IPSJ Online Transactions Vol. 4 57–72 (Mar. 2011)

Regular Paper

Practical Approach to Integrating Network Coordinates with Distributed Hash Tables

Toshinori Kojima,[†1] Masato Asahara,[†1]
Kenji Kono[†1] and Ai Hayakawa[†1]

Network coordinates (NCs) enable the efficient and accurate estimation of network latency by mapping the geographical relationship among all nodes to Euclidean space. Many researchers have proposed NC-based strategies to reduce the lookup latency of distributed hash tables (DHTs). However, these strategies are limited in the improvement of the lookup latency; the nearest node to which a query should be forwarded is not always included in the consideration scope of a node. This is because conventional latency improvement strategies assign node IDs independent of the underlying physical network and still have the possibility of detour routing. In this paper, we propose an NC-based method of constructing a topology-aware DHT by Proximity Identifier Selection (PIS/NC). PIS/NC constructs a logical ID space of a DHT from the Euclidean space constructed by NCs; a node ID corresponds to the network coordinate of the node. By doing this, the consideration scope of a node always contains the nearest node, thus, we can expect a great reduction in lookup latency. Unlike the conventional PIS strategy that poses unavoidable issues due to uneven ID distribution, PIS/NC tries to moderate these issues by a simple optimization, provided by a PIS/NC stabilizer. The PIS/NC stabilizer detects an uneven distribution of node IDs locally, and then recalculates some IDs so that the unevenness is moderated. As case studies, this paper presents Canary and Harpsichord, which are PIS/NC-based CAN and Chord, respectively. Simulation results show that PIS/NC-based DHTs improve lookup latency. Under the environment using the Transit-Stub model, where SAT-Match and DHash++ are only able to reduce the median lookup latency by 19% of CAN and 9% of Chord, respectively, Canary and Harpsichord reduce it by 40% and 35%, respectively. We also verify that the PIS/NC stabilizer moderates the non-uniform distribution of node IDs.

1. Introduction

Network coordinates (NCs) [1)–8)] are an emerging technology that allow us to estimate network latency on the Internet without explicitly communicating with target nodes. NCs map the geographical relationship among all nodes participating in the system on a single coordinate space by assigning virtual d-dimensional coordinates to each node. The Euclidean distance between any pair of coordinates approximates the network latency between the corresponding nodes. Thus, each node estimates network latency to other nodes by calculating the coordinate distance if the communicating node knows the coordinate of the target node. NCs work well on the real Internet. In fact, Azureus BitTorrent client [9)], whose network runs a million nodes, implements NCs as a plug-in. The usefulness of NCs is demonstrated in an experiment which uses 10,000 nodes of the real Azureus network [10)].

There is still a need for more research on how to apply NCs to peer-to-peer (P2P) applications, such as distributed hash tables (DHTs) [11)–17)]. Although some studies [18)–20)] improve the lookup efficiency of DHTs using NCs, they do not make maximum use of NCs. Both PNS and PRS are limited in the improvement of the lookup latency as long as node IDs are assigned randomly. In these studies, NCs are used for *Proximity Neighbor Selection (PNS)* or *Proximity Route Selection (PRS)* [21)]. For example, DHash++ [18)] (based on Chord [16)]) uses NCs for PNS, a strategy in which each node chooses closer ones among some candidates when building its routing table. Rhea, et al. combine Bamboo [14)] (based on Pastry [15)]) and NCs for PRS [20)], a strategy in which each node considers network distance when it chooses a forwarding node from the candidates. Even if there is a node very close to the destination on the physical network, it may not be included in the forwarding candidates of the source. (The details are described in Section 2.) As a result, the query detours to the destination node on the physical network.

In this paper, we explore the possibility of using NCs for *Proximity Identifier Selection (PIS)* [21)]. PIS is a fundamentally different strategy from PNS and PRS. The essence of PIS is that the nodes which are near on the underlay network are close to each other on the overlay network, too. To turn traditional DHTs into PIS-based ones, we take a novel approach called *Proximity Identifier Selection with NCs (PIS/NC)* which embeds NCs directly into the *heart* of the DHT structure. In other words, each node ID is calculated from each node's

†1 Department of Information and Computer Science, Keio University

NC so that the resulting logical ID space of DHT approximates that of NC. By doing this, nearby nodes in the NC space also become close to each other in the DHT space. On such an ID space, DHTs naturally forward queries based on the topology of the physical network.

To apply PIS to popular DHTs, PIS/NS improves three drawbacks of conventional PIS approaches. First, since conventional PIS does not reduce lookup latency very much, PIS/NS improves it more than the conventional PIS. Second, PIS/NS avoids key/value transfers caused by the adjustment of node IDs. Finally, PIS/NS uses a *stabilizer* that mitigates non-uniform ID distribution in conventional PIS.

To demonstrate the efficiency of PIS/NC, we designed two PIS/NC-based DHTs, *Canary* and *Harpsichord*. *Canary* combines CAN [13] and Vivaldi [2] (one of the popular NCs) and *Harpsichord* combines Chord [16] and Vivaldi. Obviously, PIS/NC is applicable to the DHTs using the d-dimensional Euclidean space as their ID space, like CAN, because NCs approximate the latency by the Euclidean distance. On the other hand, it seems difficult to apply PIS/NC to the other type of DHTs using non d-dimensional Euclidean space such as Chord. However, simple techniques enable us to apply PIS/NC to non-Euclidean-space-based DHTs. To map the d-dimensional space of NCs to the ring-shaped ID space of Chord, Harpsichord constructs a space-filling curve on the space of NCs. Then, the PIS/NC stabilizer moderates the non-uniform distribution of node IDs assigned by a space-filling curve. As a result, Harpsichord reduces the lookup latency as well as Canary.

Simulation results demonstrate that Canary and Harpsichord improve the lookup latency versus existing DHTs. In a synthetic environment, the lookup latency of Canary and Harpsichord are the lowest among typical PNS- or PRS-based DHTs. Compared to CAN with SAT-Match, which is one of the conventional PIS strategies, and DHash++, which is a PNS-based Chord, the reduction ratios of Canary and Harpsichord are larger by 21 and 26 points, respectively. To evaluate Canary and Harpsichord in a more realistic environment, we used the dataset derived from PlanetLab nodes [22]. In this environment, Canary and Harpsichord show the lowest latency as well. Compared to SAT-Match and DHash++, the reduction ratios of Canary and Harpsichord are larger by 20 and 12 points. We also show that a problem that PIS strategies have, i.e., the non-uniform distribution of node IDs, can be moderated by a PIS/NC stabilizer in these experiments.

The rest of the paper is organized as follows. Section 2 clarifies the design challenges and describes related work. We explain our approach, PIS/NC, and discuss the issues of PIS/NC in Section 3. Section 4 describes the details of Canary and Harpsichord. We show the simulation results in Section 5, and Section 6 concludes the paper.

2. Design Challenges and Related Work

2.1 Design Challenges

Many researchers have studied strategies to improve the lookup latency of DHTs. Gummadi, et al. analyzed the effect of the overlay geometries used in various DHTs and classified the strategies for lookup latency improvement with underlay information into three categories [21]. According to them, most of the strategies are categorized into two types, *PNS* or *PRS*.

- *Proximity Neighbor Selection* (PNS) chooses the nodes to be kept in a node's routing table according to the metrics based on the underlay information.
- *Proximity Route Selection* (PRS) considers the underlay metrics when a node chooses a forwarding node from its routing table to send a query to a particular destination.

Although PNS and PRS improve lookup latency to a certain level, unfortunately, they have inevitable limitations of the improvement of lookup latency. In PNS- or PRS-based DHTs, the nearest node to which a query should be forwarded is not always included in the consideration scope of a node. **Figures 1** and **2** show an example of such a case. Figure 1 illustrates a physical network topology which consists of five nodes. Note that node C is a newcomer node and its ID is 7, which is assigned randomly. Figure 2 illustrates a ring-shaped DHT overlay like Chord [16] constructed on the physical network topology in Fig. 1. When node A looks up C's key 7, the destination of the query is C. Due to A's routing table, node A has to forward the query to node Q, which is a high-latency node from A. Although PNS eventually adjusts A's routing table to forward a query whose destination is in the key range $[7, 8)$ to node C, completing the pro-

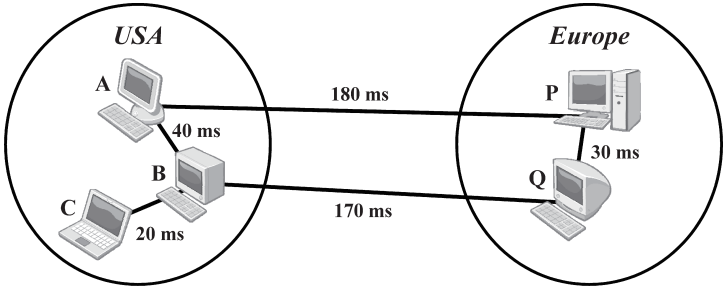


Fig. 1 An example of the underlay network which consists of five nodes.

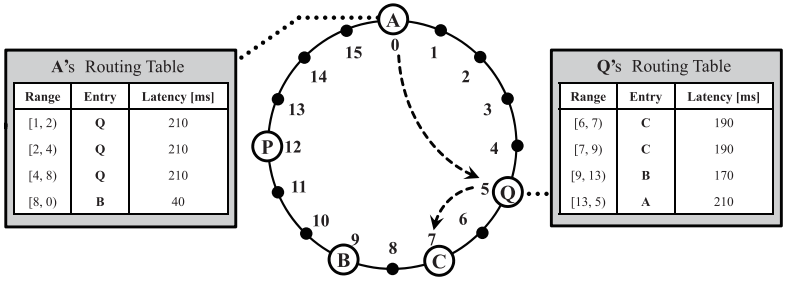


Fig. 2 An example of the cases with which PNS and PRS strategies cannot deal.

cess takes some time due to the delay for the advertising C's arrival. Meanwhile, PRS cannot improve the lookup latency due to the routing protocol of the DHT.

To clear up the improvement limitations of the lookup latency, we focus on another strategy, *PIS*.

- *Proximity Identifier Selection* (PIS) chooses nodes' identifiers so that they reflect their positions on the underlay.

PIS constructs a DHT whose overlay topology is correlated to the underlay topology. In PIS-based DHTs, if a node is close to the destination on the overlay, it is also close to the destination on the underlay. Therefore, it can improve the lookup latency more than PNS and PRS. **Figure 3** shows an example of PIS-based DHTs on the physical network in Fig. 1. The topological relationship on a PIS-based DHT space is based on the physical network topology. Since the

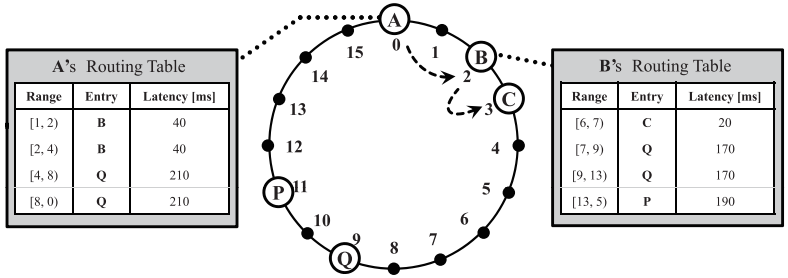


Fig. 3 An example of PIS-based DHTs.

Table 1 Qualitative comparison of design alternatives for topology-aware DHT.

	Topology-awareness	Key/value transfers	Load balance	Fault resilience	ID distribution
Original DHT	×	no	√	√	uniform
PNS & PRS	low	no	√	√	uniform
Conventional PIS	low	large	×	×	non-uniform
PIS/NC	√	negligible	moderate	moderate	moderately-uniform

nodes in USA, i.e., A, B, and C, are located close to each other on the overlay network, A's query for C's key 3 is forwarded within USA and the lookup latency is reduced significantly compared to the case illustrated in Fig. 2.

As many researchers have stated, there are several challenges posed by conventional PIS approaches. **Table 1** compares the properties between PIS/NC and conventional approaches. There are three important drawbacks of PIS/NC. First, PIS/NC should improve lookup latency more than conventional PIS approaches. Although PIS has the possibility to reduce lookup latency as much as possible, conventional PIS approaches have a limitation in improving latency. This is because conventional PIS uses an ad-hoc way or simple clustering to approximate the distance among nodes on the physical network.

Second, we need a solid strategy to generate a topology-aware identifier for each node. Conventional PIS-based DHTs use an ad-hoc method to assign a topology-aware ID to a node. For example, with SAT-Match [23], a node periodically recalculates its ID and repeats to leave from and join in the DHT until the

latency to nearby nodes is lowered enough. Such an ad-hoc strategy causes many key/value transfers between nodes.

Finally, we have to make the ID distribution of PIS-based DHTs uniform and as close to that of the original DHTs as possible. Conventional PIS makes dense regions of nodes on the DHT overlay because it naively assigns close IDs to nodes if the nodes are close to each other on the physical network. This unbalanced ID distribution makes DHTs less load-balanced and fault-resilient. Unbalanced ID distribution issue becomes more critical in non-Euclidean-space-based DHTs, e.g., ring-shaped ones. In such DHTs, there are few bypasses for forwarding a query because the overlay is one dimensional space. Therefore, many queries on such DHTs tend to pass through a node whose managing key region is larger than others and then, the node will become a hot spot, or at worst, a single point of failure.

In what follows in this section, we discuss the existing strategies by classifying them into three categories: PNS and PRS, conventional PIS, and the other strategies called multi-layered DHTs.

2.2 PNS and PRS

Pastry [15] is one of the traditional topology-aware DHTs. In Pastry, each node conducts PNS when it initializes and maintains its routing table. When a new node joins Pastry, it sends a "join" message with the key equal to the node's ID to the closest node on the physical network. Then, it initializes its routing table by obtaining the i-th row of its routing table from the i-th node encountered along the route from the first contact node to the destination one. Therefore, all routing table entries refer to a node that is near the present node on the physical network among all live nodes with a prefix appropriate for the entry. However, the improvement of lookup latency is limited due to the flexibility of selecting a candidate node for each row; a prefix of a candidate must be appropriate for the entry. Thus, an entry of a routing table will not always refer to the closest node.

In addition, a node in Pastry will forward a query on a detour path of the physical network. The entries in the i-th row of a routing table are close to the i-th node encountered along the route of the "join" message. However, they are not always close to the destination of a query. Thus, a node will forward a query to the high-latency node from the destination node of the query.

DHash++ [18] is a representative DHT for improving the lookup latency with NCs. DHash++, which is based on Chord [16], takes PNS with Vivaldi [2]. In Chord, the i-th finger table entry of the node with ID a refers to the first appearance node in the ID-space range from $a+2^i$ to $a+2^{i+1}$. On the other hand, in DHash++, it refers to the node with the lowest latency, estimated by Vivaldi coordinates, of up to the first x nodes in the i-th range. Average lookup latency per hop becomes lower than that of original Chord. However, the improvement of lookup latency is limited due to the limit of the number of candidate nodes selected for each finger table entry; an ID of a candidate for the i-th entry must be within the ID-space range $a+2^i$ to $a+2^{i+1}$. Thus, DHash++ will forward a query on a detour path, especially when it forwards to the lower row's entry of a finger table.

Rhea, et al. take PRS with NCs to improve the lookup efficiency of Bamboo [14], which is a customized Pastry to grow tolerance to churn [20]. It is true that PRS strategy reduces communicating latency per hop, but it may have to sacrifice the progress per hop on the logical space. As a result, it may take some extra hops for a query to reach the destination node. Moreover, in Bamboo with PRS, the candidate nodes will lessen as a query comes close to the destination. In addition, it still has the latency improvement limitations of Pastry described above.

Kaune, et al. take PNS and PRS to improve the lookup latency of Kademlia [12] using the metrics to estimate the underlay topology; the cluster underlay metric or NCs [19]. They modify the maintenance algorithm of buckets, which is the routing table of Kademlia, so that the nodes close to each other on the physical network are kept in the buckets of others. In Kademlia, the i-th bucket consists of k nodes whose XOR distance from the owner of the bucket is i. Thus, this solution has the same latency improvement limitations of DHash++ or Bamboo; while a PNS candidate of i-th bucket is limited to the nodes whose XOR distance is i, a PRS candidate is limited to k nodes of each bucket.

2.3 Conventional PIS

Ratnasamy, et al. proposed an overlay network of CAN which considers the underlying network with landmark binning scheme [24]. With this approach, each node is labeled according to the positional relationship between itself and several landmarks on the underlying network, and the same-labeled nodes are assigned

nearby zones to each other. Thus, the overlay network of CAN roughly approximates the physical network topology. However, the approximation accuracy of landmark binning is not so high because it is possible that far apart nodes are labeled the same bin. Moreover, this approach depends on the location of landmarks. For example, if landmarks are positioned densely in a narrow region, most of the nodes can be labeled the same bin.

SAT-Match [23] is an ad-hoc strategy for matching overlay with underlying networks. SAT-Match brings the overlay network close to the physical network by repeatedly measuring the latency between nearby nodes. In SAT-Match, each participating node periodically communicates with nodes which are reachable within small k hops (TTL-k neighborhoods) and measures the latency from itself to them. If there is a node which is close enough to itself on the underlying network, it "jumps" to that node's domain. "Jump" means that a node leaves its domain, and then joins another domain. By repeating this, the overlay network of SAT-Match approximates the underlying network. However, since SAT-Match only considers the physical network topology in a limited scope on the overlay network, there is a limitation for the effect. SAT-Match also increases key/value transfers during its operation.

eQuus [25] is a PIS-based, locality-aware and ring-shaped DHT. eQuus improves the lookup latency by clustering nearby nodes in the same "clique" in which node IDs are the same to each other. Since eQuus does not consider the clique locations on the physical network, it cannot always decrease the lookup latency, especially when a query is relayed on more than one cliques.

2.4 Multi-layered DHTs

Multi-layered DHTs reduce the lookup latency by decreasing the number of forwarding hops of a query. They decrease the number of forwarding hops so that the average latency of each forwarding hop does not increase. To achieve this requirement, they build some lower density ID spaces by clustering nodes. eCAN [26], which is based on CAN, builds hierarchical CAN spaces which the node clustering granularity is different from other layers. The coarser the clustering granularity is, the more queries reach the destination by a few of hops on the overlay. However, multi-layered DHTs do not always produce a notable effect. For example, they negligibly reduce the lookup latency of a query to a nearby one.

Coral [27] builds a multi-layered Chord or Kademlia by clustering the nodes close to each other on the physical network. The diameter of a cluster is the maximum desired round-trip time between any two nodes it contains. By using low-latency-level clusters as possible, Coral allows a query to reach the destination node with lower latency. Since Coral is one of the multi-layered DHTs, it also has the latency improvement limitations like eCAN.

3. Proximity Identifier Selection with NCs

3.1 Overview

To clear up the improvement limitations of the lookup latency, we propose *Proximity Identifier Selection with Network Coordinates* (PIS/NC). PIS/NC evolves the traditional DHTs into PIS-based ones by embedding NCs *directly* into their structures. In PIS/NC, a logical ID space of DHT is constructed from network coordinates so that nearby nodes on a physical network are assigned node IDs close to each other in the DHT logical space. PIS/NC greatly reduces the lookup latency compared with not only the non-topology-aware DHTs, but also the existing topology-aware ones.

PIS/NC is different from conventional PIS strategies such as SAT-Match; it constructs a logical ID space of a DHT based on the Euclidean space of NCs. PIS/NC determines a node ID on the basis of its network coordinate. By doing this, a network coordinate space is directly embedded into the *heart* of a logical ID space of a DHT. Then, the overlay network topology of a PIS/NC-based DHT reflects the geographical relationship of the nodes; low-latency communicable nodes are close to each other on the logical ID space of a DHT, while high-latency communicable ones are distant from each other on the space. Thus, the forwarding path of a query on a PIS/NC-based DHT is a good approximation of the path on the underlying network, i.e., the number of the query detouring on the physical network reduces significantly.

To apply PIS/NC to traditional DHTs, PIS/NC has a conversion method from a d-dimensional NC to a node ID on a DHT space. If the dimensionality of a DHT space equals that of a NC space, PIS/NC straightforwardly converts the NCs to the node IDs. If we want to use PIS/NC for other types of DHTs such

as ring-, tree- and XOR-based ones, NCs have to be approximated into low-dimensional vectors or scalar values by a mapping function such as space-filling curves. In Section 4, we demonstrate the case studies of applying PIS/NC to an Euclidean-space-based DHT and a ring-based DHT by using CAN [13] and Chord [16], respectively.

3.2 PIS/NC Advancements

PIS/NC improves three important drawbacks of conventional PIS discussed in Section 2. First, PIS/NC clears the improvement limitation of the lookup latency of conventional PIS. Conventional PIS often assigns distant IDs to nearby nodes and similar IDs to distant nodes. Unlike them, PIS/NC provides a nearly-optimal topology-aware DHT. This is because PIS/NC builds a logical ID space dependent upon an NC space.

Second, PIS/NC significantly reduces key/value transfers compared to conventional PIS. Since NCs are mapped on an absolute space shared by all the nodes, a node ID introduced by its NC can be also mapped on an absolute space. Therefore, PIS/NC can assign a topology-aware ID to a joining node without key/value transfers. Although PIS/NC may cause key/value transfers when the NCs are significantly changed, some low-pass based filters [28] can reduce the frequency of the transfers because they moderate the frequency of the movements of NCs.

Third, PIS/NC moderates the load-balance and the fault-resilience problems by "PIS/NC stabilizer". PIS/NC stabilizer makes the ID distribution moderately-uniform as close to the original DHT as possible. To moderate the unbalanced ID distribution, PIS/NC stabilizer adjusts a node ID so that the logical distance from the node ID to the nearby nodes' IDs. PIS/NC stabilizer modifies node IDs so that the relative distance on the logical space among nearby nodes becomes uniform. By repeating this operation, PIS/NC stabilizer makes unbalanced node ID distribution uniform as much as possible, keeping the positional relationship among NCs. Although PIS/NC stabilizer slightly increases key/value transfers, it makes PIS/NC-based DHTs more load-balanced and fault-resilient. To make a PIS/NC-based DHT more fault-tolerant, we can take other techniques such as Glacier [29] and Phoenix [30].

Note that the frequency of key/value transfers increases as PIS/NC stabilizer becomes more sensitive. Thus, we should carefully set the sensitivity of PIS/NC stabilizer so that the key/value transfers caused by PIS/NC stabilizer can be negligible. It is future work to find out the appropriate frequency of running PIS/NC stabilizer for each application of PIS/NC-based DHTs.

3.3 Discussion

Although PIS/NC clears the critical issues of conventional PIS, PIS/NC still poses several unavoidable issues of PIS. However, we think that the remaining issues are not really matters for practical use if we carefully consider the scope of application of PIS/NC. In this section we discuss the remaining issues of PIS/NC and the scope of application of PIS/NC.

3.3.1 Number of Forwarding Hops

Since PIS/NC assigns a node ID based on the underlay topology, the number of forwarding hops of a query in a PIS/NC-based DHT will be slightly larger than that in the original DHT. In most of the traditional DHTs, the number of forwarding hops is theoretically calculated, e.g., $O(log(N))$ in Chord and $O(N^{1/d})$ in CAN, where N denotes the number of nodes and d denotes the dimensionality of the CAN space. PIS/NC-based DHTs cannot ensure that a query reaches the destination in the theoretical number of hops. This is because the density of the node distribution in a PIS/NC-based DHT becomes non-uniform due to the underlay topology. Although PIS/NC stabilizer moderates the non-uniform ID distribution of a PIS/NC-based DHT, this issue still remains to be unavoidable.

We think that most of P2P applications can accept the demerit of the increased number of forwarding hops to get the merit of the lookup latency improvement. To significantly improve the scalability and the availability, DHTs are often used in a system not to require a strict service level agreement. Such a system needs a method to reduce the lookup latency even if the number of forwarding hops of several queries are increased. As described in Section 5, PIS/NC reduces the lookup latency more than PNS strategy, while it increases the number of forwarding hops a bit. Thus, PIS/NC-based DHTs can be used in most of P2P applications in practical use. Besides, if you can, clustering nearby nodes reduces the number of forwarding hops.

3.3.2 Security Threats

PIS/NC-based DHTs should take care of network security threats such as the Sybil attacks [31]. Since PIS/NC-based DHTs assign a node ID based on node's

NC, an attacker may maliciously maintain a part of the logical DHT space with illegally-generated NCs. However, we believe that it would be difficult because an attacker hardly guesses the network coordinate corresponding to the ID which the attacker wants to generate. In addition, PIS/NC-based DHTs can improve its security level by the encryption and the authentication of the IDs and NCs.

3.3.3 The Scope of Application of PIS/NC

PIS/NC is applicable to DHT-based systems which require maximum improving the lookup latency rather than load-balanced and fault-tolerant quality. PIS/NC reduces the lookup latency of DHTs than PNS and PRS, but PIS/NC may still break some theory-based features of DHTs even if it improves the weakness of conventional PIS. In particular, the ID distribution of PIS/NC-based DHTs cannot always be supposed to be uniform because the ID assignment is depend on the location of a node. Thus, PIS/NC is useful for P2P-based systems which are deployed on wide area and require good responsiveness. For example, PIS/NC is applicable to P2P-based file sharing services like BitTorrent and P2P-based streaming services. This is because the clients on such a P2P-based system tend to be distributed all over the Internet and require quick responses to their queries for their desired files, music or movies.

We also suggest that PIS/NC is not applicable to the critical modules of a system because PIS/NC slightly degrades the load-balanced and fault-tolerant quality of DHTs. If you need such a quality, you should take PNS or PRS strategies rather than PIS/NC.

4. Implementation

4.1 Overview

To demonstrate the effectiveness of PIS/NC, we design two DHTs; *Canary* and *Harpsichord*, which are PIS/NC-based CAN and Chord, respectively. They use Vivaldi coordinates when they construct the overlay networks of CAN or Chord.

Vivaldi [2] is one of the popular NCs and enables nodes to estimate the network latency among them from their coordinate distance. Each node updates its own coordinate based on a spring model with piggy-backing its usual communications between others [⋆1]. When node i communicates with another node j and obtains C_j and L_{ij} , where C_j is the coordinate of j, and L_{ij} is the measured link latency between them, it updates its coordinate C_i according to the following formula,

$$C_i = C_i + \delta * (L_{ij} - ||C_i - C_j||) * u(C_i - C_j)$$

where δ is a weighting factor of the coordinate movement. Ledlie, et al. improved the accuracy and the stability of Vivaldi by some schemes such as Moving Percentile (MP) filter [28]. They also verified the accuracy of Vivaldi by experiments using the real a million-node BitTorrent overlay on the Internet [10].

CAN [13] and Chord [16] are well-known DHTs. In these traditional DHTs, each node is assigned its own ID and a subset of the key space to manage *randomly*. This means that the overlay topologies of them are independent from the underlay ones. Therefore, the queries often detour geographically. On the other hand, in PIS/NC-based DHTs, i.e., Canary and Harpsichord, the IDs and the management domains of nodes are assigned on the basis of their Vivaldi coordinates. This associates the positional relationship on the overlay with that on the underlay, and results in the reduction of the detour routing.

4.2 Canary

Canary is a PIS/NC-based CAN. As the key space, CAN uses d-dimensional Euclidean coordinate space. The whole coordinate space is divided into N zones, where N denotes the number of nodes in CAN, and a zone is assigned to a node. Routing from a source node to the destination node boils down to routing from one zone to another in the Euclidean space. To build a PIS/NC-based overlay, Canary *directly* maps Vivaldi coordinates of nodes to the d-dimensional Euclidean space of CAN. Since Canary assures that the coordinates of nodes are contained in each zone, the nodes managing adjacent zones are also close on the underlay network. It reduces latency per hop and results in the significant reduction of the total lookup latency. **Figure 4** illustrates an example of Canary [⋆2]. In this example, a query is forwarded from A to B via F on the overlay. The forwarding path is significantly efficient from a point of view of the underlay. This is because Canary provides a logical space which approximates the physical network topology.

Canary implements PIS/NC stabilizer as follows. To fix the non-uniform dis-

⋆1 Height vector is not used.
⋆2 Although we let the dimensionality be two here to simplify, as described in Section 5, we set it to three or six in experiments.

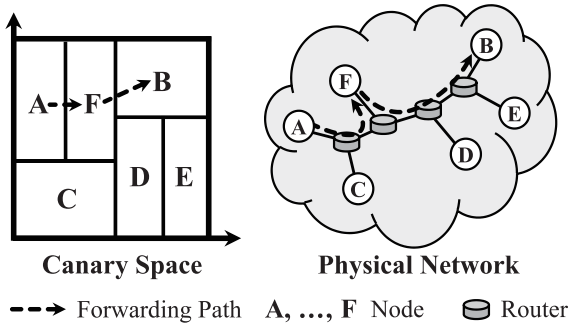


Fig. 4 A forwarding path of a query on Canary network and its physical network.

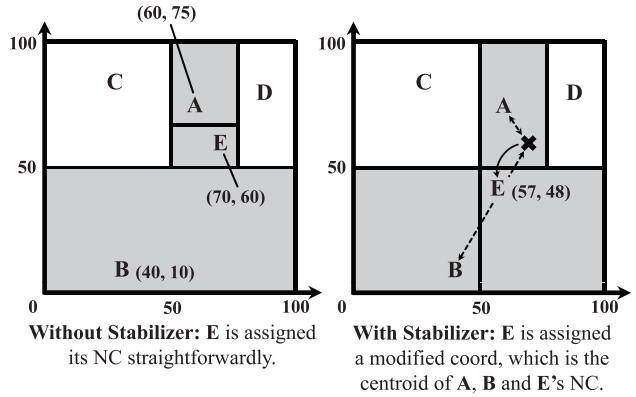


Fig. 5 The coordinate modification by Canary's PIS/NC stabilizer.

tribution of node IDs, Canary's PIS/NC stabilizer assigns each node a modified coordinate as a node ID instead of straightforwardly assigning the NC as necessary. When a node joins into Canary, the node first obtains the coordinates of the nodes nearby its NC and calculates the distance to the nearby nodes. If the ratio of the distance to the farthest node and that to the closest node is quite large, PIS/NC stabilizer assigns the node the coordinate of the centroid among the three coordinates, i.e., the closest coordinate, the farthest one and its NC. **Figure 5** shows an example of the adjustment by Canary's PIS/NC stabilizer. The left figure illustrates an example of Canary without PIS/NC stabilizer. In this case, newcomer E is assigned its Vivaldi coordinate straightforwardly. Then, it obtains its zone from A. As a result, the size of E's zone is very small whereas that of B's zone is quite large. The right figure illustrates an example of Canary with PIS/NC stabilizer. PIS/NC stabilizer assigns E a coordinate $(57, 48)$, which is the centroid of $A = (60, 75)$, $B = (40, 10)$ and E's NC $= (70, 60)$. Compared to Canary without PIS/NC stabilizer, the non-uniform distribution of node's coordinates and the size of node's zones are moderated. In the prototype of Canary, PIS/NC stabilizer runs if the ratio of the distance to the farthest nearby node and that to the closest one becomes larger than two.

To adapt to the persistent changes of the underlay topology, Canary dynamically modifies its structure as necessary. On the Internet, the underlay topology occasionally changes over time due to such as generating or eliminating physical links between routers. They would decrease the topology correlation gradually. To prevent this, Canary adjusts zones according to the Vivaldi coordinate movement. When a coordinate of a node protrudes its zone, the node adjusts its zone so that it contains the new coordinate. Thus, Canary keeps high approximate accuracy constantly. In Canary, the frequency of transferring stored values between zones is slightly larger than that in CAN. To reduce undesirably transferring stored values, we can employ a technique, such as update filter [28], to moderate the frequency of zone adjustments.

To decrease unnecessary key/value transfers, Canary allows a zone to be a complex shape when a node adjusts its zone to keep high correlation between the overlay and the physical network, while CAN allows a zone to be only a simple hyper-cuboid. If Canary confines a zone to be a simple hyper-cuboid like original CAN, an adjustment for keeping high correlation indicates many key/value transfers between not only an intruding and an intruded nodes but also other adjacent nodes. To implement this, we introduce an area and a block to replace a zone. An area is the entire domain of management for each node. A block is a simple hyper-cuboid which composes an area. In other words, an area consists of one or more blocks. A pseudo-code for Canary is shown in Appendix A.1.

4.3 Harpsichord

Harpsichord is a PIS/NC-based Chord. Chord uses ring-shaped 160-bits ID

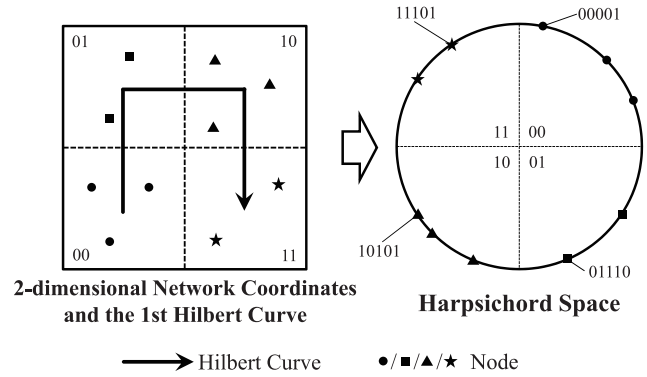


Fig. 6 The generation of node IDs based on Vivaldi coordinates by using Hilbert curve in Harpsichord.

space as the key space and assigns unique IDs to nodes at random. Each node maintains its own successor list and finger table as the routing table. On the other hand, in Harpsichord, each node generates its node ID on the basis of its NC. To map a d-dimensional NC to an ID on the ring-shaped space of Chord, Harpsichord uses a Hilbert curve [32], which is one of the popular space-filling curves. According to the m-th d-dimensional Hilbert curve (m is the approximation degree of Hilbert curve), the whole d-dimensional coordinate space is divided into 2^{md} regions. Each region is assigned its region ID, which is md-bits binary. A node belonging to a region is assigned a node ID whose prefix is the region ID. The lower bits of the node ID is assigned randomly. **Figure 6** shows an example of the conversion from a 2-dimensional coordinates to a scalar ID by the 1st Hilbert curve. In this example, the nodes indicated by a sphere are assigned IDs whose prefixes are "00". Similarly, the nodes indicated by a square, a triangle and a star are assigned IDs whose prefixes are "01", "10" and "11", respectively. As a result, Harpsichord nodes reflect their NCs into their IDs.

Harpsichord implements PIS/NC stabilizer as follows. If the ratio of node's logical distance to its predecessor and that to its successor becomes larger, the node replaces its ID with the new one where the logical distance to the predecessor equals to that to the successor. By repeating this operation for each node, unbalanced ID distribution will be fixed. **Figure 7** shows an example of the

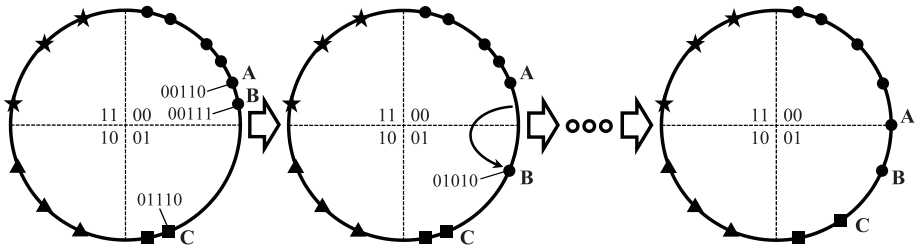


Fig. 7 The moderation of the non-uniform ID distribution by Harpsichord's PIS/NC stabilizer.

ID modification. B replaces its own ID "00111" with the modified one "01010", which is the mean value of the ID of A (i.e., B's predecessor) "00110" and the ID of C (i.e., B's successor) "01110". In the prototype of Harpsichord, PIS/NC stabilizer runs if the ratio of the distance to the predecessor and that to the successor becomes larger than two.

Harpsichord takes a simple method to deal with permanent changes of the underlay topology. If the coordinate of a node intrudes into another region of an NC space, the node leaves from and rejoins in Harpsichord. Unlike Canary, Harpsichord does not need a complex method for the movements of NCs because the effect of node ID movements is intrinsically restricted to logically adjacent four or more nodes in ring-based DHTs.

5. Experiments

We conducted two experiments to demonstrate that PIS/NC reduces the lookup latency and imposes the weak points of conventional PIS. We compared the improvement degree of lookup latency and the weak points between PIS/NC and other strategies, i.e., PNS (DHash++ [18]) and conventional PIS (SAT-Match [23]) from two perspectives, i.e., the absolute value and the relative error. In addition, we compared how PIS/NC-based DHTs change the features of each base DHT from two perspectives, i.e., the number of forwarding hops and the management domain size. We also evaluated PIS/NC without the PIS/NC stabilizer in order to verify its effect.

5.1 Evaluation Methodology

To evaluate PIS/NC, we built Canary and Harpsichord on Overlay Weaver [33], a toolkit that enables a structured overlay network to be easily constructed and its behavior to be emulated. We tested six DHTs, i.e., Canary, Harpsichord, CAN with SAT-Match, original CAN, DHash++, and original Chord, in two different environments, synthetic and real. In each environment, we used the Transit-Stub (TS) model [34] and PlanetLab nodes [22], respectively. Under these environments, each node sends lookup queries to random targets. We measured five key metrics.

- *Lookup latency*, which shows the delay time for a query to reach the destination node. The lower the lookup latency is, the better it is.
- *Relative error*, which is relative error between the actual lookup latency and the ideal lookup latency. It shows the efficiency of forwarding queries on overlay networks. Relative error is calculated by $(L_{actual} - L_{ideal})/L_{ideal}$, where L_{actual} is the latency it takes for a query to reach the destination node, and L_{ideal} is the minimum latency from the source node to the destination node on the underlay. The more the relative error approaches zero, the better the forwarding efficiency is.
- *Number of hops*, which is the number of forwarding hops for a query to reach the destination node. Its distribution shows one of the features of DHTs. If the distribution of an enhanced DHT is different from that of its base DHT, the enhanced DHT breaks the complexity order of forwarding hops of the base DHT. Basically, the lower the number of hops, the better it is.
- *Key domain size of each node*, which is the size of a key space subset managed by each node. It indicates the degree of load-balance. The smaller the variation of the size, the more balanced the load for lookups among nodes is.
- *Total transfered key range*, which is the sum of the transfered key range when participant nodes reconstruct thier management domains. This metric is aimed at Canary (with or without the PIS/NC stabilizer) and SAT-Match because its purpose is to show that the frequency of key range adjustments of PIS/NC is more moderate than that of a heuristic strategy such as SAT-Match.

In these experiments, the initial Vivaldi coordinates of nodes are calculated on the basis of GNP [4]. The parameters of the MP filter (history window size h and percentile p) are configured $h = 4$, $p = 25$. The approximation degree m of the Hilbert curve in Harpsichord is configured $m = 1$.

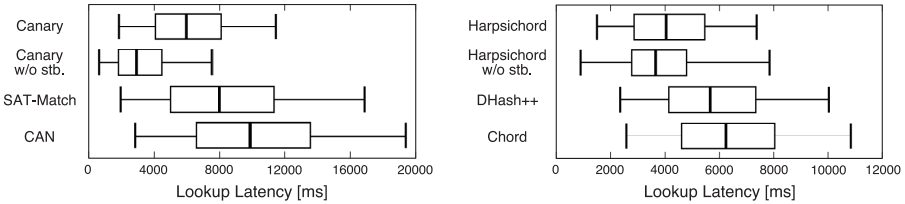


Fig. 8 The results of lookup latency with TS model.

5.2 Transit-stub Model

To evaluate and compare PIS/NC-based DHTs and other ones, we prepared TS model nodes, which represents a network as a two-level hierarchy of routing domains, i.e., transit domains interconnecting lower-level stub domains. To generate a TS topology, we assigned the parameters as follows: 228 transit domains, 5 transit nodes per transit domain, 4 stub domains attached to each transit node, and 2 nodes in each stub domain. We selected about 900 nodes from about 9,000 nodes. With this model, we tested each DHT operating 70,000 lookups. Although the network latency between nodes does not change during this experiment, the coordinate of each node moves in accordance with the Vivaldi algorithm. We set the dimensionality of the Euclidean coordinate space to three.

Figure 8 shows the lookup latency in the form of a box-and-whisker diagram. The crossbar in each box indicates the median. This figure shows that PIS/NC-based DHTs reduce lookup latency compared with not only their original DHTs but also a conventional PIS (SAT-Match) and PNS-based DHT (DHash++). The median latency of Canary and Harpsichord are reduced by 40% and 35% from their base DHTs; CAN and Chord, respectively. These reduction ratios are higher than those of SAT-Match (19%) and DHash++ (9%), respectively. Without the PIS/NC stabilizer, the reduction ratios become better because the PIS/NC stabilizer slightly degrades the approximation accuracy of the underlay.

Figure 9 shows the cumulative distribution function (CDF) of the relative error. The x-axis is log scale. These graphs compare the lookup latency from

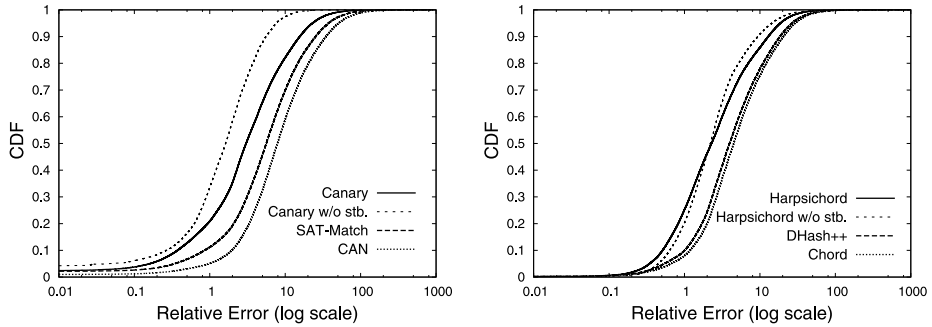


Fig. 9 The results of relative error of lookup latency with TS model.

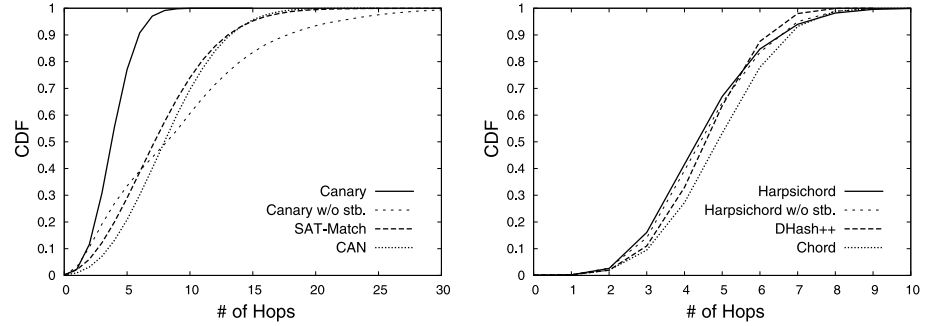


Fig. 10 The results of # hops with TS model.

another perspective, i.e., forwarding efficiency. They show that the relative error of PIS/NC-based DHTs is less than that of other DHTs. This means that Canary and Harpsichord forward queries with a higher degree of efficiency than others. Compared between the median, they are 3.03 and 2.28 in Canary and Harpsichord, respectively, while they are 5.61 and 3.95 in SAT-Match and DHash++, respectively.

Figure 10 shows the CDF of the number of forwarding hops. From these graphs, we can see that the distribution of forwarding hops of SAT-Match, DHash++, and Harpsichord are scarcely different from each base DHT, i.e., CAN or Chord. In contrast, the distribution of Canary is different from that of CAN. In CAN, most of the hops are in the range from 5 to 15. On the other hand, in Canary, almost all the hops are within 10. This result means that Canary breaks the complexity order of forwarding hops of CAN. This is because CAN and Canary use different ways to divide the management domains. While CAN divides a zone into halves, Canary divides an area at the center of the coordinates. Therefore, in Canary, the number of neighbors tends to be larger than that in CAN. As a result, a node in Canary can access other nodes within less hops than CAN.

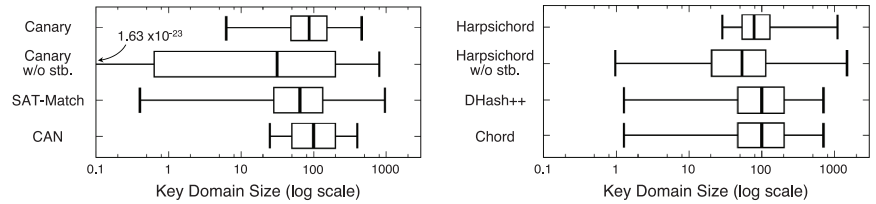


Fig. 11 The results of the distribution of key domain sizes with TS model.

Figure 10 also shows that the number of forwarding hops are very widely distributed in Canary without the PIS/NC stabilizer, even though a node has many neighbors. While 30% of lookups are completed within 5 hops, 20% of them take more than 15 hops. This is because the coordinates are not distributed uniformly when the PIS/NC stabilizer does not run. Thus, there are some spots on which many nodes lie densely. In such spots, many tiny zones would be produced. Therefore, if a query passes through such a spot, it tends to take more hops than in CAN. Since the non-uniform distribution of the coordinates is moderated, such an increase in forwarding hops does not occur in Canary with the PIS/NC stabilizer enabled.

Figure 11 shows the distribution of the key domain size in the form of a box-and-whisker diagram. The values in the diagram are normalized, setting the value of the median in CAN and Chord as 100, and the horizontal axis is a log scale. These graphs show that the PIS/NC stabilizer moderates the load-balancing issue. When the PIS/NC stabilizer runs, both Canary and Harpsichord diminish the non-uniform distribution of key domain size of nodes. On the other hand, Canary without the PIS/NC stabilizer generates many tiny zones and a few vast ones. As a result, several nodes which manage such vast zones have to

Table 2 The sum of the transfered key range during the experiment with TS model (normalized).

Canary	Canary w/o stb.	SAT-Match
43.3	46.2	100.0

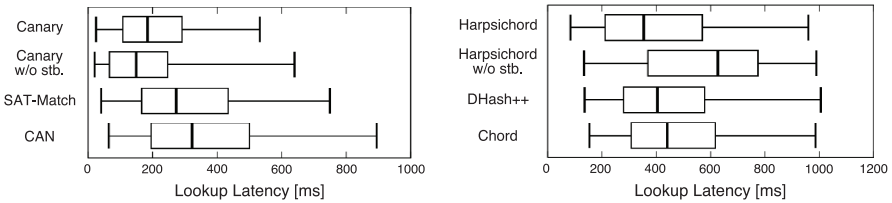


Fig. 12 The results of lookup latency with PlanetLab.

deal with a large number of queries.

Table 2 shows the sum of the transfered key range on the Euclidean space between nodes of Canary (with or without the PIS/NC stabilizer) and SAT-Match during the experiment. The values are normalized, setting the value in SAT-Match as 100. The total transfered key range of Canary, even if the PIS/NC stabilizer does not run, is less than half of that of SAT-Match.

5.3 PlanetLab Data Set

To evaluate PIS/NC-based DHTs and other ones in a real network environment, we used the latency measurements from 105 PlanetLab nodes between September 4$^{\text{th}}$ and 22$^{\text{nd}}$, 2008 as our simulator's delay matrix. With this data set, we tested each DHT operating 190,000 lookups. In this experiment, the network latency between nodes changes over time based on the measurement data of PlanetLab nodes. We set the dimensionality of the Euclidean space to six.

Figure 12 shows the lookup latency in the form of a box-and-whisker diagram. This figure shows that Canary and Harpsichord forward the queries with lower latency than not only their base DHTs but also the existing enhanced ones on the Internet as well. Compared with the reduction ratios of the median from each base DHT, those of Canary and SAT-Match are 43% and 23%, respectively. And those of Harpsichord and DHash++ are 20% and 8%, respectively. Intriguingly, Harpsichord without the PIS/NC stabilizer increases the median lookup latency.

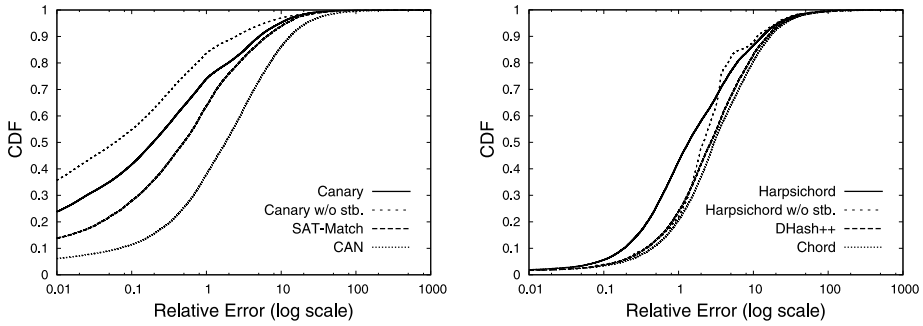


Fig. 13 The results of relative error of lookup latency with PlanetLab.

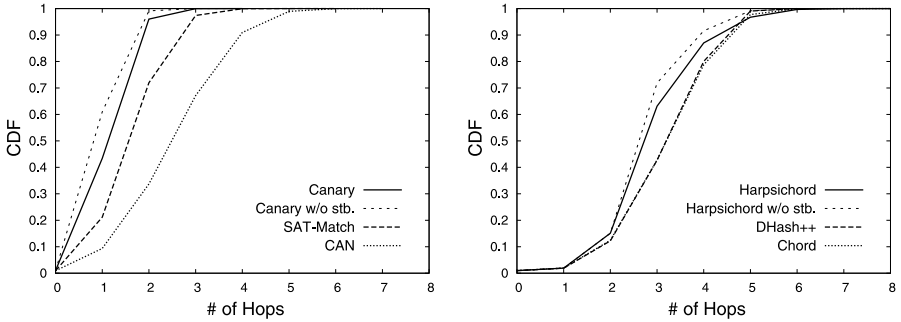


Fig. 14 The results of # hops with PlanetLab.

This is because some high latency nodes are forced to manage the large key ranges due to the non-uniform distribution of the NCs.

Figure 13 shows the relative error of the DHTs in CDF. PIS/NC-based DHTs forward queries more efficiently than the existing DHTs. Canary with the PIS/NC stabilizer shows better results than not only original CAN but also SAT-Match; about one-quarter of all the queries are forwarded on almost the ideal path. When the PIS/NC stabilizer does not run, Canary shows more remarkable results; 35% of all the queries are forwarded on almost the ideal path. Due to the unbalanced ID distribution and the low number of nodes, many queries reach their targets within a few hops in Canary without the PIS/NC stabilizer.

Figure 14 shows the number of hops in CDF. Unlike the experiment using

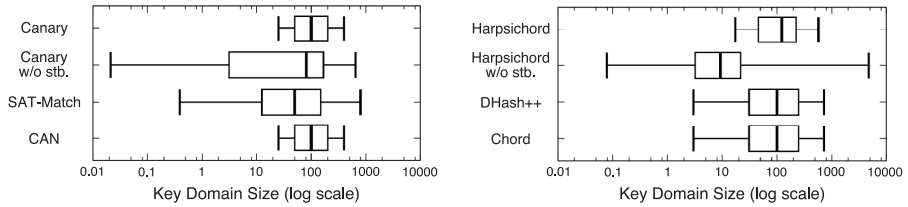


Fig. 15 The results of the distribution of key domain sizes with PlanetLab.

Table 3 The sum of the transfered key range during the experiment with PlanetLab (normalized).

Canary	Canary w/o stb.	SAT-Match
17.9	22.4	100.0

the TS model, Canary with and without the PIS/NC stabilizer are superior to other DHTs in terms of the number of hops. This would be because there are few spots which are densely-packed with nodes. Thus, the increase of hops is curbed in this experiment.

Figure 15 shows that the distribution of the key domain size in the form of a box-and-whisker diagram. In a real environment, Canary and Harpsichord with the PIS/NC stabilizer diminish the non-uniform distribution greatly. On the other hand, the non-uniform distribution is remarkable in Harpsichord without PIS/NC stabilizer. Compared with Chord, the median is one-tenth of that and the maximum is more than six times larger. These results suggest that most of the nodes manage small key ranges and a few nodes are forced to manage extremely large ones and will become hotspots.

Table 3 shows the sum of the transfered key range between nodes of Canary (with or without PIS/NC stabilizer) and SAT-Match. As well as the experiment with TS model, the values are normalized, setting the value in SAT-Match as 100. The total transfered key range of Canary is around one fifth of that of SAT-Match.

6. Conclusion

We propose a novel approach, called *PIS/NC*, to improve lookup latency in DHTs. PIS/NC embeds NCs into the heart of traditional DHTs. By using PIS/NC, we can improve the performance of traditional DHTs. PIS/NC also clears three critical issues of conventional PIS by using NCs and simple optimizations. For our first attempt, we designed Canary and Harpsichord to demonstrate the possibility of reducing the lookup latency of DHTs. By embedding Vivaldi coordinates in the mapping spaces of CAN and Chord, we integrated the physical link latency into the overlay networks. Simulation results demonstrate that the lookup latency of Canary and Harpsichord are the lowest among all prepared DHTs. Compared to original CAN and Chord, Canary and Harpsichord reduce the median lookup latency by 40% and 35%, respectively. The results also show that the degree of improvement is larger than those of SAT-Match or DHash++. For future work, we plan to apply PIS/NC to actual applications.

References

1) Costa, M., Castro, M., Rowstron, A. and Key, P.: PIC: Practical Internet Coordinates for Distance Estimation, *Proc. IEEE ICDCS*, pp.178–187 (2004).
2) Dabek, F., Cox, R., Kaashoek, F. and Morris, R.: Vivaldi: A Decentralized Network Coordinate System, *Proc. ACM SIGCOMM*, pp.15–26 (2004).
3) de Launois, C., Uhlig, S. and Bonaventure, O.: A Stable and Distributed Network Coordinate System, Technical report, Universite Catholique de Louvain (2004).
4) Ng, T.S.E. and Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches, *Proc. IEEE INFOCOM*, pp.170–179 (2002).
5) Ng, T.S.E. and Zhang, H.: A Network Positioning System for the Internet, *Proc. USENIX*, pp.141–154 (2004).
6) Pias, M., Crowcroft, J., Wilbur, S., Harris, T. and Bhatti, S.: Lighthouses for Scalable Distributed Location, *Proc. IPTPS* (2003).
7) Shavitt, Y. and Tankel, T.: Big-bang Simulation for Embedding Network Distances in Euclidean Space, *IEEE/ACM Trans. Networking*, Vol.12, No.6, pp.993–1006 (2004).
8) Tang, L. and Crovella, M.: Virtual Landmarks for the Internet, *Proc. ACM SIGCOMM IMC*, pp.143–152 (2003).
9) Azureus BitTorrent Client: http://azureus.sourceforge.net
10) Ledlie, J., Gardner, P. and Seltzer, M.: Network Coordinates in the Wild, *Proc. USENIX Symp. NSDI*, pp.299–311 (2007).
11) Kaashoek, F. and Karger, D.R.: Koorde: A Simple Degree-optimal distributed Hash Table, *Proc. IPTPS*, pp.98–107 (2003).
12) Maymounkov, P. and Mazières, D.: Kademlia: A Peer-to-peer Infomation System Based on the XOR Metric, *Proc. IPTPS*, pp.53–65 (2002).

13) Ratnasamy, S., Francis, P., Handley, M., Karp, R.M. and Shenker, S.: A Scalable Content-Addressable Network, *Proc. ACM SIGCOMM*, pp.161–172 (2001).
14) Rhea, S., Geels, D., Roscoe, T. and Kubiatowicz, J.: Handling Churn in a DHT, *Proc. USENIX* (2004).
15) Rowstron, A.I.T. and Druschel, P.: Pastry: Scalable Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems, *Proc. IFIP/ACM Middleware*, pp.329–350 (2001).
16) Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *Proc. ACM SIGCOMM*, pp.149–160 (2001).
17) Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D. and Kubiatowicz, J.D.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment, *IEEE Journal on Selected Areas in Communications*, No.22, pp.41–53 (2004).
18) Dabek, F., Li, J., Sit, E., Robertson, J., Kaashoes, M.F. and Morris, R.: Designing a DHT for low latency and high throughput, *Proc. USENIX Symp. NSDI*, pp.85–98 (2004).
19) Kaune, S., Lauinger, T., Kovacevic, A. and Pussep, K.: Embracing the Peer Next Door: Proximity in Kademlia, *Proc. IEEE Int'l Conf. P2P*, pp.343–350 (2008).
20) Rhea, S., Chun, B.-G., Kubiatowicz, J. and Shenker, S.: Fixing the Embarrassing Slowness of OpenDHT on PlanetLab, *Proc. WORLDS*, pp.25–30 (2005).
21) Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S.J. and Stoica, I.: The Impact of DHT Routing Geometry on Resilience and Proximity, *Proc. ACM SIGCOMM*, pp.381–394 (2003).
22) Bavier, A.C., Bowman, M., Chun, B.N., Culler, D.E., Karlin, S., Muir, S., Peterson, L.L., Roscoe, T., Spalink, T. and Wawrzoniak, M.: Operating Systems Support for Planetary-Scale Network Services, *Proc. USENIX Symp. NSDI*, pp.253–266 (2004).
23) Ren, S., Guo, L., Jiang, S. and Zhang, X.: SAT-Match: A Self-Adaptive Topology Matching Method to Achieve Low Lookup Latency in Structured P2P Overlay Networks, *Proc. IEEE IPDPS*, pp.83–91 (2004).
24) Ratnasamy, S., H, M., Karp, R. and Shenker, S.: Topologically-Aware Overlay Construction and Server Selection, *Proc. IEEE INFOCOM*, pp.1190–1199 (2002).
25) Locher, T., Schmid, S. and Wattenhofer, R.: eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System, *Proc. IEEE Int'l Conf. P2P*, pp.3–11 (2006).
26) Xu, Z. and Zhang, Z.: Building Low-maintenance Expressways for P2P Systems, Technical report, Hewlett-Packard Laboratories Palo Alto (2002).
27) Freedman, M.J. and Mazières, D.: Sloppy Hashing and Self-Organizing Clusters, *Proc. IPTPS*, pp.45–55 (2003).
28) Ledlie, J., Pietzuch, P. and Seltzer, M.: Stable and Accurate Network Coordinates, *Proc. IEEE ICDCS* (2006).
29) Haeberlen, A., Mislove, A. and Druschel, P.: Glacier: Highly durable, decentralized storage despite, *Proc. USENIX Symp. NSDI* (2005).
30) Junqueira, F., Bhagwan, R., Hevia, A., Marzullo, K. and Voelker, G.M.: Surviving Internet Catastrophes, *Proc. USENIX*, pp.45–60 (2005).
31) Douceur, J.R.: The Sybil Attacks, *Proc. IPTPS*, pp.251–260 (2002).
32) Hilbert, D.: Über die stetige Abbildung einer Linie auf ein Flächenstück, *Mathematische Annalen*, No.38, pp.459–460 (1891).
33) Shudo, K., Tanaka, Y. and Sekiguchi, S.: Overlay Weaver: An Overlay Construction Toolkit, *IPSJ Trans. Advanced Computing Systems*, 46(SIG 4 (ACS 9)), pp.33–44 (2005).
34) Zegura, E.W., Calvert, K.L. and Bhattacharjee, S.: How to Model an Internetwork, *Proc. IEEE INFOCOM*, pp.594–602 (1996).

Appendix

The pseudo-code of Canary and Harpsichord is shown below.

A.1 Query Message Handler of Canary

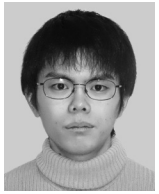
```
// At server i
if i receives a query message from node j then
  Get L_ij, C_j, and E_j from the query
  // L_ij is the latency between i and j
  // C_j is the Vivladi coordinate of j
  // E_j is the Vivaldi error of j
  Update its own Vivaldi coordinate based on the Vivladi algorithm using
  L_ij, C_i, C_j, E_i, E_j
  // Begin of PIS/NC stabilizer
  l_1 = ||C_i − n's ID ||
  l_2 = ||C_i − f's ID ||
  // Note, C_i is the updated coordinate
  // n and f denote the nodes whose coordinate distance from i are nearest
  or farthest in the neighbors of i, respectively
  if l_1/l_2 > T_1 then
    // T_1 is a threshold to determine whether a node ID should be modified
    Change i's ID to the centroid of n's ID, f's ID, and C_i
  else
    Set i's ID to C_i
```

```
    end if
    // End of PIS/NC stabilizer
    if i's ID is out of i's area then
      for all neighbor k do
        if k's area contains i's ID then
          Send a request message of dividing are to k
          Receive the divided are from k
          if the divided one is adjacent to i's area then
            Marge both areas
          else
            for all block b do
              Assign b with a neighbor node whose area is adjacent to b
            end for
            Set the divided area to i's area
          end if
        end if
      end for
    end if
    Choose a next node based on the CAN protocol
    Forward the query to the netx node
  end if
```

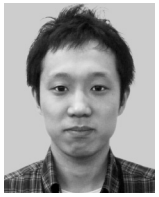
A.2 Query Message Handler of Harpsichord

```
// At server i
if if i receives a query message from node j then
  Get L_ij, C_j, and E_j from the query
  // L_ij is the latency between i and j
  // C_j is the Vivladi coordinate of j
  // E_j is the Vivaldi error of j
  Update its own Vivaldi coordinate based on the Vivladi algorithm using
  L_ij, C_i, C_j, E_i, E_j
  if ||C_i − C_p|| > T_2 and ||C_i − C_s|| > T_2 then
    // Note, C_i is the updated coordinate
    // p and s denote i's predecessor and successor, respectively
    // T_2 is a threshold to determine whether a node should leave and rejoin
    to the Harpsichord network to change its node ID to be appropriate to
    the topology of the underlay network
    Choose a next node based on the Chord protocol
    Forward the query to the next node
    Once leave, and rejoin to the Harpsichord network
    return
  end if
  // Begin of PIS/NC stabilizer
  l_1 = ||i's ID − p's ID||
  l_2 = ||i's ID − s's ID||
  if max(l_1/l_2, l_2/l_1) > T_3 then
    // T_3 is a threshold to determine whether a node ID should be modified
    Change i's ID to the mean of p's ID and s's ID
  end if
  // End of PIS/NC stabilizer
  Forward the query to the next node
end if
```

(Received May 7, 2010)
(Accepted September 6, 2010)
(Released March 4, 2011)
(Original version of this article can be found in the IPSJ Transactions on Advanced Computing Systems, Vol.4, No.1, pp.95–110.)

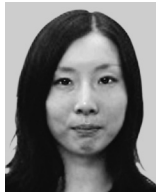


Toshinori Kojima received his B.E. and M.E. degrees from Keio University in 2008 and in 2010, respectively. Since 2010 he works for Research and Development Headquarters, NTT DATA CORPORATION. His research interests include network coordinates and distributed hash tables.



Masato Asahara received his B.E. degree from University of Electro-communications in 2005 and M.E. degree from Keio University in 2007, respectively. He was a Ph.D. student in Keio University from 2007 to 2010. Since 2010 he works for Service Platforms Research Labs., NEC Corp. His research interests include Peer-to-Peer systems, distributed systems, middleware and operating systems. He is a member of the IPSJ, IEEE/CS, ACM and USENIX.

Kenji Kono received his B.Sc. degree in 1993, M.Sc. degree in 1995, and Ph.D. degree in 2000, all in computer science from the University of Tokyo. He is an associate professor of the Department of Information and Computer Science at Keio University. His research interests include operating systems, system software, and Internet security. He is a member of the IEEE/CS, ACM and USENIX.



Ai Hayakawa received her B.E. and M.E. degrees from Keio University in 2008 and in 2010, respectively. Since 2010 she works for Nomura Research Institute, Ltd. Her research interests include Peer-to-Peer systems and content distribution networks.