbors except peer N . Second, the first incoming query from each of P 's existing neighbors after peer N 's coming will piggyback a corresponding piggy message (with the distance to this neighbor) when it is forwarded to N . However, the query message will not piggyback a piggy message when it is forwarded to the other existing neighbors, as illustrated in Figure 3. Another option is to let the first query message from P 's previous neighbors piggyback all the piggy messages (except the one for the new neighbor) to the new neighbor, N . Compared with the PPB policy, NNT policy is relatively complicated while it has smaller traffic overhead. We will further discuss the selection of the probability for this policy, and the selection of these two policies in detail in Section 5.

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

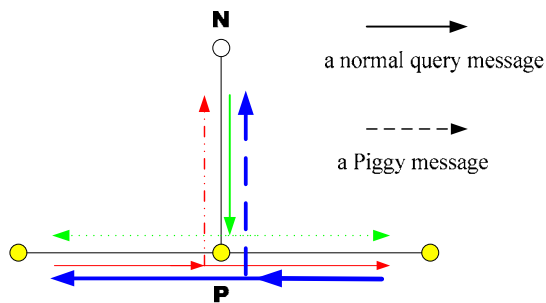


Figure 3: New neighbor triggered policy

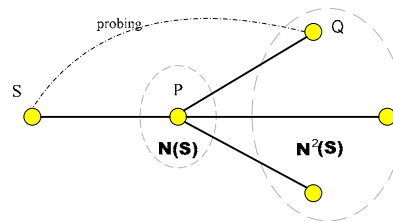


Figure 4: Probing two hop away neighbors

4.2 Neighbor Comparison and Selection

The behavior of THNACS peers in this neighbor comparison and selection component is demonstrated through an example in Figure 4. An arbitrary peer, S , probes the distance to all the known un-probed $N^2(S)$ peers. The distance of SP is known to S . When peer S receives a piggy message from peer P with the distance of PQ , there will be two cases.

Case 1: Peer Q is also a direct neighbor of peer S , i.e. $Q \in (N(S) \cap N^2(S))$. In this case, peer S will compare the cost of SQ , SP , and PQ . If SQ or SP is the longest in these three connections, S will put the longest connection into its *will-cut list* that is a list of connections to be cut later, e.g. 50 second later. If PQ is the longest, peer S will do nothing because the system is fully distributed and peer P or Q will disconnect PQ shortly. A peer will not send or forward queries to connections in its *will-cut list*, but these connections have not been cut in order for query responses to be delivered to the source peer along the inverse search path.

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

5. PERFORMANCE EVALUATION

Using BRITE, we generate three physical topologies, each with 27,000 nodes. For P2P topologies we use DSS Clip [4] trace. We simulate flooding search used in a Gnutella network by conducting the Breath First Search algorithm from a specific node. Basic settings of the workloads used in this simulation follow a 200-day trace

of KaZaA P2P traffic collected at University of Washington [11].

5.1 Effectiveness of THANCS in Static Environments

We first study the effectiveness of THANCS in a static P2P environment where the peers do not join and leave frequently. We have proven in Section 3.4 that the DMAD problem is NP-hard, so this simulation is conducted based on a small size overlay topology. Specifically, in this simulation, 128 out of 27,000 nodes are randomly selected as peering nodes. We then generate 100,000 queries, and simulate flooding search for different overlay topologies including a Gnutella-like overlay, a THANCS optimized overlay, and an optimal overlay that is obtained by a brute force algorithm. Here we have two assumptions. First, we assume that the underlying physical layer uses shortest path routing with delay as the metric. Second, we assume the average number of neighboring peers is 4, or there are totally 256 overlay connections in the 128-peer topology. We run the simulation 20 times with different random choices of peers and report the average.

In Figure 5 we show the percentage of query responses along mismatched paths of three schemes. An optimal overlay effectively replaces all the mismatched paths as shown by Figure 5. THANCS is proven an effective approach optimizing up to 58% out of the 70% mismatched paths. As a result, system performance is improved significantly. Figure 6 shows that the traffic cost, c , decreases by up to 75% when optimization operations of THANCS are conducted. In Figure 7 we show that THANCS can effectively shorten the query response time by about 60%. Although there is no bound indicating how well THANCS could do, the simulation results show the overall performance of THANCS is close to the optimal solution in a static environment.

5.2 THANCS in Dynamic Environments

We further evaluate the performance of THANCS in a dynamic environment, where peers are randomly coming and leaving. We first discuss how to select a better policy from PPB and NNT. We then evaluate the effectiveness of THANCS by varying the size of the

P2P system and the average number of connections in the overlay network.

Selection of PPB or NNT policy

THANCS is a completely distributed approach, so each single peer determines its own operations independently without any help from a central server. One key issue for peers is which query message should be piggybacked with a piggy message. We simulate flooding based search for 50,000 queries, which means around 30 minutes in a real P2P environment.

Figure 8 plots the performance of THANCS on solving the topology mismatch problem when using NNT policy and PPB policy. The curve of ‘PPB- α ’ shows the performance of PPB, where the probability of an incoming query to piggyback a piggy message is $\alpha\%$. Figure 9 plots the traffic overhead of these two policies.

The most attractive advantage of PPB policy is its simplicity. In PPB policy, each peer blindly selects some of the queries using a straightforward rule to piggyback piggy messages without monitoring logical neighbors’ coming or leaving. Indeed, the overhead of PPB is not intolerable, although it is high as shown in Figure 9. In this simulation, the mismatching reduction of PPB-1.0 is close to NNT policy, but PPB-1.0 has the largest traffic overhead. This overhead in the simulated environment, however, is only equivalent to the traffic of a couple of queries per minute, accounting for less than 2% of the traffic savings by THANCS. The NNT policy can further reduce optimization traffic overhead, but it is a bit more complicated compared with PPB policy. Simulation results in Figures 8 and 9 show that NNT outperforms PPB in the sense that it has a higher optimization rate and a smaller traffic overhead. Thus, we adopt NNT policy. The traffic overhead of NNT is less than 0.3% of the traffic savings.

THANCS in Various Size P2P Networks

To better evaluate the performance of THANCS in dynamic environments, we select the size of P2P overlay networks from 2,000 to 8,000. We simulated the search process on each overlay for 30 minutes, repeat with different random seeds for 20 times, and plot the average results with and without the THANCS scheme.

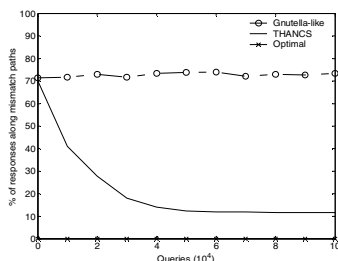


Figure 5: % of query responses along mismatched paths

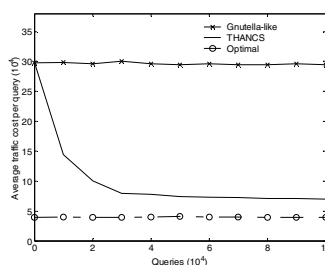


Figure 6: Average traffic cost per query.

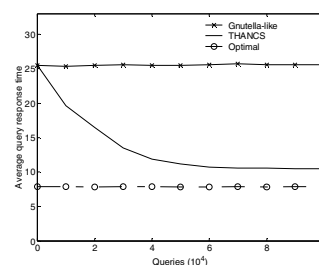


Figure 7: Average query response time.

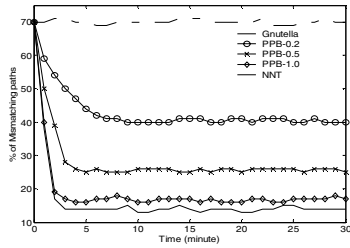


Figure 8: Comparison of NNT and PPB policies

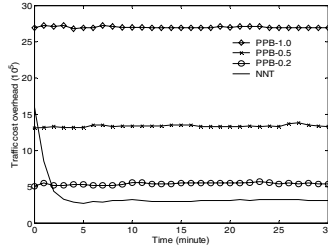


Figure 9: Traffic cost overhead with NNT or PPB policies

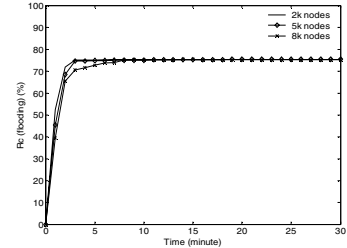


Figure 10: Traffic cost reduction with 2k to 8k node overlays

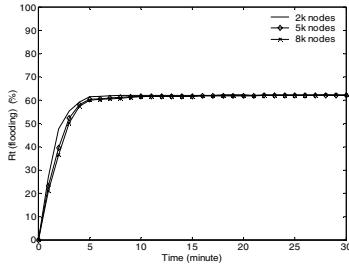


Figure 11: Response time reduction in 2k to 8k node overlays.

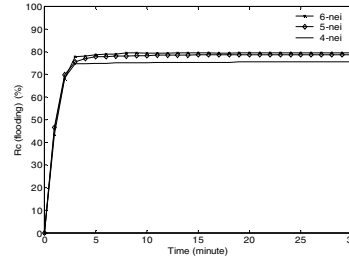


Figure 12: Traffic cost reduction in 5k-node overlays with 4 to 6 neighbors in average.

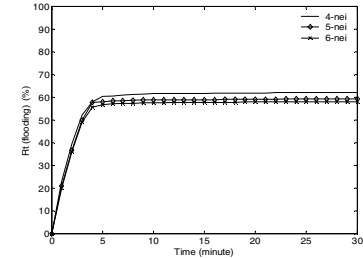


Figure 13: Response time reduction in 5k-node overlays with 4 to 6 neighbors in average.

Here we use two metrics, traffic cost reduction rate ($R_c(*)$) and response time reduction rate ($R_t(*)$). $R_c(*)$ is defined by:

$$R_c(*) = \frac{C(\text{Gnutella-like}) - C(\text{THNACS})}{C(\text{Gnutella-like})} \times 100\%$$

where $C(\text{Gnutella-like})$ represents the traffic cost incurred by searching all the peers using the given mechanism (*), such as blind flooding, in a Gnutella-like overlay. $C(\text{THNACS})$ represents the traffic cost incurred in THNACS enabled P2P networks. $R_t(*)$ is given by:

$$R_t(*) = \frac{T(\text{Gnutella-like}) - T(\text{THNACS})}{T(\text{Gnutella-like})} \times 100\%$$

where $T(\text{Gnutella-like})$ denotes the average response time of queries in Gnutella-like overlays, and $T(\text{THNACS})$ is that of THNACS enabled systems. In this simulation, we study the performance of THNACS based on the blind flooding mechanism, and we will demonstrate the effectiveness of THNACS on other existing advanced search strategies shortly.

The variance of data in different simulation runs is very small, so we do not show the confidence intervals in the figures. Figures 10 and 11 show that the performance of THNACS is not strongly dependent on the size of the network. With THNACS enabled, $R_c(\text{flooding})$ is up to approximately 75%, which means that the total network traffic incurred by blind flooding search will be decreased by 75% without shrinking the search scope. At the same time, average query response time is also decreased by approximately 60%. We also tried THNACS in physical topologies based on real

AS-level Internet topologies [3] and the results proved consistent with those of generated topologies.

Different average number of logical neighbors

In previous discussions in Section 3.3, we proved the relationship between the number of overlay connections and message duplications. Larger average numbers of logical neighbors means more overlay connections in the networks. The traffic cost increases when peers have more logical neighbors. Meanwhile, average response time decreases. There is a tradeoff between traffic cost and average response time. We demonstrate the effectiveness of THNACS in various average number of logical neighbors from 4 to 6, and plot the results in Figures 12 and 13. As shown in Figure 12, $R_c(\text{flooding})$ is higher than 75% when peers have 4 neighbors, and it grows slightly when average connection number is increased. Figure 13 shows that the reduction in response time drops slightly when peers have more edges, but still at a level of higher than 55%.

We also design schemes which combine THNACS with two popular advanced search mechanisms, Random Walk [16], and Index Caching[23]. We first simulate search under a simple random walk scheme. For each query we issue 30 walkers. We then combine THNACS with Index Caching, in which query responses are cached in passing peers along the returning path. Each peer keeps a local cache and a response index cache. Results show that THNACS reduces traffic cost and response time of the Index caching scheme by approximately 80% and 39% respectively, and reduces

both traffic cost and response time of the Random Walk by more than 55%.

6. CONCLUSION AND FUTURE WORK

We model the unstructured P2P systems based on our observations and implementations of Gnutella peers. We then study the relationship between the property of the overlay and the corresponding message duplications incurred by queries in a given overlay, and prove that even with global knowledge, computing an optimal overlay is an NP-hard problem. We refer to this problem as the *Degree-bounded Minimum Average Distance (DMAD) overlay problem*.

We propose a completely distributed algorithm, THNACS, which effectively optimizes P2P overlays to approach an optimal overlay topology. THNACS is scalable in the sense that it does not need any global knowledge about the whole system, and each single peer performs operations independently. By comprehensive simulations, we show that the performance of THNACS is close to an optimal solution.

Future research will be in the following directions. First, we will evaluate and integrate existing orthogonal advanced search methods so as to develop a comprehensive search protocol to make unstructured P2P systems more scalable. Second, we intend to deploy and test THNACS enabled prototypes in Planet Lab and implement them in real systems.

Acknowledgement

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

References

- [1]"Fasttrack," <http://www.fasttrack.nu>
- [2]"Limewire," <http://www.limewire.com>
- [3]"NLANR, National Laboratory for Applied NNetwork Research," <http://moat.nlanr.net/Routing/rawdata>, 2001
- [4]"The Gnutella Protocol Specification v4.0," <http://www.clip2.com/GnutellaProtocol04.pdf>
- [5]"The Gnutella protocol specification 0.6," <http://rfc-gnutella.sourceforge.net>
- [6]G. Chartrand and O. R. Oellermann, "Applied and Algorithmic Graph Theory," *McGraw-Hill, New York*, 1993.
- [7]Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," in proceedings of ACM SIGCOMM, 2003.
- [8]Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," in proceedings of ACM SIGMETRICS, 2000.
- [9]N. Duffield, F. Presti, V. Paxson, and D. Towsley, "Inferring Link Loss Using Striped Unicast Probes," in proceedings of IEEE INFOCOM, 2001.
- [10]M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," *Freeman, New York*, 1979.
- [11]K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," in proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), 2003.
- [12]S. Jiang, L. Guo, and X. Zhang, "LightFlood: An Efficient Flooding Scheme for File Search in Unstructured Peer-to-Peer Systems," in proceedings of International Conference on Parallel Processing (ICPP), 2003.
- [13]B. Krishnamurthy and J. Wang, "Topology Modeling via Cluster Graphs," in proceedings of SIGCOMM Internet Measurement Workshop, 2001.
- [14]Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-Aware Topology Matching in Unstructured P2P Systems," in proceedings of IEEE INFOCOM, 2004.
- [15]Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "A Distributed Approach to Solving Overlay Mismatch Problem," in proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS), 2004.
- [16]Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-peer Networks," in proceedings of the 16th ACM International Conference on Supercomputing, 2002.
- [17]E. P. Markatos, "Tracing a Large-scale Peer-to-peer System: An Hour in The Life of Gnutella," in proceedings of the Second IEEE/ACM International Symp. on Cluster Computing and the Grid, 2002.
- [18]A. Nakao, L. Peterson, and A. Bavier, "A Routing Underlay for Overlay Networks," in proceedings of ACM SIGCOMM, 2003.
- [19]V. N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," in proceedings of ACM SIGCOMM, 2001.
- [20]S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," in proceedings of IEEE INFOCOM, 2002.
- [21]M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing*, 2002.
- [22]S. Sen and J. Wang, "Analyzing Peer-to-peer Traffic across Large Networks," in proceedings of ACM SIGCOMM Internet Measurement Workshop, 2002.
- [23]"The popularity of Gnutella queries and its implications on scalability," <http://www2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html>
- [24]C. Tang and P. K. Mckinley, "On the Cost-Quality Tradeoff in Topology-Aware Overlay Path Probing," in proceedings of the 11th IEEE Conference on Network Protocols (ICNP), 2003.
- [25]Z. Xu, C. Tang, and Z. Zhang, "Building Topology-aware Overlays Using Global Soft-state," in proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS), 2003.
- [26]Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the Constancy of Internet Path Properties," in proceedings of ACM SIGCOMM Internet Measurement Workshop, 2001.