A Topology-Aware Hierarchical Structured Overlay Network based on Locality Sensitive Hashing Scheme

Guangtao Xue, Yi Jiang, Jinyuan You, Minglu Li
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, P.R. China, 200030
xue-gt, yijiang, you-jy, li-ml@cs.sjtu.edu.cn

ABSTRACT
The DHT scheme without any knowledge about underlying physical topology could cause a serious topology mismatching between the P2P overlay network and the physical underlying network. In this paper, a new mechanism, TSO, is proposed for constructing the two layer topology-aware structured overlay network based on locality sensitive hashing scheme. In TSO, the physical close nodes have been clustered into a local level P2P ring which is regarded as a virtual node in the high level Chord ring of the overall P2P overlay network. A large portion of routing hops previously executed in the global P2P ring are now replaced by hops in local level rings, thus routing overheads can be reduced. The intensive simulation experiments, we have shown the effectiveness of TSO.

Categories and Subject Descriptors
C.2.4 [**Distributed Systems**]

General Terms
Management, Measurement, Performance, Design.

Keywords
topology awareness, peer-to-peer, locality sensitive hash, hierarchical structure

1. INTRODUCTION
There are many researches in peer-to-peer (P2P) overlay architecture. The consistent hash function is used to map the nodes to the overlay network in a structured P2P network, which ensures the load balancing and deterministic of the network. But the DHT scheme could cause a serious topology mismatching between the P2P overlay network and the physical underlying network. Because routing tasks are distributed across all system peers, a routing hop could happen between two widely separated peers with high network link latency which greatly increases system routing overheads.

In this paper, we propose a novel mechanism to construct the two layer topology-aware structured overlay network, called TSO, to alleviating the mismatching problem and relieve routing overheads. In TSO, we create several lower level P2P rings under the higher level P2P ring of the overall P2P overlay network. By using locality sensitive hashing scheme, we group the physical close nodes into a local level P2P ring. Each local level P2P ring will be a virtual node in the Chord ring of the overall P2P overlay network. Routing tasks are first executed in local level rings before they go up to higher level rings, a large portion of routing hops previously executed in the global P2P ring are now replaced by hops in lower level rings, thus routing overheads can be reduced. The intensive simulation experiments, we have shown the effectiveness of TSO.

The rest of the paper is organized as follows, we introduce design of the topology-aware overlay in Section 2. In Section 3, we describe the TSO hierarchical structure. We present our simulation environment and evaluate the TSO routing algorithm efficiency in Section 4. We discuss the related works in Section 5 and give out conclusions and future works in Section 6.

2. DESIGN OF THE TOPOLOGY-AWARE OVERLAY
The design goal of TSO is [1]:

(1) Efficiency: the neighboring hosts in the overlay should be closely located in the underlying network. This can reduce the delivery time and consequently the cost of communications.

(2) Scalability: The overlay construction process should be implemented in a distributed fashion. Reducing the number of long-distance hops traversed per overlay route and reducing the bandwidth usage in constructing the overlay network will improve the scalability of the overlay network.

To achieve this goal, we should take the node locality into account. The DHT scheme, for example SHA-1, is used to produce the ID of the nodes mechanically and map the node to a binary integer representing the position of node in a ring [2] and arrange all the nodes in a structured and deterministic way. But it could cause a serious topology mismatching between the P2P overlay network and the physical underlying network.

We use locality sensitive hash functions instead of the consistent hash function with to produce the node ID and take the locality sensitive hash value of the node's physical position as the node's ID in the ring. In this way, the neighboring hosts in the overlay should be closely located in the underlying network. Briefly, the methods used to make the TSO topology aware are:

(1) Use nodes' coordinate in network instead of the node name to produce node ID.

(2) Use locality sensitive hash functions instead of the consistent hash to produce node ID.

While we use locality sensitive hash functions to produce node ID, the data key is still produced by consistent hash function against the name of the data.

2.1 Locality sensitive hash functions

From[3][4] a family of hash functions *F* is said to be a locality sensitive hash function family corresponding to similarity function *sim(A,B)* if for all $h \in F$ operating on two sets *A* and *B*, we have

$$\mathrm{P}_r\left[h_r(u) = h_r(v)\right] = sim(u,v)$$

Where P is the probability, and *sim(A,B)* ∈ [0,1] is some similarity functions.

2.2 Cosine similarity and LSH

Definition: Let *u* and *v* be two vectors in a k dimensional hyperplane. Cosine similarity is defined as the cosine of the angle between them: cos*(θ(u,v))*. We can calculate cos*(θ(u,v))* by the following formula:

$$\cos(\theta(u,v)) = \frac{|u \cdot v|}{|u| \cdot |v|} \tag{1}$$

Here $|u \cdot v|$ is the scalar dot product of *u* and *v*, and $|u|$ and $|v|$ represent the length of the vectors *u* and *v* respectively.

The LSH for cosine similarity is given by the following theorem [4].

Theorem: Suppose we are given a collection of vectors in a *k* dimensional vector space. Choose a family of hash functions as follows: Generate a spherically symmetric random vector *r* of unit length from this *k* dimensional space. We define a hash function, hr, as:

$$h_r = \begin{cases} 1 : r \cdot u \geq 0 \\ 0 : r \cdot u < 0 \end{cases} \tag{2}$$

Then for vectors u and v,

$$\mathrm{P}_r\left[h_r(u) = h_r(v)\right] = 1 - \frac{\theta(u,v)}{\pi} \tag{3}$$

The proof of the above theorem was given by Goemans and Williamson. From the above equation, we can infer that

$$\cos(\theta(u,v)) = \cos(\pi(1 - Prob_{h \in F}\left[h(u) = h(v)\right])) \tag{4}$$

According to [3], this equation gives us an alternative method for calculates cosine similarity through stochastic way.

And more important to TSO, after having calculated *$h_r(u)$* with d random number vectors for each of the vectors *u*, we can observe that:

$$\mathrm{P}_r\left[h_r(u) = h_r(v)\right] = 1 - \frac{\text{hamming disttance}}{d} \tag{5}$$

Where *d* is the length of the vector. The hamming distance in equation is the hamming distance between *$h_r(u)$* and *$h_r(v)$*. This is an important property to the TSO; the hamming distance between two TSO IDs reflects the network distances between them. But up to now, we haven't found a suitable overlay to use this feature so it cannot be utilized.

2.3 The selection of network coordinate system

Studies in [5] and [6] have shown the latencies matrix between network nodes can be embedded into a low dimensional space.

NPS [5] is based on coordinates computed from modeling the Internet as a geometric space. Each host maintains its own coordinates and network distances to other hosts are predicated by evaluating a distance function over their coordinates. In GNP[6], the coordinates of node are evaluated by the distances to the reference nodes. In NPS, those reference nodes are called landmarks. In GNP and NPS, the coordinating system is running in a centralized way. There is a centre node in NPS. When the system is bootstrapping, the coordinates of the references nodes will be computed by the centre node. The coordinates of the general node are determined by its distances to the reference nodes. This coordinates are embedding into a low dimensional space. The distances between nodes can be inferred by computing the Euclidean distance between the coordinates in the low dimensional space. GNP adopts a hierarchy structure instead of a centre node. The landmarks compute their coordinates themselves.

Vivaldi[7][8] is a fully decentralized solution, which requires no landmarks. In Vivaldi, all nodes are placed in a two dimensional space with weight. Each node computes coordinates for itself. Each time a node communicates with another node, it measures the latency to that node, and then adjusts its coordinates to minimize the error between measured latencies and predicted latencies. Vivaldi requires the user to set only a single parameter, which describes how much to adjust a node's coordinates in response to one new latency sample. This parameter is largely independent of the input and can be set conservatively to ensure accuracy at the cost of convergence time.

Vivaldi applies a similar accuracy to GNP. Both Vivaldi and GNP can also be applied in our system. We choose Vivaldi to provide node coordinates in this paper.

For the simplicity, in this paper, we assume a stable overlay network that no node joining or leaving.

2.4 The dimension of the coordinate

According to [8], through measuring the distances to the other nodes, we can use point in a low dimensional Euclidean space to represent the position of node in network. The distance between two nodes can be calculated from the Euclidean distance of the two nodes in coordinate space. In practice, a multiple dimensional space can be used to represent the distance between nodes. The higher dimension can be applied to afford more accuracy. But the increasing room of the accuracy is limited. The dimension of network coordinates system is also determined by the length of LSH ID. It is because that the LSH ID is generated by the location

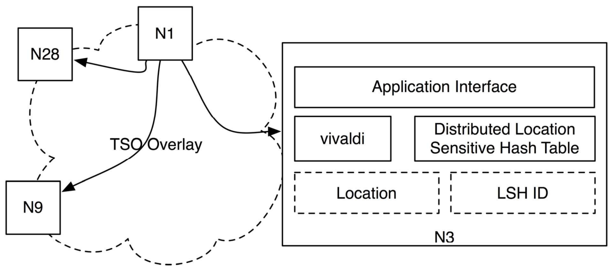

Figure 1. The architecture of TSO

sensitive hash function to the coordinates of node. To work with the locality sensitive hash functions, we choose to use a 20 dimensional coordinate space to represents the position of the node in Internet. When applied the locality sensitive hash functions to this 20 dimensional coordinates, a 20 bits node ID can be produced. The ID length in the TSO is far smaller than Chord and other DHT based structured overlay. For the locality sensitive nature of the ID in TSO, we need not to use a vast ID space to avoid hash collision. So a 20 bits ID space is enough. In addition, a bigger ID space requires a high dimensional coordinate space. This will increase the burden of the Vivaldi. So it is a tradeoff between accuracy and cost. The chosen of dimensional coordinate space is also argued in section 3.4.

2.5 TSO node ID

The node ID in TSO is generated by locality sensitive hashing the coordinates of node. Suppose we use *d* bits binary number as the TSO ID, we should apply *d* hash functions in the node coordinate to produce the d bits in TSO ID. The hash functions used to generate the ID can be represented as:

$$H(x) = \big(h_1(x), h_2(x), ..., h_d(x)\big)$$

The *i*-th bit of the TSO ID is generated by applying the $h_i(x)$ according to the node's coordinates.

As we mentioned in previous section, nodes with the similar coordinates will have the same TSO ID with high probability. The probability can be calculated by:

$$P_r\left[H(x) = H(y)\right] = (sim(x,y))^d = (1 - \frac{\theta(x,y)}{\pi})^d$$

This equation gives us the hint that the dimension of the ID will gain a high probability of collision when the dimension of ID is high. In section 5, we will show the statistics of collisions of TSO ID with the different *d*.

3. TSO HIERARCHICAL ARCHITECTURE

3.1 Architecture

As shown in Fig.1, the TSO contains network coordinate system, application interface and the distributed locality sensitive hash functions. Vivaldi is used in the TSO to provide network coordinate for nodes. Before node joins the TSO overlay, it obtains its network coordinate from Vivaldi in the form of the *d* dimension vector. Then it uses the locality sensitive hash functions to convert it to a *d* bits length binary number. This

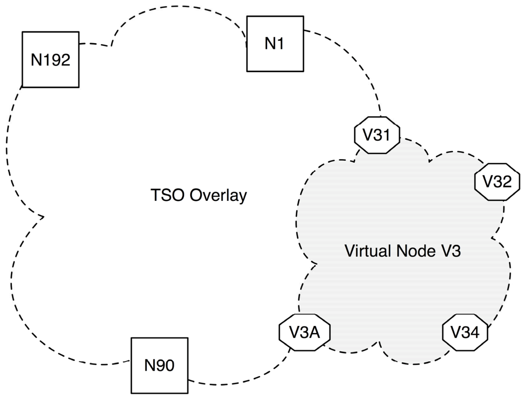

Figure 2a. Virtual nodes in TSO. The cloud in the figure denotes the overlay network.

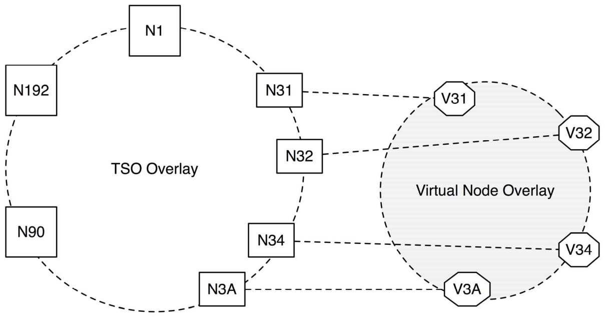

Figure 2b. The actual overlay of TSO. All nodes are placed in a virtual ring. Nodes with the same virtual node ID are close and virtually grouped into the same virtual node.

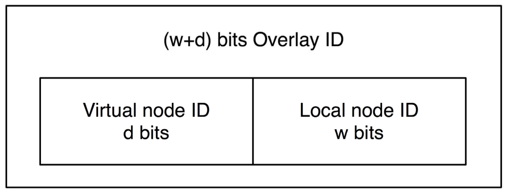

Figure 2c. TSO overlay node ID structure.

binary number is the ID of node used in the TSO virtual node. As nodes may have the same TSO ID with high probability as there exist a high similarity between the coordinates of nodes. In TSO, the node has the same TSO ID been clustered into virtual node. The virtual node is shown in Fig.2. At this time, we name the TSO ID the virtual node ID. The node inside a virtual node has a local ID. The unique node ID in the overlay network is the concatenating of the virtual node ID and the local ID. In this figure, the TSO overlay is a Chord like ring.

3.2 Algorithm to generate the TSO ID

To generate a virtual node ID, node needs to first obtain its Vivaldi coordinates firstly. Then applying the locality sensitive hash functions to the coordinates.

The algorithm is:

(1) Initial the N random d dimensional unit vectors, $r_1 \ldots r_N$;

(2) Get network coordinates from Vivaldi, represented as $C = (c_1, c_2, \ldots, c_N)$;

(3) For each i, calculate e $ID_i = h_r(c_i)$, $r = r_i$;

Once the virtual node ID was generated, the node needs to generate a local ID inside the virtual node. In this paper, we organize the virtual node as a Chord ring. So the local ID can be generated by applying the SHA-1 hash function to the address or name of the node. The TSO node ID is the concatenation of the virtual node ID and the local ID as shown in Fig.2c.

3.3 Virtual node

Virtual node is used in the TSO to handle the ID collision and topology aware. Virtual node is the group of nodes that are similar in network geometry. In TSO, nodes with the same ID are grouped into a local layer overlay and look like a single virtual node in the first layer overlay. The nodes in the same virtual node have the same virtual node ID.

Fig.2a and Fig.2b show that each node in the virtual node has unique local ID with the virtual node head. The head node in the virtual node is the node with the lowest node ID. The organization inside the virtual node can be any structured or unstructured methods. The only requirement is that the implementation of virtual node must be load balancing. For simplicity, the overlay inside the virtual node is the same as the first layer overlay network, which is Chord overlay network. So the TSO can be considered as a two layer Chord overlay network. In Figure.2, node V3 is virtual node; it consists of four nodes, they are N31, N32, N34 and N3A. These nodes have the same virtual node ID-3.

3.4 TSO ID Collision

The hash function used in TSO has the feature that it can preserve the physical locality of the node. TSO use this feature to cluster the nearest nodes into virtual node. The probability of the collision is the function of the dimension of the vector space and the similarity between the object hashed. In Fig.3b, we show the different probability of the collision. The higher the dimension is, the lower the probability at the same similarity. In TSO, the dimension selection is the tradeoff between complexity and collision probability.

Fig.3a show the collision coefficient calculated in the experiments. The collision coefficient is calculated by the following formula:

$$\text{Collision Coeff}=1-\frac{\text{Number Of Different Hash Values}}{\text{Total Number Of Different Objects}}$$

We randomly generated 10000 coordinates with corresponding dimension setting. And locality sensitive hashing scheme hashs those coordinate vectors to the TSO ID. Calculate the collision coefficient according the simulation results. We have repeated the experiments for 100 times and plot the average in Figure 3a.

Fig.3a shows that the collision coefficient is dropping while the dimension of ID space is increasing. It doesn't contradict with Fig.3b. This phenomenon is because with the increasing of dimension of ID space, the TSO ID is scattered in a vast ID space.

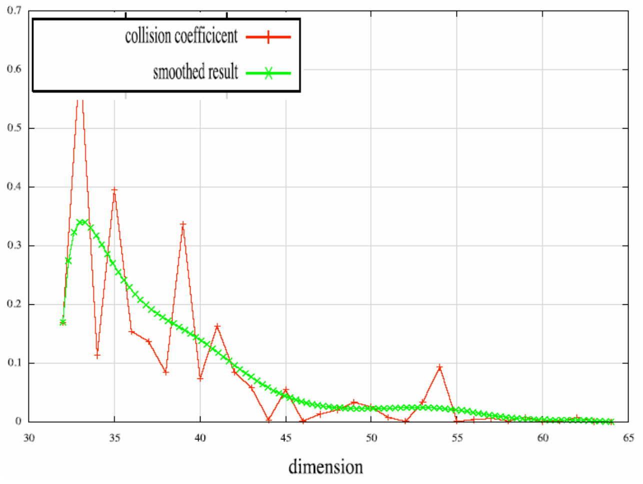

Figure 3a. The collision coefficient in experiments with different dimension of ID space.

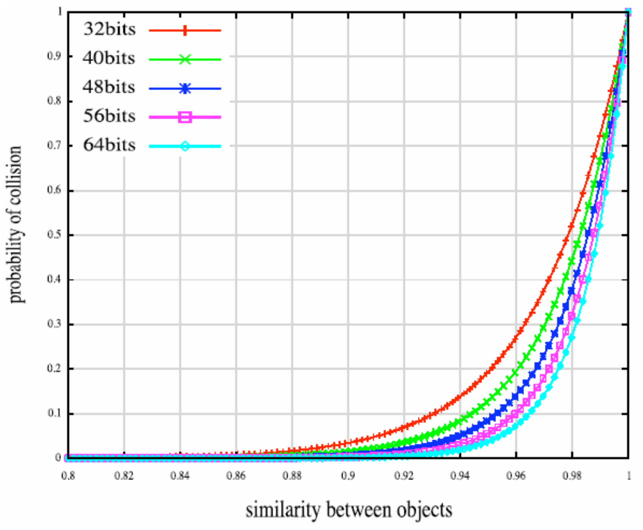

Figure 3b. The probability of hash collision at different vector dimensions

So with the limited number of samples, the collision rate drops down to a little value.

Now, a 20 dimensional vector space is used in Vivaldi and TSO ID generation. Fig.3a shows the statistics of the collision rate in our simulation.

This figure also shows that the overlay in TSO is related to the coordinate dimensions. If d is large, the collision probability will remain at a low level, and then the virtual node is not necessary. If d is small, the collision probability will remain at a high level. The TSO will be degenerated to be a ring composed by several virtual nodes like Chord, and each virtual node will be composed by lots of nodes. In order to reduce collision probability, the node can run the hash function several times to generate a unique ID.

3.5 Routing algorithm

As we can see from Fig.2, the nodes are grouped in the TSO overlay. The overlay network can be implemented in a two layer distributed hash table. The TSO ID generated by locality sensitive hash functions is virtual node ID.

Suppose the virtual node ID has *d* bits wide, each node has a *w* bits wide internal ID within each virtual node, we name it local ID. The ID of the node in TSO is composed of virtual node ID and local ID; the length of ID is w+d bits.

In TSO, the physical close nodes have been clustered into a local level P2P ring which is regarded as a virtual node in the high level Chord ring of the overall P2P overlay network. Routing tasks are first executed in local level rings, inside the virtual node, before they go up to higher level rings, a large portion of routing hops previously executed in the global P2P ring are now replaced by hops in local level rings, thus routing overheads can be reduced. In the local ring, the routing algorithm of routing message from node to node is the same as Chord protocol.

4. EXPERIMENTAL RESULTS

4.1 Experiment setup

In the experiments we simulate a network with 10000 nodes and measure the average latencies between nodes within the virtual node. We obtain the pair-wise RTTs between nodes by putting each node in a 150x150ms square randomly and use the distance in the square as the RTT between nodes. Each node obtains a network coordinates from Vivaldi and the network coordinates of each node in the experiment network are static. But in a real network, the node ID can be changed while its coordinates changed. In order to avoid the frequently changing of the node ID, a threshold can be used to limit the frequency of the changing of node ID triggered by the changing of coordinates.

The dataset used in the experiments contains the pair-wise RTTs between 10000 nodes. The dataset contains the RTTs among 10000 nodes.

We simulate a TSO overlay with different number of nodes, varying from 100 to 10000. In the simulation, message is generated at randomly selected node with randomly generated destination key. At each network size, the simulation is repeated *N* times, where *N* is the number of nodes in simulation.

4.2 The experiment results

Figure 4 shows the result of our simulation on the dataset which 10000 nodes. The figure shows the routing latencies of Chord routing protocol and TSO protocol with d = 20. The figure 4a shows that the routing latencies in TSO is smaller than the standard Chord protocol. The figure 4b shows that the TSO routing latencies at different d (12, 20 and 30). From this figure, we can see that with bigger d(30) or smaller d(12), the TSO has the same performance with Chord. This is because with bigger d, the probability of the close nodes will be in a same virtual node is low. Then the TSO act as a standard Chord. With a small d, most of nodes will be hashed in few virtual node, the most routing tasks are executed in few local level rings. And the virtual node can not distinguish the close nodes and the distant nodes. In our experiments, the appropriate value of d is 20.

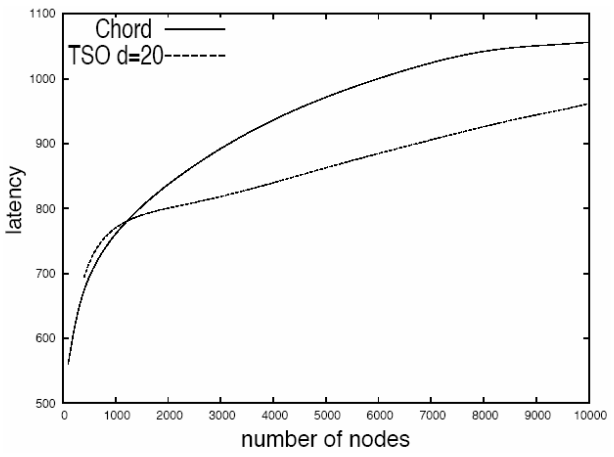

Figure 4a. Routing latencies in TSO and the standard Chord protocol

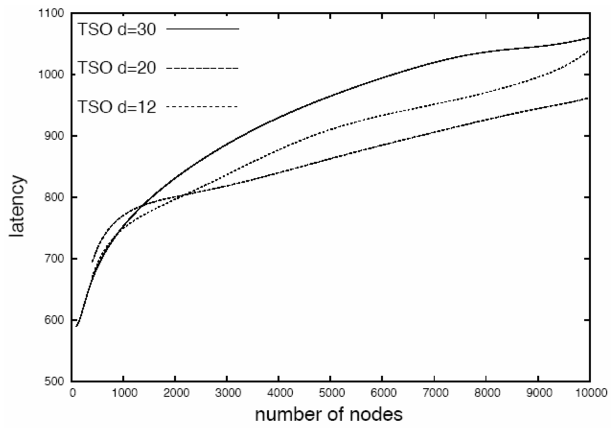

Figure 4b. Routing latencies in TSO at different d (12, 20 and 30)

5. RELATED WORKS

Studies in [3][10] attempt to cluster the nodes according to their IP address. But it is not always reasonable to use the distances between IP address strings to reveal the topology.

In [11], the topology matching is studied in unstructured network. In their work, every node tries to probe the distances between their neighbors. According to the RTT information, nodes take the nearest nodes as their neighbors and remove the faraway nodes from their routing table. The experiments show that it is efficient to reduce the routing latency in application layer. But this approach is not applicable to structured overlay.

Studies in [12] present a protocol to generate an unstructured overlay to achieve locality characteristic. They build a two level overlay with network proximity to decrease the communication cost between end hosts.

GIA[14] try to solve the problem in view of load balancing. In GIA, the powerful node (with more powerful processor, hard disk, memory, and network bandwidth) has higher degree so that it takes responsibility of more work. On the contrary, the less powerful nodes are clustered around the powerful nodes.

The main difference between TSO is different to other approaches is that we propose a construction of a topology-aware two layer structured overlay network based on locality sensitive hashing scheme. The other structured peer-to-peer overlay, such as Chord, can optimize the routing by proximity aware route selection, which is different to our solution. Our proposed system can be treated as proximity-aware node ID assignment in overlay network.

6. CONCLUSIONS

In this paper, we propose a new mechanism to construct a topology-aware structured overlay network based on locality sensitive hashing scheme. In TSO, the locality sensitive hashing scheme was proposed to alleviate the mismatching problem. The two layer architecture of TSO will reduce the system routing overhead efficiently. And the simulation shows that the performance of our overlay improves significantly comparing with the one in traditional structured overlays.

In the future, we will continue to investigate the topology matching in overlay network and do more experiments in varied network size and under high churn situation.

7. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant No. 60673166 and 60433040.

8. REFERENCES

[1] Xin Yan Zhang, Qian Zhang, Zhensheng Zhang, Gang Song, and Wenwu Zhu, *A Construction of Locality-Aware Overlay Network: mOverlay and Its Performance*, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 22, NO. 1, JANUARY 2004.

[2] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*, Proc. ACM SIGCOMM, 2001.

[3] Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. *Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering*. In Proceedings of the Association for Computational Linguistics 43rd Annual Meeting, Jun 25-30, 2005.

[4] Charikar, Moses. *Similarity Estimation Techniques from Rounding Algorithms*. In Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002.

[5] T. S. E. Ng and H. Zhang. *Predicting Internet networking distance with coordinates-based approaches*. In Proceedings of IEEE INFOCOM, June 2002.

[6] T. S. E. Ng and H. Zhang. *Towards Global Network Positioning* , ACM SIGCOMM Internet Measurement Workshop, November,2001.

[7] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. *Vivaldi: A Decentralized Network Coordinate System*. In Proc. of the ACM SIGCOMM'04 Conference, 2004.

[8] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. *Practical Distributed Network Coordinates*. In Proceedings of the ACM HotNets Workshop, 2003.

[9] B. Krishnamurthy and J. Wang, *Topology Modeling via Cluster Graphs*, Proc. SIGCOMM Internet Measurement Workshop, 2001.

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

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

[12] D. Watts and S. Strogatz. *Collective dynamics of small-world networks*. Nature, 363:202–204, 1998.

[13] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*, Proc. ACM SIGCOMM, 2001.

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