Proximity-Aware DHT for Efficient Lookup Service in Peer-to-Peer Applications

Satoshi FUJITA
Department of Information Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527, Japan

***Abstract*—Distributed Hash Tables (DHTs) attract considerable attention as a way of realizing an efficient lookup service in Peer-to-Peer (P2P) applications. However, many of conventional DHTs such as Chord and Kademlia do not take into account the proximity of nodes in the physical network, which causes a significant performance degradation as the size of the underlying network grows. In this paper, we propose a method to construct a proximity-aware DHT based on the notion of virtual coordinate space realized by a network coordinate system (NCS). The performance of the proposed method is evaluated by simulation. The result of simulations indicates that it reduces the average physical distance between adjacent nodes in the P2P overlay by 10% for random or scale-free networks, and by 80% for grid-structured networks.**

Keywords: P2P, DHT, proximity-awareness.

I. Introduction

According to the recent advancement of network technologies, Peer-to-Peer (P2P) systems have attracted considerable attention as a way of realizing scalable network services such as instant messaging, video conferencing, and file sharing. Among several technologies supporting those P2P applications, **Distributed Hash Tables** (DHTs) is considered as a key technique to realize an efficient lookup service in a P2P environment. Overlay network based on the notion of DHT is referred to as P2P DHT [13], [15], [17]. In the following, we will refer to each node (i.e., host) participating in a P2P DHT as **peer**, and distinguish it from the other nodes existing in the (underlying) physical network.

Suppose that there are several shared resources in a P2P network. In P2P DHT, the set of "indexes" to such resources[1] are stored in a logical space called **hash table** in an autonomous manner, i.e., each index is mapped to a point in the table by using an appropriate hash function and is stored at a peer managing that point. The table is partitioned into several subtables and those subtables are associated to the participants so that an index mapped to a point in a subtable is managed by a peer associated with the subtable. The access to an index held in P2P DHT is conducted by forwarding a request message to a point corresponding to the index through the overlay network in a greedy manner. Hence, the access time is dominated by the routing time to the target point, which is proportional to the number of hops in the overlay network and the physical distance between two adjacent nodes in the overlay (a detailed explanation of P2P DHT will be given in the next section). In general, the average hop count to the target point in the overlay depends on the coordinate system adopted in the DHT; e.g., it is $O(n^{1/d})$ in the d-dimensional Cartesian coordinate space adopted in CAN [13], and it is $O(\log n)$ in the hypercubic coordinate space adopted in Chord [17], where n is the number of peers participating in the overlay. On the other hand, many of conventional P2P DHTs do not explicitly take into account the physical distance between adjacent peers in the overlay. In other words, conventional P2P DHTs do not avoid redundant message transmissions between distant peers.

In this paper, we consider the problem of constructing a proximity-aware P2P DHT which reflects the proximity of nodes in the physical network. Proximity of peers in P2P DHT has been extensively investigated in the literature [3], [5], [7], [9], [14], [16]. As will be described later, if we do not allow the reconfiguration of the overlay connecting existing peers, it is equivalent to the problem of selecting an appropriate location in the P2P DHT to which a newly arrived node should be inserted. A solution to this problem would be to simply select a closest peer to the new node as its adjacent peer, which is known as the nearest neighbor method in the literature. Unfortunately however, such a naive scheme does not work well since it merely tries to reduce the distance between adjacent peers in the overlay, and the average distance between arbitrary nodes could not be minimized in general.

[1]The index of a resource consists of the name, attribute, and the name and address of the peer who holds the contents of the resource.

978-0-7695-4477-9/11 $26.00 © 2011 IEEE
DOI 10.1109/CSE.2011.207

To overcome such a drawback of the conventional method, we will take another approach based on a logical space reflecting the proximity of nodes in the physical network. More concretely, we consider a logical coordinate space realized by the Vivaldi network coordinate system [2] and try to map points in the coordinate space to the hash table through an appropriate Affine transformation. The effect of the proposed method is evaluated by simulation. The result of simulations indicates that the proposed scheme reduces the average physical distance between adjacent nodes in the overlay by 10% for random physical networks, and by 80% for grid-structured physical networks.

The remainder of this paper is organized as follows. Section II describes an outline of the routing in P2P DHT. The proposed method is described in Section III. Section IV illustrates the result of simulations. Finally, Section V concludes the paper with future work.

II. P2P DHT

P2P DHT provides an efficient tool to manage and retrieve objects distributed over a network. In this section, we describe the routing in P2P DHT by using CAN as a concrete example. The design of CAN [13] is based on a virtual d-dimensional Cartesian coordinate space on d-torus. At any point in time, the entire coordinate space is dynamically partitioned among all peers in the system, in such a way that every peer *owns* its individual, distinct space within the entire space, called a **zone**. Each peer learns and maintains the IP addresses of the peers that own coordinate zones adjacent with its corresponding zone, which serves as a coordinate routing table to realize a routing between arbitrary points in the coordinate space (similar idea is used in P2P DHTs based on different coordinate space, such as hypercubic space adopted in Chord [17] and Kademlia).

Consider a resource held by a node in the system. To realize an efficient management of the resource, CAN stores a key-value pair (K_1, V_1) to the virtual coordinate space, where K_1 is the name of the resource and V_1 is the name of a peer who holds the contents of the resource. More concretely, the scheme first maps key K_1 onto a point in the d-dimensional coordinate space by using d uniform hash functions. It then stores the corresponding key-value pair at the peer that owns the zone within which the point lies, which will be referred to as the *target* peer, in what follows. The retrieval of a stored key-value pair could be done in a similar manner. If the calculated target point is owned by the requesting peer or an immediate neighbor of him, the value associated to the given key can easily be obtained by using a local communication. However, if it is not the case, the request must be *routed* through the CAN until it reaches the peer who owns the zone containing the target point, and such a routing is easily realized by using routing tables owned by the individual peer (the easiest way is to adopt a greedy routing, but we could extend it to be fault-tolerant).

In CAN, the entire coordinate space is divided into zones, and those zones are assigned to peers in a one-to-one manner. Thus, to allow CAN to grow incrementally, a new node joining the system must be assigned its own portion of the coordinate space, by splitting a zone owned by an existing peer to several subzones and by taking over one of them to the new node. More concretely, such an assignment is conducted as follows:

procedure JOIN

1) First, the new node u locates a peer v by using an appropriate locating mechanism, and sends a join message to v.
2) Upon receiving the message, peer v splits his zone into two halves by using an appropriate coordinate, and sends back one half to node u.
3) After receiving it, u becomes the owner of the zone received from v, and notifies the fact to all neighbors to keep the consistency of the routing table.

In the method shown in the original paper [13], the selection of peer v in the first step is conducted in a random manner, which were later extended to take into account the load balancing of the peers in terms of the number of inquiries received from the other peers [19].

III. Proposed Method

A. Overview

A natural way to reduce the communication time in P2P DHT is to reduce the physical distance between two peers adjacent in the overlay. In this paper, we attempt to reduce such distance in CAN. To this end, we will take an approach to modify the procedure for inserting a new node to the overlay in such a way that the average physical distance between adjacent peers will be minimized.

A key idea of the proposed method is to use a network coordinates system called Vivaldi [2], to navigate such a proximity-aware assignment of zones to newly arrived peers. More concretely, in the proposed method, it first approximates the distance between peers in a virtual m-dimensional coordinate space by using Vivaldi, and

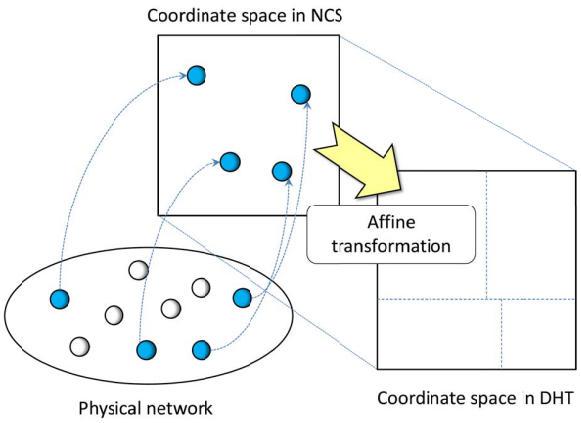


Fig. 1. Overview of the proposed scheme.

then maps the resultant points to the d-dimensional coordinate space of CAN by using an appropriate Affine transformation. See Figure 1 for illustration.

B. NCS

As a tool to estimate the distance between peers in a physical network, NCS (Network Coordinates System) has attracted considerable attention in recent years [4], [11], [12], [6], [8], [1], [18]. The objective of NCS is to accurately estimate the RTT (round trip time) between arbitrary two peers in the network, by each peer measuring RTT values to a limited number of sample peers. More concretely, it tries to assign a coordinate point in the virtual space to each peer, in such a way that the distance between two peers in the coordinate space is (almost) proportional to the actual RTT value between them.

GNP [10] and Vivaldi [2] are typical NCSs proposed in the literature. In GNP (Global Network Positioning), an assignment of coordinate point to a new peer is realized by measuring RTT value to several *landmarks* with a given coordinate point, and by selecting a point in the space that minimizes the error to the measured values as the estimated coordinate point. On the other hand, Vivaldi relies on no landmark peers, in contrast to GNP. In Vivaldi, each peer measures RTT values to several randomly selected sample peers, and then tries to converge the resultant points by repeating a compensation based on a "spring" model.

In Vivaldi, each peer i keeps two variables x_i and e_i representing the **coordinate point** and its **local error**, respectively, and it repeatedly updates those variables by using measured RTT values. More concretely, peer i acquires a pair of variables $\langle x_j, e_j \rangle$ from peer j when it measures RTT value r_{ij} between i and j, and if $\parallel \vec{x}_i - \vec{x}_j \parallel \neq r_{ij}$, then peer i updates its coordinate point so that the difference between $\parallel \vec{x}_i - \vec{x}_j \parallel$ and r_{ij} will be minimized, where $\vec{x}_i$ denotes a vector from the origin to point x_i. Let $\vec{u}_{ji}$ denote the unit vector from point x_j to point x_i, which is defined as a random unit vector if $x_i = x_j$. Then, a concrete procedure for updating variables x_i and e_i is described as follows:

Step 1: Let w be the weight of e_i defined as $w \stackrel{\text{def}}{=} e_i/(e_i + e_j)$, and let e_s be the relative error defined as $e_s \stackrel{\text{def}}{=} |\parallel x_i - x_j \parallel - r_{ij}|/r_{ij}$.

Step 2: Update e_i as $e_i \leftarrow e_s \times w + e_i(1 - w)$.

Step 3: Update x_i such that $\vec{x}_i \leftarrow \vec{x}_i + c_c \times w \times (r_{ij} - \parallel \vec{x}_i - \vec{x}_j \parallel) \times \vec{u}_{ji}$, where c_c is a tuning parameter less than one.

C. Basic Procedure

In the following exposition, for simplicity, we assume that both CAN and Vivaldi use a two-dimensional Cartesian coordinate as the underlying coordinate space.

Let i be a new node who wants to insert itself to the CAN. In the proposed scheme, node i first communicates with several peers j in the CAN to acquire the RTT value r_{ij} to those peers, and pairs of coordinate points $\langle p_j, x_j \rangle$, where p_j is the center of the zone that is assigned to j, and x_j is the Vivaldi coordinate point of peer j. (At the same time, it executes the Vivaldi algorithm described in the last subsection to compensate the coordinate points of peers i and j.)

It then calculates the center p_i of a zone that should be assigned to peer i in CAN, in the following manner: 1) Peer i first estimates the coefficients of an Affine transformation that approximately maps points in the Vivaldi coordinate to the corresponding points in the CAN coordinate, by using sample pairs of the coordinate points (a detailed way for the estimation is described in the next subsection). 2) It then transforms point x_i in the Vivaldi coordinate to a point p_i in the CAN coordinate by using the resultant Affine transformation, where if the calculated point is in the outside of the CAN coordinate, then a point on the boundary is selected as the target point.

D. Estimation of Affine Transformation

Suppose that a point $x = (x_1, x_2)$ in the Vivaldi coordinate is mapped to a point $p = (p_1, p_2)$ in the CAN coordinate by an Affine transformation. Then, by using five constants $\lambda_1, \lambda_2, \mu_1, \mu_2$, and θ, such an Affine

transformation is represented as follows:

$$\begin{cases} p_1 &= \lambda_1 x_1 \cos\theta - \lambda_2 x_2 \sin\theta + \mu_1 \\ p_2 &= \lambda_1 x_1 \sin\theta + \lambda_2 x_2 \cos\theta + \mu_2. \end{cases}$$

Recall that in the proposed scheme, a new node communicates with several peers before calculating the CAN coordinate to which it is associated. We call such peers **probes** for the calculation, and use symbol k to denote the number of probes. By definition, the value of five coefficients of an Affine transformation can be uniquely determined by giving two pairs of coordinate points (if they are independent). Thus, after receiving k probes, it can derive (at most) ${}_kC_2$ sets of coefficients. Those sets are identical if and only if there is an Affine transformation that exactly matches the mapping defined by those k probes. However, even if there is no such unique Affine transformation, we can approximate the mapping in the following manner: 1) for each coefficient, it considers a set of ${}_kC_2$ values, and 2) it takes the median of them as the coefficient of the estimated Affine transformation.

IV. Simulation

The performance of the proposed scheme is evaluated by simulation. In the simulation, we realize an RTT between two peers by inserting a delay component between them; i.e., we do not take into account the queuing delay and traffic congestions. In addition, we assume that each peer continuously executes the Vivaldi algorithm as the background job, and repeats an update of its local variables by communicating with eight random peers per second. In the simulation, we fix the configuration of the physical network in advance. Each node in the physical network joins the CAN according to a Poisson distribution with mean 60 sec, and leaves the CAN according to an exponential distribution with mean 300 sec. In addition, each peer participating in the CAN issues a request for retrieving an index according to a Poisson distribution with mean 60 sec, where the target point in the CAN coordinate is randomly selected with a uniform probability.

In the following, we will evaluate the performance of peer insertion schemes in terms of the following three metrics: 1) hop count to the target peer in the overlay, 2) hop count to the target peer in the physical network, and 3) the **stretch** of the overlay which is defined as the ratio of the hop count in the physical network to the hop count in the overlay.

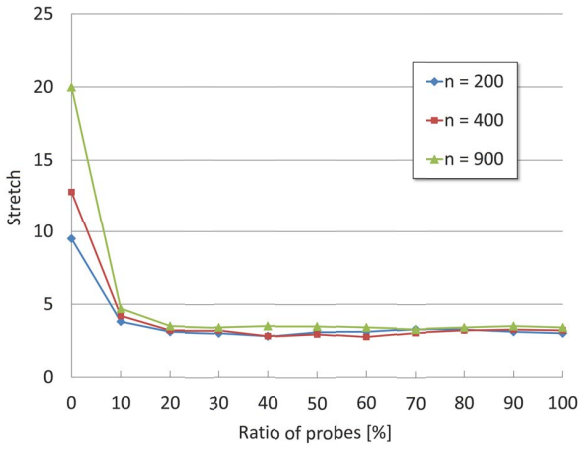


Fig. 2. Average stretch for grid-structured physical network.

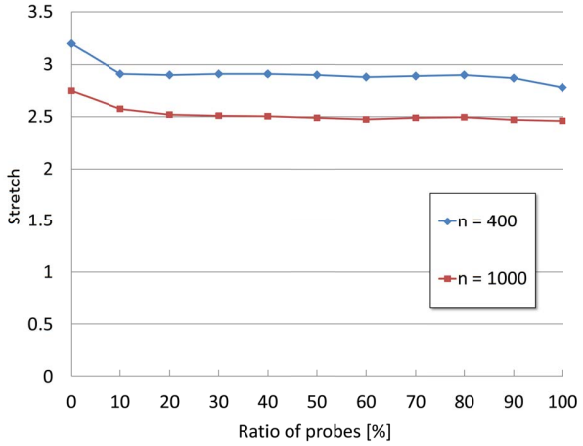


Fig. 3. Average stretch for scale-free networks.

A. Influence of the Topology of Physical Network

At first, we evaluate the stretch of the overlay by fixing the structure of the underlying physical network to a two-dimensional grid (note that since we are assuming that the topology of the P2P DHT is a two-dimensional grid, it is expected that the proposed scheme exhibits the best performance in such a specific situation). The result is shown in Figure 2. The horizontal axis is the number of probes used in the proposed scheme (e.g., "30%" indicates that the 30% of the peers in the P2P are selected as probes), and three curves in the figure correspond to the stretch for a grid of size 200, 400, and 900, respectively. The reader should note that the case of "zero probes" coincides with a conventional scheme based on a randomized peer selection. As shown in the figure, the average stretch decreases as increasing the number of probes. In particular, it reduces to one

TABLE I
RANDOMIZED METHOD.

Hop count (overlay)	Hop count (physical layer)	Delay [sec]
8.50	113.31	1.19

fifth by increasing k from 0 to a specific constant such as 10 or 20. This result indicates that the proposed scheme certainly organizes an efficient overlay in a two-dimensional coordinate space when the underlying physical network has a similar (i.e., two-dimensional) structure.

We conducted similar experiments for other classes of graphs, including random graphs and scale-free graphs. Figure 3 summarizes a part of the results. By the figure, we can observe a similar phenomena to Figure 2, i.e., the average stretch monotonically decreases as increasing k, and it reduces to 90% by increasing k from 0 to n. In addition, we can see that the performance degradation for small k's is relatively small. For example, even if k is fixed to 10% of the total number of nodes in the system, we could realize an improvement of the average stretch by 9% compared with the case of $k = 0$.

B. Comparison with Nearest Neighbor Method

In the nearest neighbor method (NN, for short), a node which newly becomes a member of the overlay selects the closest peer in the physical network as the adjacent peer in the overlay network. More concretely, it randomly selects k probe peers from the set of peers in the overlay network, and selects a closest peer among those probes with respect to the RTT values. It then asks the selected peer to hand over a half of the zone managed by him. In this section, we compare the performance of the proposed scheme with such NN in terms of the hop counts in the physical network, by assuming that the target point of each request in the CAN coordinate is randomly selected with a uniform probability. As the underlying physical network, we use a grid structure of size 20×20, where RTT of each link is fixed to 20 ms. The simulation time is fixed to five hours. Note that in the current setting of parameters, the average number of participant peers ranges from 320 to 340.

Figure 4 illustrates the relationship between k and the average hop count in the physical network. For comparison, we show the result for the randomized method used in the original CAN in Table I (note that it does not depend on the number of probes). By the figure, we

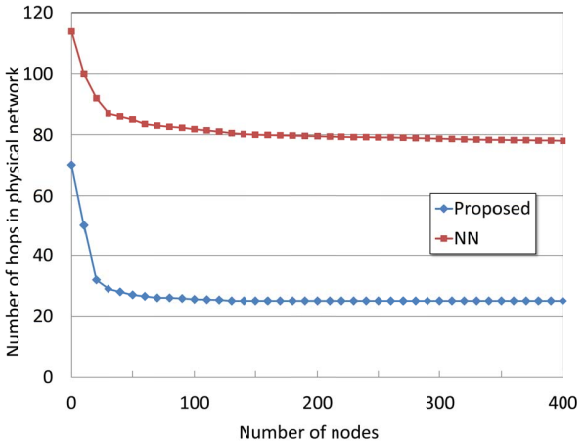


Fig. 4. Average hop counts in the physical network.

can observe that the average hop count of the proposed method is about 30% of NN, which is apparently due to the effect of a global optimization adopted in the proposed method. Another observation is about the effect of the number of probes; i.e., in both schemes, the average hop count monotonically decreases as increasing probe number k. For example, by increasing k from zero to 400, the hop count in NN reduces to 70% and that in the proposed method reduces to 20%.

C. Convergence of Affine Transformation

Figure 5 illustrates the transition of the points mapped to the CAN coordinate during the simulation time described in the last subsection (i.e., physical network is a two-dimensional grid of size 20×20), where a large square in the figure represents the boundary of the CAN coordinate. As shown in the figure, at an early stage of the construction of P2P DHT, the mapped points are not regularly arranged as a two-dimensional grid in the CAN coordinate. However, by repeating the arrival and departure of peers, it converges to a regular structure as shown in the right hand side of the figure.

The above interesting phenomena could be explained as follows. First, it should be noted that the coordinate point obtained by the Affine transformation is different from the point to which the new node is actually assigned in the CAN coordinate (i.e., the center of a zone to which it is associated). Such a difference will be increased as decreasing the regularity of the zone distribution. In other words, in the proposed scheme, each of the newly arrived nodes automatically compensates the mapping by using the center of zones that is actually assigned to peers instead of the point obtained by applying the

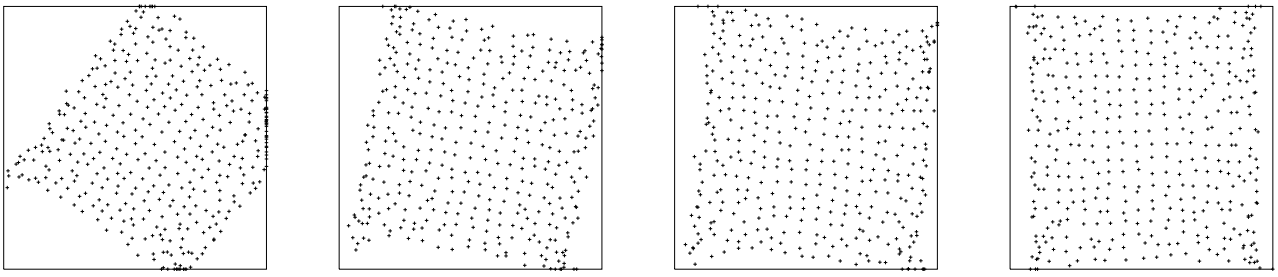
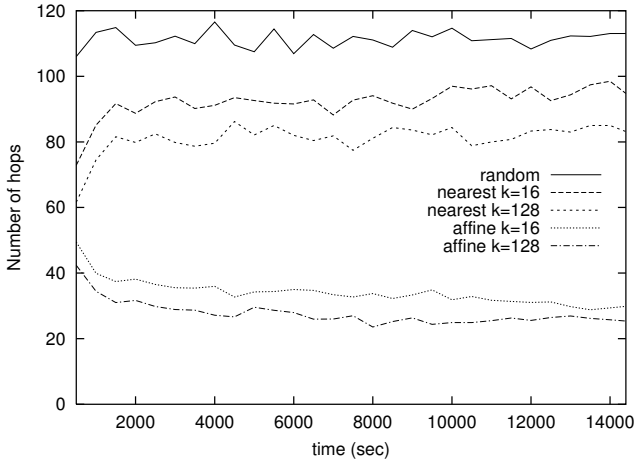
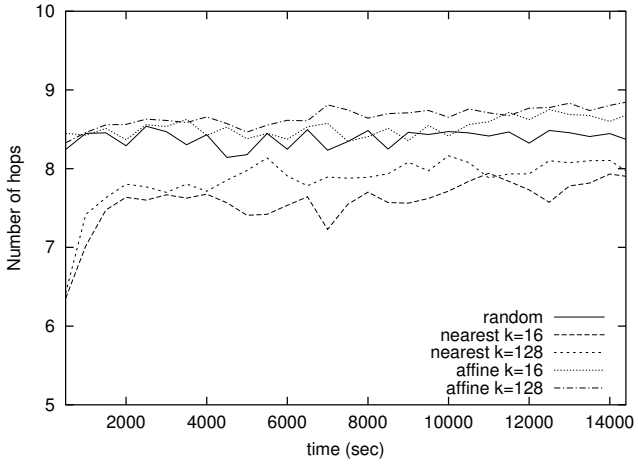


Fig. 5. Compensation of the Affine transformation.



(a) Physical network.



(b) Overlay network.

Fig. 6. Transition of average hop count.

current Affine transformation. Such a compensation will be repeated as repeating the arrival and departure of peers, and it eventually causes to a regular arrangement illustrated in the right hand side of the figure.

In order to verify the above conjecture, we trace the transition of the hop count during the simulation, i.e., we compare the average hop count of three schemes (i.e., random selection in the original CAN, NN, and the proposed method) for every 500 sec in the simulation. The number of probes is set to either 16 or 128. Figure 6 illustrates the results, where (a) and (b) represents the hop count in the physical and the overlay networks, respectively.

From Figure 6 (a), we can observe that in the proposed method, the hop count in the physical network monotonically decreases until 2000 sec, and particularly when $k = 16$, a moderate reduction still continues after that time period. In contrast, such a reduction cannot be observed for the other two schemes although the average hop count slightly changes during the simulation time. In addition, Figure 6 (b) indicates that the hop count of NN in the overlay decreases until 2000 sec, which is because of an imbalance of zone partitioning based on NN. Although a similar observation holds for the reduction of the hop count in the physical network in the NN method, we could observe that it is different from the behavior of the proposed method.

V. Concluding Remarks

This paper proposed a proximity-aware P2P DHT based on the Vivaldi network coordinate system. The simulation result indicates that the proposed scheme actually improves the proximity-awareness of the resultant overlay, and it was clarified that the behavior of the proposed scheme is completely different from conventional schemes. Our future work is as follows. First, since the convergence speed of the Affine transformation in the proposed method is very slow, we have to develop a new technique to accelerate it. The second issue is the refinement of the definition of proximity; i.e., we should consider other metrics such as the communication

bandwidth and the load of peers, in addition to RTT adopted in the current paper.

REFERENCES

[1] M. Costa, M. Castro, R. Rowstron, and P. Key. "PIC: Practical Internet coordinates for distance estimation," In *Proc. 24th International Conference on Distributed Computing Systems*, pp.178-187, 2004.

[2] F. Dabek, R. Cox, F. Kaashoek and R. Morris. "Vivaldi: A Decentralized Network Coordinate System," In *Proc. ACM SIGCOMM*, Portland, OR, Aug. 2004.

[3] N. B. Dang, S. T. Vu, H. S. Nguyen. "Building a Low-latency, Proximity-aware DHT-Based P2P Network," In *Proc. the 1st International Conference on Knowledge and Systems Engineering (KSE 2009)*, pp.195-200, 2009.

[4] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gruniewicz, and Y. Jin. "An architecture for a global Internet host distance estimator service," In *Proc. IEEE INFOCOM*, pp.210–217, 1999.

[5] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, I. Stoica. "The Impact of DHT Routing Geometry on Resilience and Proximity," In *Proc. the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*, pp.381-394, 2003.

[6] L. Lehman and S. Lerman. "PCoord: Network position estimation using peer-to-peer measurements," In *Proc. IEEE International Symposium on Network Computing and Applications (NCA)*, August, 2004.

[7] Z. Li, Z. Zhu, G. Xie, Z. Li. "Fast and proximity-aware multi-source overlay multicast under heterogeneous environment," *Computer Communications*, 32(2): 257-267, 2009.

[8] H. Lim, J. C. Hou, and C.-H. Choi. "Constructing Internet coordinate system based on delay measurement," *IEEE/ACM Transactions on Networking,* 13(3): 513–525, June 2005.

[9] S. Morimoto, F. Teraoka. "CHOP6: A DHT Routing Mechanism Considering Proximity," In *Proc. International Symposium on Applications and the Internet Workshops (SAINTW'07)*, 2007.

[10] T. Ng and H. Zhang. "Predicting Internet Network Distance with Coordinates-Based Approaches," In *Proc. IEEE INFOCOM*, pp.170–179, 2002.

[11] T. Ng and H. Zhang. "A network positioning system for the Internet," In *Proc. USENIX Annual Technical Conference*, June, 2004.

[12] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. "Lighthouses for scalable distributed location," In *Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, pp.278-291, 2003.

[13] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. "A Scalable Content-Addressable Network," In *Proc. ACM SIGCOMM*, Aug. 2001.

[14] L. Ribe-Baumann, K.-U. Sattler. "A small-world DHT built on generalized network coordinates," In *Proc. the 2010 EDBT/ICDT Workshops (EDBT '10)*, 2010.

[15] A. Rowstron and P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems," In *Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp.329–350, 2001.

[16] D. Smith, N.-F. Tzeng. "Proximity-Aware Distributed Mutual Exclusion for Effective Peer-to-Peer Replica Management," In *Proc. the Eighth IEEE International Symposium on Network Computing and Applications*, pp.36-43, 2009.

[17] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking*, 11(1): 17–32, 2003.

[18] M. Szymaniak, D. Presotto, G. Pierre, and M. van Steen. "Practical large-scale latency estimation," *Computer Networks*, 52(7): 1343–1364, May, 2008.

[19] D. Takemoto, S. Tagashira, S. Fujita. Distributed Algorithms for Balanced Zone Partitioning in Content-Addressable Networks, *Proc. 10th ICPADS*, Newport Beach, pp.377–384, 2004.