HAND: An Overlay Optimization Algorithm in Peer-to-Peer Systems*

Xiaoming Chen[1], Zhoujun Li [2], Yongzhen Zhuang[3], Jinsong Han[3], and Lei Chen[3]

[1] National Laboratory of Parallel and Distributed Processing, Changsha 410073
[2] School of Computer Science & Engineering, Beihang University, Beijing 10083
[3] Dept of Computer Science, Hong Kong University of Science & Technology, Hong Kong
chxmwp@163.com, lizj@buaa.edu.cn,
{cszyz, jasonhan, leichen}@cs.ust.hk

Abstract. Recently, peer-to-peer (P2P) model becomes popular due to its outstanding resource sharing ability in large-scale distributed systems. However, the most popular P2P system – unstructured P2P system, suffers greatly from the mismatching between the logical overlay and physical topology. This mismatching problem causes a large volume of redundant traffic, which makes peer-to-peer system far from scalable and degrades their search efficiency dramatically. In this paper we address the mismatching problem in an unstructured P2P architecture by developing a distributed overlay optimizing algorithm – HAND (Hops Adaptive Neighbor Discovery). Through comprehensive simulations, we show that HAND significantly outperforms previous approaches, especially in large-scale P2P systems.

Keywords: peer-to-peer, search efficiency, topology mismatching problem.

1 Introduction

Peer-to-peer computing, as a promising model to share and manage huge volumes of information in large-scale distributed systems, has gained more and more attentions in recent years [1-5]. In P2P systems, participating peers form a logical overlay on the top of the physical internet topology, so that they can provide computing and sharing services efficiently. In most of the overlays, neighbors of a peer are always chosen randomly. However this random neighbor choosing may cause severe *topology mismatching* problem, *i.e.* the logical topology does not match the underlying physical topology. In a mismatched overlay, the overlay paths are inefficient in the physical layer, leading to heavy traffic overheads in the search [6, 7]. Figure 1 is an example of topology mismatching. Figure 1(b) is the logical topology and Figure 1(a) is the underlying physical topology. A query message is delivered along the overlay path $A \rightarrow B \rightarrow C \rightarrow D$ in 1(b). However, in 1(a) we can observe that the query traverses twice on the several physical links. For example, the query is sent from A to B, and route through the same path back to A. Duplicated query messages may reach a certain peer

* This work is supported in part by the National Natural Science Foundation of China under Grants No. 60473057, 60573057 and 90604007.

M. Gerndt and D. Kranzlmüller (Eds.): HPCC 2006, LNCS 4208, pp. 290 – 299, 2006.

several times. In this example, node A, who initiates the query, afterwards receives its own query twice. This duplication is because that the overlay path is unaware of the underlying physical topology. The situation of mismatching in a real P2P system is even worse. According to Liu *et al*, topology mismatch problem incurs a large volume of unnecessary traffic, and more than 70% of overlay links are suffered [8]. Many previous studies have tried to address this problem by getting rid of the redundant traffic [1, 9-11]. However, the existing approaches have various drawbacks. Some of them, such as LTM [11], rely on clock synchronization, which is impractical and may lead to tremendous synchronization overheads and delays. Other approaches are based on distance probing. However these approaches bring additional traffic overheads, and work well only in static P2P systems.

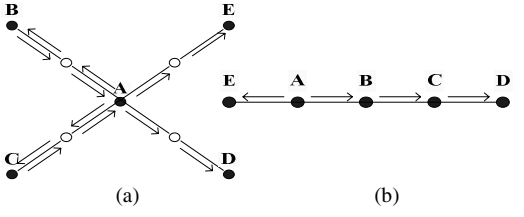

Fig. 1. An example of topology mismatch

We propose a novel algorithm, HAND (Hops Adaptive Neighbor Discovery), to address the topology mismatch problem. We first state that an optimal overlay is a logical overlay that matches the underlying physical topology and name it LCN – logical communication network. Here LCN is the target optimal overlay we want to achieve. LCN can effectively minimize the redundant traffic in the P2P searching. In HAND algorithm, we use a fully distributed triple hop adjustment strategy to adjust the mismatched overlay to LCN.

Our algorithm has several advantages comparing with previous ones. (a) HAND does not need any time synchronization. (b) The traffic overhead involving in the triple hop adjustment is very low. (c) It is applicable to the dynamic P2P environment. (d) HAND can maintain lower query response time.

The remainder of this paper is organized as follows. Section 2 introduces the related work. In Section 3, we present the HAND algorithm. In Section 4, we evaluation the performance of HAND and compare it with previous approaches. Finally, we conclude our work in Section 5.

2 Related Work

Many approaches have been proposed to reduce the large amount of unnecessary traffic caused by topology mismatching in decentralized unstructured P2P systems. They can be categorized into three types: forwarding-based, cache-based and overlay optimization approaches [11].

In forwarding-based approaches, a peer only selects a subset of its neighbors to re-broadcast query messages. When peers select neighbors, they use some metrics to determine what kind of neighbors should be selected. These metrics are some statistic information, including the number of query responses received from neighbors, the latency to connections to neighbors, and so on. Some algorithms use multiple metrics to select neighbors [10]. These forwarding-based approaches can improve the search efficiency. However, they shrink the search scope.

Cache-based approaches can be further divided into two sub-types: data index caching and content caching. Usually, researchers use two cache policies: the centralized caching policy and the local caching policy. Centralized P2P systems provide centralized index servers to keep indices of shared files of all peers. In the local caching policy, each peer maintains some information about other peers. These caching information can be the index of files, query strings and results, replicate data (file contents or query responses), and so on. For example, caching the index of files lets a peer, who maintains these files, process the query on behalf of all nodes within the given hops [12]. Caching can significantly reduce traffic costs and response time.

In the above types of approaches, duplication messages still incurred by the topology mismatching problem still exist. The performance of these approaches is limited.

A lot of overlay optimization algorithms have been proposed [8, 13-18]. HAND also belongs to this type. Here, we can divide all these overlay optimization algorithms into three sub-types: spanning tree based [14], cluster based [13, 15-17], minimum latency first approaches [8, 11]. The spanning tree based approaches first construct a rich graph; and then build a minimum spanning tree from that rich graph [14]. This kind of approaches incurs large traffic overheads to the system. Cluster based approaches use different techniques to identify physically closer nodes and connect them [13, 15-17]. These approaches may shrink the search scope significantly. Minimum latency first approaches use latency as the main metric to measure the distance between peers [8, 11]. However, most of the previous approaches require global latency information to conduct the optimization. In this paper, we also use latency as an optimization metric, but our algorithm HAND only requires local knowledge to perform the overall optimization.

3 HAND Algorithm

To overcome the drawbacks of the previous algorithms, we designed a HAND algorithm that meets the following criteria.

(1) The algorithm performs optimization without any clock synchronization.

(2) The algorithm is fully distributed. It is robust and reliable in decentralized P2P systems.

(3) The traffic overhead incurred in the optimization is trivial.

(4) The algorithm is tolerable to the dynamic P2P environment.

3.1 LCN

The goal of HAND is to adjust a mismatched overlay to a matched one, called a *LCN (logic-communication network)* in the context of this paper. LCN is an undirected

graph $G=(V, E)$, which includes a vertex set V represents peers and an edge set E represents links of all nature neighbors.

Definition 1. If a pair of peers (v_1, v_2) satisfy the following condition it is called nature neighbors:

(I) (v_1, v_2) keeps a direct link in the overlay

(II) The shortest physical path between them does not include any other peer nodes.

That is to say if a peer sends a message to its nature neighbor, this message would not reach any other peer nodes. We call the link between the nature neighbors a *matched link*. If a pair of peer nodes satisfy only the second condition, they are named as *matched neighbors*. Note that matched neighbors are not necessary to be nature neighbors.

Definition 2. Let $G=(V, E)$ represent the topology graph of an overlay network, and $V=\{v_1, v_2, \ldots v_n\}$ be the set of vertices, we define a n x n square matrix $A(G)=(a_{ij})$ where

$$a_{ij}=\begin{cases}1 & \text{if } (v_i, v_j) \text{ is a pair of neighbor peers of overlay} \\ 0 & \text{otherwise}\end{cases} \quad (1)$$

In addition, we define $B(G)=(b_{ij})$as a *matched adjacency matrix***,** where

$$b_{ij}=\begin{cases}1 & \text{if } (v_i, v_j) \text{ is a pair of nature neighbors} \\ 0 & \text{otherwise}\end{cases} \quad (2)$$

Based on the above definition, we derive lemma 1, which can be proved obviously.

Lemma 1. The overlay network matches LCN *if* $A(G)=B(G)$.

3.2 Peer Sequence Characteristics

In practice, it is impossible to build a LCN using the centralized method, which assumes to have the global knowledge. However, a LCN implies the sequence relationship of peers, which lead to our decentralized adjustment algorithm - HAND.

3.2.1 Peer Sequence Mismatching

Definition 3. Let G* be a LCN of the network, and G' be the current overlay. A peer sequence $(v_1, \ldots, v_k)$ of G' is matched if this sequence exists in G* in the same order, otherwise it is mismatched.

In practice, we use a *triple sequences* (v_1, v_2, v_3) to characterize overlay mismatching. A peer sequence of three adjacent nodes in a path is called a triple sequence. Figure 2 and Figure 3 is an example of peer sequence mismatching. Figure 2 represents a LCN and Figure 3 represents a P2P overlay network. The peer sequence *F-A-E* in Figure 3 has a different order in its LCN (*A-E-F* or *F-E-A* in Figure 2), so this sequence is mismatching. Another mismatching example is sequence *F-A-C* in Figure 3, which is split into two other sequence *F-E-A* and *E-A-C* in Figure 2.

Lemma 2. Let G* be LCN of the network, and G' be an overlay. G* and G' is matched if all the triple sequences of the two graph are matched.

3.2.2 Mismatching Detection

Based on Lemma 2, we propose a matching detection algorithm. Suppose a pair of probing messages is sent from a peer v_1 to its neighbors v_2 and v_3 to test the sequence v_2-v_1-v_3. Figure 4, 5 and 6 are used to illustrate the process. We suppose the delays of path (v_1, v_2) and (v_1, v_3) are x' and z', respectively. When the probing message arrives at v_2, it is forwarded to v_3 directly. We therefore obtain the delay of physical path (v_2, v_3) and denote it as y'. Similarly, when the message reaches v_3, it is forwarded to v_2 directly to obtain the delay of physical path (v_3, v_2), which is also y'.

Lemma 3. If $y'=z'-x'$, then the sequence v_2-v_1-v_3 is mismatched and should be adjusted to v_1-v_2,-v_3.

Lemma 4. If $y'=x'-z'$, then the sequence v_2-v_1-v_3 is mismatched and should be adjusted to v_1-v_3,-v_2.

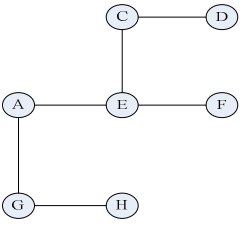

Fig. 2. LCN

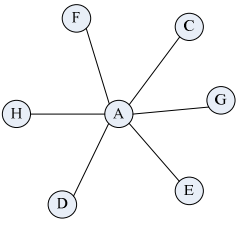

Fig. 3. An inefficient overlay

Fig. 4. The communication in overlay

Fig. 5. The matching triple in Lemma 3

Fig. 6. The matching triple in Lemma 4

Table 1. Route table

Neighbor's IP	*Mark*	*Use-mark*
192.168.0.7	*A*	*B*
192.168.0.111	*B*	*A*

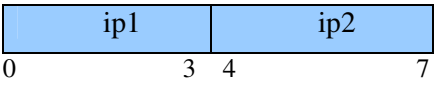

Fig. 7. Probing message body

If v_2-v_1,-v_3 is already matched, $y'=z'+x'$ is satisfied. However, if $y'=z'-x'$, it implies that the matched sequence should be v_1-v_2,-v_3 as in Figure 5, where the delay of path (v_1, v_2), (v_2, v_3) and (v_1, v_3) are x', y' and z' respectively. Similarly, if $y'=x'-z'$, it implies that the matched sequence should be v_1-v_3,-v_2 as in Figure 6. In practice, when the sequence is mismatched, the equation $y'=z'-x'$ and $y'=x'-z'$ can still be false due to error caused by forwarding delays and delay jitters. So we relieve the mismatching condition to be:

$$y' = z' - x' \pm \varepsilon,\ y' = x' - z' \pm \varepsilon \text{ and } y' = x' + z' \pm \varepsilon$$

where ε is a small positive real number.

3.3 Triple Hop Adjustment

In the previous section we describe the mismatching detection of a single sequence. In this section we will describe the mismatching detection of the whole P2P overlay. It includes designing the routing table, probing detect messages and adjusting mismatching sequences.

3.3.1 Routing Table and Probing Messages

Firstly, we describe our routing table and the probing message format. Table 1 shows the format of the routing table. Similar with Gnutella, it comprises a list of neighbor IPs. To guarantee that the pair of neighbors isn't probed repeatedly, we add two items in the routing table, *Mark* and *Use-mark*. *Mark* is a label given to each neighbor, the second node of a triple sequence. *Use-mark* records the candidates of the third node of the triple sequence. Therefore, *Mark* and *Use-mark* is a probing pair. When a peer adds a new neighbor, it must gives this neighbor a *Mark*; and then, this *Mark* should be added to the *Use-mark* of every other neighbor in the routing table; finally, the *Mark* of those old neighbors should be added to *Use-mark* of the new neighbor. When a neighbor leaves the network, we remove its record, and delete its appearance in the *Use-mark* of all other neighbors. Figure 7 is the format of probing message. The probing message has two segments, which are a pair of neighbor's IPs. For example, a probing message for sequence v_2-v_1-v_3 (Figure 4) consists of v_2's IP in the first segment and v_3 's IP in the second segment.

3.3.2 Probing Message Construction

Before we send detect messages, we must build the probing pair. For example in table 1, we find neighbor A, and select *B* from its *Use-mark*. Let A and B be a probing pair (A, B), and remove B from this *Use-mark* of A. We also delete A from the *Use-mark* of neighbor *B*. Finally, we encapsulate their IP in a probing message. In this example ip1 is 192.168.0.7, ip2 is 192.168.0.111. According to the above algorithm, each peer sends the probing message to a pair of neighboring peers. The peer receiving this message then re-sends it to the other peer. The probing continues until the *Use-mark* of the neighbors is empty.

3.3.3 Probing Algorithm

According to Lemma 3 and 4, a pair of probing messages can detect if the original sequence is matched or not and decide how to adjust the sequence to be a matched one. If the probing condition satisfies Lemma 3, edge (v_1, v_3) is deleted and a new edge (v_2, v_3) is added. If Lemma 4 is satisfied, edge (v_1, v_3) is deleted and a new edge (v_2, v_3) is added. In both situations, the routing tables of the corresponding peers are updated accordingly.

The probing message is issued by each node periodically. The HAND algorithm executed at each peer is as follows: for each triple sequence (*h*-*i*-*j*), we use *timestamp*(*i*,*h*) to denote the timestamp when the probing message arrives at *h* and *timestamp*(*i*,*j*,*h*) to denote when the probing message forwarded by *j* arrives at *h*.

4 Performance Evaluation

In this section, we present our simulation results. We first introduce the evaluation metrics and the simulation methodology, and then present the results comparing HAND with LTM.

4.1 Evaluation Metrics

We use two metrics to evaluate the performance of HAND - traffic cost and query response time. Traffic cost is the overhead in message transmission. In our simulation, we use comprehensive traffic cost, which is calculated from a function involving the consumed network bandwidth and the other related parameters. Response time of a query is the time period from when the query is issued until when the source peer receives a response result from the first responder.

In the comparison of HAND and LTM, we use three metrics: traffic cost reduction rate, response time reduction rate and traffic overhead, which are defined in [8]. The traffic cost reduction rate $R_c(*)$ and response time reduction rate $R_t(*)$ is calculated as follows:

$$R_c(*) = \frac{C_{(\text{Gnutella - like})} - C_{(\text{op - algorithm})}}{C_{(\text{Gnutella - like})}} \times 100\% \tag{3}$$

$$R_t(*) = \frac{T_{(\text{Gnutella - like})} - T_{(\text{op - algorithm})}}{T_{(\text{Gnutella - like})}} \times 100\% \tag{4}$$

where (*) represents a given mechanism that is used to search for all peers in a Gnutella-like overlay. Under this mechanism C(*Gnutella-like*) is the traffic cost incurred in a Gnutella-like overlay, C(*op-algorithm*) is the traffic cost incurred in an optimizing algorithm enabled Gnutella-like overlay. T(*Gnutella-like*) is the average response time of queries in a Gnutella-like overlay and T(*op-algorithm*) is that of an optimizing algorithm enabled Gnutella-like overlay. In this simulation we use blind flooding mechanism. Traffic overhead is the per minute traffic cost incurred by an optimizing algorithm on the P2P overlay.

4.2 Simulation Methodology

To accurately simulate the algorithms in P2P systems, we use trace driven simulation. First, we generate two topologies: one for physical application layer and the other for P2P overlay layer. We used a topology generator -BRITE[19] to generate a physical internet topology including 22,000 nodes. The logical topologies are obtained from real traces Clip2 [20]. The network size is ranging from 5,000 to 8,000 peers. The average number of neighbors in the P2P overlay ranges from 6 to 10. Both the physical topology and overlay topology follow the small world and power law properties [21].

We employed flooding search in our simulations. To accurately simulate flooding search, every node issues 0.3 queries per minute in our simulation, which is calculated from the observation in [22]. The file popularity follows a Zipf distribution [23].

In P2P systems, peers join or leave the overlay frequently. We simulate peer joining and leaving behaviors via tuning on/off logical peers. When a peer joins, a lifetime will be assigned to this peer. The lifetime is period the peer stay in the system, counted by seconds. The assignment of lifetime is defined according to the observation in. The lifetime will be decreased by each passing 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 turn on the same number of peers from the physical network to join the overlay.

4.3 Simulation Results

We first simulate our algorithm in the static environment where the peers do not join and leave frequently. In this simulation, we use the overlay topology with 8,000 peers on the top of the physic network with 22,000 nodes. Figures 8 and 9 plot the comparison of HAND and Gnutella. The results show that HAND can quickly converge to an optimal overlay. During this converging process, both the traffic cost and query response time are decreasing. Figure 8 and 9 show that our algorithm can effectively decrease the traffic cost by about 77% and shorten the query response time by about 49% in less than two minutes. In the simulation, each peer has 10 neighbors on average (*i.e.* the average connection of a peer is 10). The tradeoff between query traffic cost and response time is affected by the average connection. P2P systems with a higher average connection offer a faster search speed while increasing traffic. It has been proven that the more neighbor connections, the less query response time, but the more redundant traffic [17].

We then evaluate HAND in a dynamic environment with peer joining and leaving defined in Section 4.2. We evaluate HAND with different average neighbor connections and increasing network size in the dynamic environments.

First, we test two overlays of size 5,000 and 8,000. The average number of neighbors is always 10. Figure 10 shows that their traffic reduction is about 76% and 64% respectively. After converged, the average traffic costs of the two overlays are close. In addition, Figure 11 shows that HAND can effectively shorten the query response time by about 48%~54%. From the above analysis, we know the size of overlay network has a little impact on the effectiveness of our algorithm.

Second, we vary the average number of neighbors from 6 to 10 (6, 8, and 10). This means at the very beginning, the overlay network has 6, 8, and 10 average neighbor connections respectively. Figure 12 and Figure 13 plot the average traffic cost and average query response time running HAND algorithm on the three overlays. At the first few steps of the algorithm, the more neighbor connections, the higher the traffic cost and the shorter the response time. This is consistent with the relationship of neighbor connections, traffic cost and response time discussed in Section 4.3.1. After several steps, three topologies all converge to an optimal topology - LCN. So their average traffic costs and average query response time are almost the same. These figures also show that the reduction of traffic costs is about 42%~64% and query response time is about 40%~54%.

We then compare the performance of HAND with the performance of LTM. Figure 14 and Figure 15 plot traffic reduction rate and response time reduction rate respectively. From figure 14 we can see that HAND and LTM have almost the same

traffic reduction rate. However, the response time reduction rate of HAND is higher than LTM in Figure 15.Their difference is about 4%. Figure 16 plots the traffic overhead of HAND and LTM. The traffic overhead of HAND is much less than that of LTM (including the traffic overhead of clock synchronization). On average, this traffic overhead reduction is about 55%.

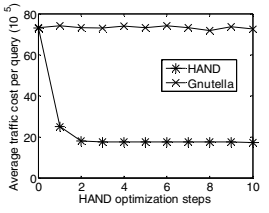

Fig. 8. Traffic reduction in static environments

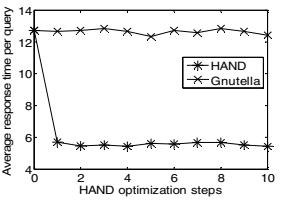

Fig. 9. Response time in static envrionments

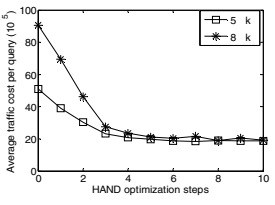

Fig. 10. Traffic cost of different topology size

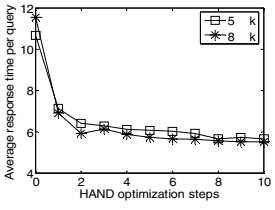

Fig. 11. Response time of different topology size

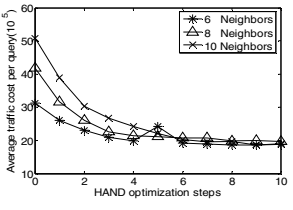

Fig. 12. Traffic cost of different average conn-ections

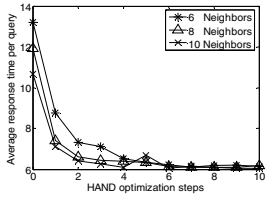

Fig. 13. Response time of different average connections

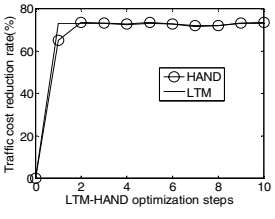

Fig. 14. Traffic reduction rate of HAND and LTM

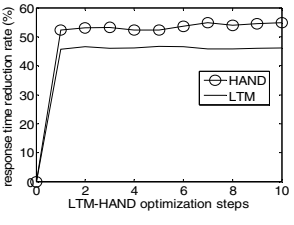

Fig. 15. Response time reduction rate of HAND and LTM

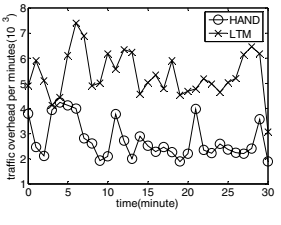

Fig. 16. Per minute traffic overhead of HAND and LTM

5 Conclusions and Future Work

In this paper, we propose a new solution for topology matching problem. The goal is to adjust the mismatching topology to an optimal LCN. We designed an algorithm HAND to optimize the overlay by considering the relation of LCN and the physical topology. This algorithm can be executed distributed by triple hops adjustment. In the future, we intend to deploy HAND into a real P2P streaming system.

References

[1] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," In Proceedings of ACM SIGCOMM, 2003.

[2] A. Nakao, L. Peterson, and A. Bavier, "A Routing Underlay for Overlay Networks," In Proceedings of ACM SIGCOMM, 2003.

[3] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "A Distributed Approach to Solving Overlay Mismatching Problem," In Proceedings of IEEE ICDCS, 2004.

[4] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-Peer Live Streaming," In Proceedings of IEEE INFOCOM, 2006.

[5] L. Xiao, Z. Zhuang, and Y. Liu, "Dynamic Layer Management in Superpeer Architectures," In Proceedings of IEEE Transactions on Parallel and Distributed Systems (TPDS), 2005.

[6] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron., "Practical locality-awareness for large scale information sharing," In Proceedings of IPTPS, 2005.

[7] Y. Liu, L. Xiao, and L. M. Ni, "Building a Scalable Bipartite P2P Overlay Network," In Proceedings of 18th International Parallel and Distributed Processing Symposium (IPDPS), 2004.

[8] Y. Liu, A.-H. Esfahanian, L. Xiao, and L. M. Ni, "Approaching Optimal Peer-to- Peer Overlays," In Proceedings of the 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2005.

[9] 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.

[10] Z. Zhuang, Y. Liu, L. Xiao, and L. M. NI, "Hybrid Periodical Flooding in Unstructured Peer-to-Peer Networks," In Proceedings of IEEE ICPP, 2003.

[11] Y. Liu, X. Liu, L. Xiao , L. M. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems", IEEE INFOCOM 2004, Hong Kong, China, March 2004.

[12] B. Yang and H. Garcia-Molina, "Efficient Search in Peer-to-Peer Networks," In Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS), 2002.

[13] B. Krishnamurthy and J. Wang, "Topology Modeling via Cluster Graphs," In Proceedings of SIGCOMM Internet Measurement Workshop, 2001.

[14] Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," In Proceedings of ACM SIGMETRICS, 2000.

[15] V. N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," In Proceedings of ACM SIGCOMM, 2001.

[16] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," In Proceedings of IEEE INFOCOM, 2002.

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

[18] X. Liu, L. Xiao, A. Kreling, Y. Liu, "Optimizing Overlay Topology by Reducing Cut Vertices", In Proceedings of ACM NOSSDAV, 2006.

[19] BRITE, "http://www.cs.bu.edu/brite/."

[20] Clip2, " The Gnutella Protocol Specification v0.4, Document Revision 1.2."

[21] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," In Proceedings of Multimedia Computing and Networking (MMCN), 2002.

[22] K. Sripanidkulchai, "The Popularity of Gnutella Queries and Its Implications on Scalability," 2005.

[23] L. Breslau, P. Cao, L. Fan, G.Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," In Proceedings of IEEE INFOCOM, 1999.