Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

HybridNN: An accurate and scalable network location service based on the inframetric model

Yongquan Fu [a,*], Yijie Wang [a], Ernst Biersack [b]

[a] National Key Laboratory for Parallel and Distributed Processing, College of Computer Science, National University of Defense Technology, Hunan province, 410073, China
[b] EURECOM, France

ARTICLE INFO

Article history:
Received 15 January 2012
Received in revised form
4 October 2012
Accepted 7 December 2012
Available online xxxx

Keywords:
Distributed nearest server location
Peer-to-Peer
Inframetric
Growth dimension

ABSTRACT

Locating servers that have shortest interactive delay towards an Internet host provides an important service for large-scale latency sensitive networked applications, such as VoIP, online network games, or interactive network services on the cloud. Existing algorithms assume that the delay space is a metric space, which implies that the delay between two nodes is symmetric and the triangle inequality holds. In practice, the delay space is not metric, which lowers the accuracy of metric-based algorithms. We develop a new scheme whose theoretical foundation is based on the inframetric model, which has weaker assumptions than the metric model. We prove that the location requests can be completed efficiently if the delay space exhibits modest inframetric dimensions, which we can confirm empirically. Finally, we propose HybridNN (*Hybrid N*earest Service *N*ode Location) that finds the closest service node accurately thanks to the inframetric model and scalably by combining delay predictions with direct probes to a pruned set of neighbors. Simulation results show that HybridNN locates in nearly all cases the true nearest service nodes. Experiments on PlanetLab show that with modest query overhead and maintenance traffic HybridNN can provide accurate nearest service nodes that are close to optimal.

1. Introduction

1.1. Motivation

Latency-sensitive applications, such as overlay based VoIP [1–3], peer-to-peer distributed virtual environments [4], IPTV [5], interactive network services in the cloud (e.g. Office Live Workspace [6], Google Maps [7]), data sharing on e-science grids [8], High Performance Computing (HPC) over grid or cloud platforms [9], or online network games need to transmit data from geo-distributed servers (called service nodes) in real-time to many hosts. High transmission delays reduce the Quality of Experience (QoE) of users [10] and may lead to significant business losses.

Since there can be hundreds or thousands of service nodes that provide identical services to hosts, there is an increasing need to redirect hosts' requests to the closest service nodes. For example, Google routes search queries to nearby servers [11]; Akamai redirects content requests to replica servers based on the proximity criteria [12].

Locating nearest service nodes for large-scale Internet hosts is nontrivial. Selecting the service node with the minimum RTT value based on all-pair RTT values does not scale with increasing number of service nodes or Internet hosts. On the other hand, organizing service nodes as an overlay and searching the node nearest to the target in a distributed manner [13,14] is easily trapped into local minima, because of the clustering [15] and the Triangle Inequality Violations (TIV) [16] of the Internet delay space. As a consequence, selecting nearest service nodes for Internet hosts is still far from being fully solved.

1.2. Contribution

The goal of this paper is to design new algorithms that accurately and efficiently find the nearest servers for Internet hosts. We develop a general distributed nearest service node location scheme based on the inframetric model [17] that allows the existence of the TIV and asymmetry.

Based on the inframetric model, we analytically demonstrate that we can quickly find the approximately nearest server for arbitrary hosts by recursively searching for closer nodes in a distributed manner. We formally prove the relation between the search accuracy, search costs, and search hops. Our theoretical result requires to randomly sample enough neighbors to each node, which can be easily performed in a distributed manner. As a consequence, our theoretical results provide guidelines for devising new and practical nearest-server-location algorithms.

* Corresponding author. Tel.: +86 13875828390.
E-mail addresses: yongquanf@nudt.edu.cn, quanyongf@126.com (Y. Fu), wangyijie@nudt.edu.cn (Y. Wang), erbi@eurecom.fr (E. Biersack).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.12.002

We then design a novel distributed nearest-service-node-location algorithm, called HybridNN, which finds nearest service nodes for any Internet node (called a target). HybridNN is informed by our theoretical results for the inframetric model and preserves the accuracy and the speed proven theoretically. HybridNN

- Reduces the measurement cost by using the network coordinate based delay estimation.
- Performs a small number of active probes during the query process to increase the accuracy of the delay estimations, and to avoid that the queries are misled by inaccurate network distance estimations.
- Organizes neighbors in concentric rings and samples enough neighbors in each ring with biased neighbor sampling in order to selectively sample nodes among one node's closest neighbors and farthest neighbors.

We validate HybridNN using real-world delay data sets, which contain asymmetric delays, and a PlanetLab deployment. Through simulations we show that HybridNN finds servers close to the optimal one. In fact, in more than 80% of all cases, HybridNN locates the ground-truth nearest server. Most queries terminate within four search hops, which implies that HybridNN can return the search result fast. Using a PlanetLab deployment, we confirm that HybridNN accurately locates nearest servers with low query and control traffic overhead.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 states the nearest service node location problem. Section 4 theoretically analyzes the problem. Section 5 presents the details of HybridNN. Sections 6 and 7 evaluate HybridNN via simulation and PlanetLab experiments. Finally, Section 8 summarizes the contributions of the paper.

2. Related work

Existing work on the problem of the nearest service node location can be grouped into either centralized or distributed approaches by whether there exists a centralized node selects the nearest node to the target based on the global pairwise proximity distances.

A related problem is the nearest neighbor search in the metric space [18–20]. Unfortunately, since the pairwise RTTs often violate the triangle inequality that is assumed to hold in a metric space [21], it is infeasible to apply these theoretical results to the nearest service node location in our context.

2.1. Centralized approaches

For centralized approaches, a node (called repository node) stores pairwise proximity distances between all nodes and finds the nearest service node to an arbitrary target by sorting the proximity distances from all service nodes to the target. Depending on the ways of obtaining pairwise proximity, existing centralized approaches can be categorized into: (1) direct measurement based approach, (2) coordinate based approach, (3) topology based approach.

(1) The direct measurement based approach collects RTTs from all service nodes to the target based on direct delay measurements. Carter and Crovella [22,23] collect RTT measurements and bandwidth measurements from Internet hosts to servers and decide the nearest server to the host and select the optimal server that minimizes the file transmission time in a centralized manner.

(2) The coordinate based approach embeds all nodes into a geometric space and estimates pairwise proximity based on the pairwise coordinate distances. The repository node collects the coordinates of all nodes and selects the one with the minimum coordinate distance as the service node nearest to the target. Since the service nodes need not probe the target, the coordinate based approach reduces the measurement costs compared to the direct measurement based approach. Unfortunately, the estimation errors of coordinate distances also decreases the accuracy of locating the nearest service nodes.

Guyton et al. [24] pioneered the research on finding the closest server in terms of the routing hops estimated by the Hotz's metric [25]. Smaller-hop nodes are not necessarily the nearby nodes, since one hop may traverse a continent or stay inside a data center.

Ratnasamy et al. [26] cluster Internet hosts into proximity groups that have the same relative coordinates computed as vectors of proximity from Internet hosts to landmarks. Nodes in the same cluster are assumed to be equally close to each other. The clustering process can not directly compute the nearest service node. Furthermore, the cluster's accuracy is also not bounded because of the heuristic nature of the clustering process.

Netvigator [27] sorts the proximity from service nodes to targets based on the relative coordinates computed as vectors of RTTs from each node to landmarks and some routers found on the routing paths. Netvigator may return out-of-date results since the relative coordinates may be obsolete.

CRP [28] sorts the proximity from service nodes to targets with the coordinates computed as the vectors of redirection frequencies to CDN edge servers. CRP can degrade its accuracy when the redirection process is also affected by non-proximity policies such as loads or transit traffic costs. Besides, CRP fails to select the nearest service node when the candidate nodes and the target shares no common CDN edge servers.

(3) The topology based approach constructs a virtual Internet topology structure for a set of nodes and predicts pairwise proximity between these nodes based on the topology distances. For proximity queries, the targets and all services must be included into the topology; then the repository node computes the nearest service node to the target by sorting the topology distances from all service nodes to the target.

iPlane [29,30] predicts a compact Internet routing model for arbitrary Internet hosts and estimates the nearest node to any target based on the ordinal information from candidate nodes to the target. iPlane has to perform extensive measurements from distributed landmarks to wide-area Internet addresses in order to maintain an up-to-date routing model, with overall 2 million probes per day. Besides, iPlane has to select a large number of geo-distributed landmarks to cover the PoP-level links.

In summary, for large-scale service nodes or targets, the centralized approaches fail to trade off the accuracy and the scalability of locating nearest service nodes. Besides, the repository node may introduce performance bottlenecks as the system size increases.

2.2. Distributed approaches

For distributed approaches, service nodes are organized as an overlay and a greedy search process is adopted to recursively locate a next-hop neighbor that is nearer to the target. As a result, distributed approaches scale better than the centralized approaches, since no repository nodes are required. Besides, the measurement costs are also lowered, since each node determines a next-hop neighbor based on the local proximity information between neighbors and the target.

According to the ways of selecting the next-hop neighbor, existing approaches can be categorized into: (1) distributed clustering based approach, (2) on-demand probing approach, (3) coordinate based approach.

(1) The distributed clustering based approach partitions nodes into proximity clusters of nearby nodes with distributed heuristics. The target needs to probe RTT values to the cluster-head nodes

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

of all clusters and select the cluster of the target to be the same with the cluster-head node with the minimum RTT. Then, from the nodes in the same cluster with the target, the one with the minimum RTT to the target is selected as the target's nearest service node.

Tiers [31] maintains a distributed hierarchical clustering tree of nodes based on distributed heuristics. Each vertex in the tree denotes a real-world service node. Each proximity query is forwarded in a top-down approach: each upper-layer node selects the child node with the minimum RTT to the target as the next-hop node answering the proximity query. Unfortunately, Tiers may return a node far away from the ground-truth nearest node because of the heuristic based clustering process. Besides, the tree introduces load imbalances for nodes near the root of the tree.

Sequoia [32] organizes nodes as a virtual tree based on pairwise RTT or bandwidth values. The leaf nodes in the tree are real-world nodes; the internal vertices are virtual nodes for connecting different branches. Sequoia assumes the target and the service nodes are all embedded into the tree structure. When locating the nearest service node for a target, the target directly compares the tree distances from itself to nodes in nearby branches. However, since Sequoia assumes the delay space to be a tree metric that requires the triangle inequality to hold, the accuracy of proximity queries degrades when TIV occurs.

(2) The on-demand probing approach organizes nodes into an overlay and recursively finds a next-hop neighbor that is nearer to the target than the current node by comparing the direct RTT measurements from the neighbors to the target.

Mithos [33] constructs an overlay based on the gossip based peer sampling process similar as [34]. A proximity query is recursively forwarded to a neighbor nearer to the target. Unfortunately, Mithos is easily trapped into local minimum since the neighbor set of Mithos has limited diversity.

Meridian [13] improves the neighbors' diversity by maintaining neighbors in concentric rings with exponentially increasing radii. Meridian finds new neighbors through an anti-entropy gossip protocol [34] and retains neighbors that maximize the diversity of neighbors' locations. To search a server nearest to a target, Meridian recursively locates a neighbor that is β ($\beta \leq 1$) times closer to the target than the current node. Meridian has an interesting theoretical foundation that assumes the Internet delay space to be a growth or doubling metric [35,19].

Unfortunately, several subsequent study has shown that Meridian may be trapped at a local minimum due to the clustering phenomenon of the delay space [15] or the TIVs of the network delay space [21]. To address these problems, two adjustments have been proposed:

- explicitly being aware of the set of subnet nodes in the local clusters [15]. However, finding the local clusters becomes insufficient when clusters covering service nodes spreading multiple subnets. Besides, adding extra neighbors also increases Meridian's query overhead.
- adding all neighbors that may not be chosen due to the TIVs based on Vivaldi's prediction errors [21]. However, selecting all neighbors affected by TIVs is difficult because of the heuristic based TIV detection [21].

We have systematically analyzed the difficulties of finding the nearest nodes by Meridian [36].

(3) The coordinate based approach assigns a coordinate to each node, and recursively determines the neighbor nearer to the target in terms of coordinate distances. As a result, the measurement costs are reduced compared to the on-demand probing approach. Unfortunately, the estimation errors of coordinates also degrade the accuracy of the proximity queries.

PIC [37] is similar with Mithos, but recursively finds a neighbor that has lower coordinate distance to the target, assuming that the target maintains their own coordinates. OASIS [14] borrows Meridian's idea and recursively finds a neighbor closer to the target in terms of the geographical distances. As a result, OASIS has similar weaknesses with Meridian. Besides, OASIS may be trapped at local minimum since the wide-area RTTs are not always consistent with the geographical distances [11]. Finally, DONAR [38] combines the constraints of the geographical distances, the routing optimization and server loads to select the optimal nodes to the target, which generalizes our work.

Besides, although Netvigator and CRP select the nearest node to the target in a centralized manner, their coordinates can be leveraged in a decentralized manner like in PIC or OASIS.

In summary, existing distributed approaches are still far from practical in terms of trading off the accuracy and the scalability.

Table 1
Notation used.

Notation	Meaning
T	Host that issues the DNNL queries, i.e., the target
P	Set of service nodes
N	Number of service nodes
V	All nodes $V = P \cup T$
d	Pairwise delays between node pairs in V
β	The DNNL distance reduction threshold
ρ	Inframetric parameter
Δ	Maximal size of the ring
K	Size of sampled neighbors by the random walks
m	Number of neighbors for direct probes
τ	Number of non-empty rings
N_r	Number of rings in the concentric ring

3. Background

In this section we first explain how the *D*istributed *N*earest Service *N*ode *L*ocation (denoted as DNNL) works, we then present Meridian, a well-known DNNL system that assumes that the delay space is metric before we briefly describe the delay data sets used in this paper and introduce the inframetric model. Table 1 summarizes key notations using in the paper.

3.1. Problem definition

Throughout the paper we refer to the node for which we need to find the closest service node as the *target*. We assume service nodes may be added or removed at any time, which causes system churn.

We define the *Distributed Nearest Service Node Location* (DNNL) to operate as follows: For a set of dynamic service nodes, given any target T on the Internet, DNNL finds one service node that has the smallest delay to T, based on the collaboration among service nodes.

DNNL proceeds in iterations. At each step, the current service node P_i tries to locate a new service node P_{i+1} that is closer to the target T than P_i. For an illustration see Fig. 1: When a host T wants to access a networked service, it issues a DNNL query to locate the nearest service machine to T. The query message is first forwarded to any service node $P1$ of the DNNL service (Step 1). Then the DNNL query system will forward the query message recursively until locating a nearest service node $P3$ (Step 2 $\rightarrow$ 3). Finally, DNNL returns the address of node $P3$ to host T (Step 4). The service node $P3$ will then provide the required network service to host T.

To be useful for latency-sensitive applications, we identify the following key requirements for a DNNL:

- Accuracy, to find a service node with the lowest delay in order to increase the Quality of Experience of users.
- Speed, to obtain the nearest service node quickly. Since too long a query time makes the DNNL less attractive for server redirections of latency-sensitive applications.
- Scalability, the DNNL process should incur low bandwidth cost with increasing system size.

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

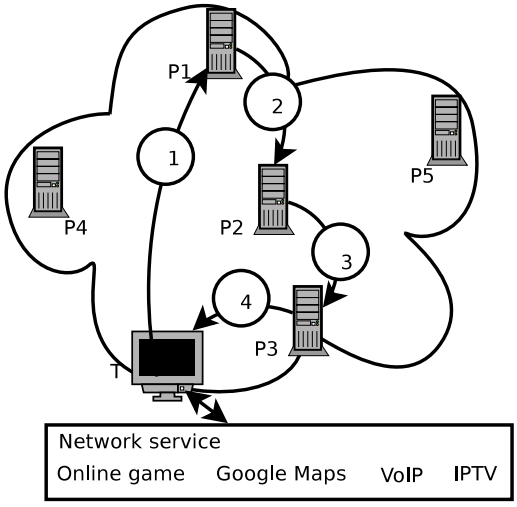


Fig. 1. DNNL query service substrate for network services.

- Resiliency to churn, to return accurate results even when some service nodes crash or new service nodes are added.

3.2. Meridian

We take Meridian as a state-of-art DNNL scheme that assumes that the Internet delay space is a metric space.

Each Meridian node P maintains a $O(\log(N))$-sized set of other Meridian nodes as logical neighbors, where N is the number of service nodes. The neighbors are organized in concentric rings with exponentially increasing radii: the i-th ring contains neighbors whose delays to node P lie in the interval $(\alpha s^{i-1}, \alpha s^{i}]$, with $i > 0$, α a constant, s a multiplicative increase factor. We will use $\alpha = 1, s = 2$ ms for Meridian.

To limit the storage overhead, the number of rings is limited to be a constant i^*, and the number of neighbors in each ring is bounded to be a constant k_r. Consequently, all rings $i > i^*$ are collapsed into a single outermost ring spanning the interval $(\alpha s^{i^*}, +\infty]$. Besides k_r neighbors in each ring, each Meridian node also maintains l additional neighbors (called secondary ring members) within each ring that are used to replace primary ring members when updating the rings.

Neighbors in each Meridian node are uniformly sampled with a gossip process. The gossip process is an anti-entropy "push" based gossip protocol, where each node periodically selects a node in its ring as the gossip counterpart, and pushes one neighbor per non-empty ring to the gossip counterpart.

A ring maintenance process is triggered periodically to update the set of primary neighbors of each ring by optimizing the geographic diversity, based on a maximum hypervolume polytope algorithm [13]. Each time, the ring maintenance process updates one ring, all-pair delay probes among the primary and secondary neighbors of that ring are necessary.

For a DNNL request, Meridian iteratively locates one next-hop node that is β ($\beta < 1$) times closer to the target T than the current Meridian node. Assuming the triangle inequality and symmetry to hold in the delay space, Meridian can find all candidate neighbors that are β times closer to the target in the following way: a Meridian node P measures its delay d_{PT} to the target T, then node P selects candidate neighbors from its concentric ring whose delays to P are within $[(1-\beta)d_{PT}, (1+\beta)d_{PT}]$, where β is the delay reduction threshold.

In Fig. 2 we have a Meridian node P and a target T, the neighbor selection objective of node P is to locate all next-hop neighbors whose distances to the target T are smaller than βd_{PT}. For each candidate neighbor A in the concentric ring of node P, thanks to the triangle inequality of the triple (P, T, A) as well as the symmetry of delays, the delay value d_{AP} will satisfy the following constraints:

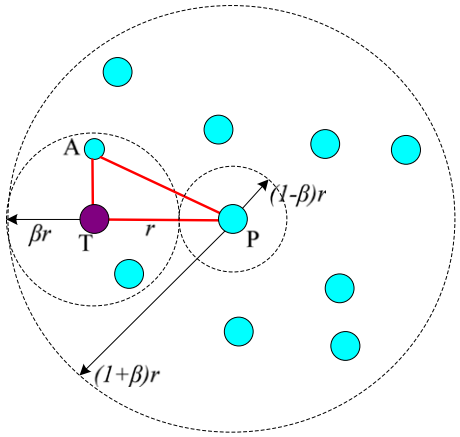


Fig. 2. Choosing a β times closer neighbor A to node T from P.

$$\begin{cases} d_{AP} \le d_{AT} + d_{PT} \\ d_{PT} \le d_{AP} + d_{AT} \\ d_{AT} \le d_{AP} + d_{PT} \\ d_{AT} \le \beta d_{PT} \end{cases} \Rightarrow \begin{cases} d_{AP} \ge |d_{PT} - d_{AT}| \ge (1-\beta)\, d_{PT} \\ d_{AP} \le d_{PT} + d_{AT} \le (1+\beta)\, d_{PT}. \end{cases}$$

The pseudo-code of Meridian is depicted in Algorithm 1. Line 2 determines the delay between the Meridian node P to the target T. Line 3 selects the candidate neighbors in the concentric ring of P that are within certain delay ranges to P by the triangle inequality. Then Lines 4–7 find the neighbor that is closest to the target T. Lines 8 $\longrightarrow$ 16 determine whether the DNNL process should be continued or stopped by testing the existence of the β times closer neighbor.

The measurement overhead of each step is $1 + |U|$, since each neighbor in U needs to probe target T. Furthermore, node P has to wait for the measurement results from neighbors in U, thus the completion period of the algorithm depends on the slowest response of neighbors in U.

Algorithm 1: Meridian procedure.

```
1  Meridian(P, T)
   input  : current node P, the target T
   output: nearest node to T
   // RTT measurements
2  d_PT ← RTTProbe(P, T);
3  U ← {i | i ∈ R_P ∧ (1 − β) d_PT ≤ d_iP ≤ (1 + β) d_PT };
4  for i ∈ U do
5      d_iT ← RTTProbe(i, T);
6  end
7  A ← arg min_{i∈U} d_iT;
8  if d_AT ≤ β d_PT then
9      Meridian(A, T);
10 else
11     if d_AT < d_PT then
12         return A;
13     else
14         return P;
15     end
16 end
```

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

3.3. Data sets

During the design of our DNNL algorithm we used four different delay data sets to evaluate its performance. These data sets cover the delays between wide-area DNS servers and those between end hosts [36].

Using publicly available delay data sets is a popular approach to assess the performance of nearest service node location methods [13,39,12,27]. Moreover, since the delay data sets are directly collected from the real Internet environments, the data sets are more faithful than those generated by synthetic tools. The data sets may contain some measurement noise due to the probing environment, which is valuable for testing the robustness of proposed methods.

In this paper, due to space limitations, we will report results for two of these data sets:

- *DNS3997*. A RTT matrix of delay measurements collected between 3997 DNS servers by Zhang et al. [40] using the King method [41]. The matrix is symmetric in that $d_{ij} = d_{ji}$, for any pair of items i and j, where d denotes the delay matrix.
- *Host479*. A RTT delay matrix based on RTT measurements among Vuze BitTorrent clients [42]. Each item d_{ij} denotes the aggregated RTT measurements from host i to host j. Due to the dynamics of Vuze clients, the measurements are not synchronized and different hosts may make measurements at different time slots. Since the delay variations (or jitter) occur frequently because of queueing and transient network congestion [43], it is not surprising that the Host479 matrix is asymmetric. The delay pairs d_{AB} and d_{BA} differ by more than a factor of four in about 40% of the cases [42].

3.4. The inframetric model for the delay space

Existing DNNL schemes such as Meridian have made the assumption that the delay space is metric. However, in general this is not true and the TIV phenomenon occurs frequently as shown in Fig. 3. We calculate the *TIV Severity* T_{ABC} for each triple (A, B, C) that does not contain missing edges, $T_{ABC} = \frac{d_{AC}}{d_{AB}+d_{BC}}$. Therefore, $T_{ABC} > 1$ means there exists a TIV for the triple (A, B, C). We see that DNS3997 and Host479 both contain a fraction of TIVs. Besides, Host479 causes more severe TIVs than DNS3997 since asymmetric latencies cause more triples to have TIVs.

TIVs occur due to systematic effects such as the suboptimal Internet routing policies and have been recognized as a fundamental property of the Internet delay space [44,45,21,16]. Moreover, the end-to-end RTT measurements are also asymmetric (e.g., the Host479 data set), because of the fact that end hosts do not coordinate their measurements can lead to persistent asymmetric RTT samples.

To put the design of DNNL schemes on a theoretical foundation whose assumptions faithfully model the characteristics of the Internet delay space, we consider a general delay model that allows for TIV and asymmetry to hold. For this purpose, we extend the inframetric model [17], which generalizes the notion of the metric space.

We directly introduce the extended inframetric model.[1] In order to account for asymmetric RTTs, we define the inframetric for a triple (A, B, C) as the ratio between the RTT value d_{AB} and the maximum of that between node A and C and that between node B and C in condition (2) as follows:

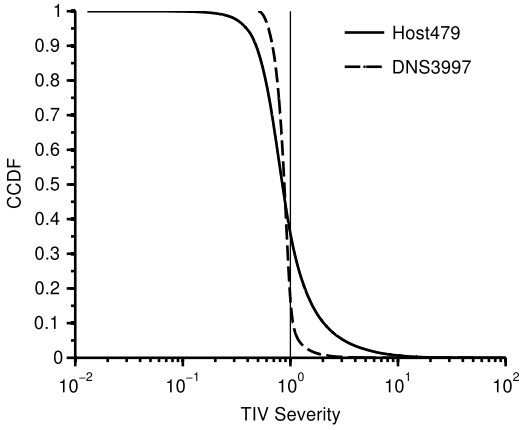


Fig. 3. The complementary cumulative distribution function (CCDF) of the TIV for the data sets.

Definition 1. Let a distance function $d : V \times V \to \Re^+$ denote the pairwise RTT values between nodes in V. For two distinct nodes $P, Q \in V$, $d(P, Q)$ or d_{PQ} means the delay from node P to node Q and back to node P. d is called a ρ-inframetric ($\rho > 1$), if d satisfies the following conditions:

(1) For any pair of nodes P_1 and P_2, where $P_1, P_2 \in V$, $d(P_1, P_2) = 0$, then $P_1 = P_2$;
(2) For any triple (P_1, P_2, P_3), where $P_1, P_2, P_3 \in V$,

$$\begin{aligned} d(P_1, P_2) \le \rho \max \{ & \max \{d(P_1, P_3), d(P_3, P_2)\}, \\ & \max \{d(P_1, P_3), d(P_2, P_3)\}, \\ & \max \{d(P_3, P_1), d(P_3, P_2)\}, \\ & \max \{d(P_3, P_1), d(P_2, P_3)\}\} \end{aligned} \quad (1)$$

hold.

The second condition of Definition 1 is defined for each triple (P_1, P_2, P_3). When some RTT values are asymmetric between nodes P_1, P_2 and P_3, the minimum ρ values for triples (P_1, P_2, P_3), (P_3, P_2, P_1) or (P_1, P_3, P_2) may differ significantly.

Besides, by appropriately choosing the constant ρ, the above relaxed inframetric model admits for TIVs to occur.

For each triple (i, j, k), we compute the minimal ρ that satisfies the second constraint of the inframetric:

$$\rho_{\langle i,j,k\rangle} \leftarrow \max \left\{ \frac{d_{ij}}{\max\{d_{ik}, d_{kj}\}}, \frac{d_{ij}}{\max\{d_{ik}, d_{jk}\}}, \frac{d_{ij}}{\max\{d_{ki}, d_{kj}\}}, \frac{d_{ij}}{\max\{d_{ki}, d_{jk}\}} \right\}. \quad (2)$$

Table 2 summarizes the statistics of ρ values for all data sets. While ρ is low for most triples, there also exists a small fraction of triples with high ρ values. Among the triples whose ρ values are larger than 2, their ρ values are around 3 on average but may take values around 5 and higher for a small fraction of the triples. The Host479 data set shows higher ρ values than other data sets, which may be caused by the delay aggregations.

Therefore, selecting $\rho = 3$ is reasonable to model most of the triples. Consequently, it is possible to choose a low inframetric parameter ρ.

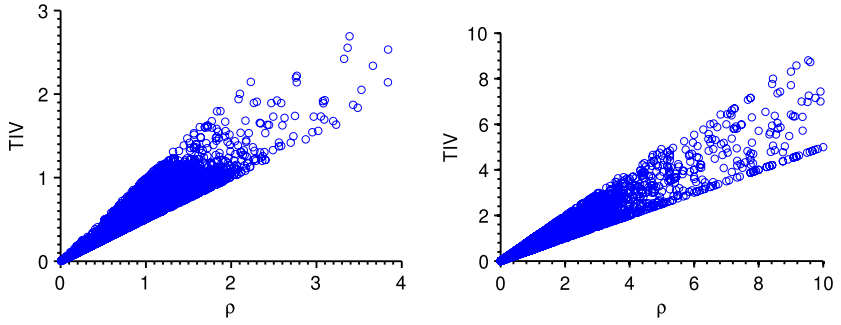
We next study the relation between the inframetric parameter ρ values and the TIV severity of triples in the data sets. Fig. 4 shows the scatter plots of ρ values versus TIV severity. The TIV values generally increase with increasing ρ values, but the correlation between the TIV values and the ρ values becomes weaker with increasing TIV values. This is because a larger TIV value means that one RTT value x is bigger than the sum of the other two RTT values y and z, but the ρ value that equals the ratio between x and the maximum of y and z may not increase significantly.

[1] The original definition in [17] also required symmetry, i.e. $d(P_1, P_2) = d(P_2, P_1)$; since delay in the real Internet can be asymmetric, we have dropped the symmetry requirement.

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

Table 2
The inframetric parameter ρ statistics. To differentiate ρ from the average case against the extreme case, we calculate the inframetric $\rho_{\langle i,j,k\rangle}$ for all available triples, as well as those triples whose $\rho_{\langle i,j,k\rangle}$ are over 2.

	$P\ (\rho > 2)$	$P\ (\rho > 3)$	$\rho_{\langle i,j,k\rangle}$				$\rho_{\langle i,j,k\rangle} > 2$			
			Mean	pct50	pct5	pct95	Mean	pct50	pct5	pct95
DNS3997	0.03	0.01	0.83	0.83	0.17	1.57	2.64	2.23	2.01	4.42
Host479	0.32	0.23	3.32	1.08	0.09	13.00	8.65	4.69	2.13	28.37



(a) DNS3997. (b) Host479.

Fig. 4. ρ-by-TIV-severity scatter plots.

3.5. Stable RTTs lead to static inframetric model

We assume that the pairwise RTTs are reasonably stable in time and analyze the ρ-inframetric properties of the stable RTTs. Several empirical study have confirmed that RTT values remain constant over time periods in the order of minutes [46,30,29]. For example, the iPlane project [30] has shown that the all-pair RTTs can be well approximated with measurements updated every few hours.

3.6. Growth metric in the inframetric model

In the following we introduce some more definitions that we need. Let V be the whole set of nodes. Let S_V be the set of service nodes. Let $B_P(r)$ be a *closed ball* in S_V *covering* the set of service nodes defined by:

$$B_P(r) = \{Q | d(P, Q) \leq r, P, Q \in S_V\} \tag{3}$$

where P denotes the *center* and r denotes the *radius*. As a result, a node Q is covered by the ball $B_P(r)$ if and only if the RTT from node P to node Q is equal to or smaller than r. The *cardinality* $|B_P(r)|$ of a closed ball $B_P(r)$ is the number of nodes covered by that ball.

The growth dimension represents the ratio between the number of nodes covered by two closed balls with the same center and different radii [17,35]:

Definition 2 (*Growth [17]*)**.** Given a ρ-inframetric model, for any $r \in \Re^+$, $\gamma_\rho \in \Re^+$ and $P \in S_V$, if $|B_P(\rho r)| \leq \gamma_\rho |B_P(r)|$, the ρ-inframetric model is said to have a growth $\gamma_\rho \geq 1$.

Moreover, the doubling metric [17] can also be defined in a way similar to the growth metric. However, we will only use the growth metric for brevity. More information can be found in the online technical report [36].

A low growth value γ_ρ means that the number of nodes covered by the closed ball $B_P(\rho r)$ is comparable to the number of nodes covered by $B_P(r)$ of smaller radius. As we expand the radius of a closed ball centered at a node $P \in S_V$, new nodes in S_V "come into view" at a constant rate [35]. This implies that each node P can find a node that is closer to any other node than node P by uniformly sampling a modest number of nodes [35].

As a result, based on the above uniform sampling process, we can recursively find nodes closer to a target node. However, the design of Meridian [13] assumed the triangle inequality to hold. Since we consider that the delay space of the Internet is more correctly described as an inframetric, we need to redesign our DNNL scheme for the inframetric model.

For any node P, we compute the growth by determining the ratio of the cardinality between the ball $B_P(\rho r)$ and the ball $B_P(r)$ for a variable r. As a large ρ can lead to significant imbalance between $|B_P(\rho r)|$ and $|B_P(r)|$, which will not reflect the fine-granularity relations of balls of different radii, so we choose ρ as 3 to compute the growth.

Fig. 5 shows the median and 90th percentile growth values for varying radii. The median growth of most data sets is relatively small, and declines quickly with increasing radii for most data sets except for Host479. For Host479, the median growth may increase as the radii increase. On the other hand, the 90th percentile growth shows divergent dynamics for different data sets, revealing "M"-shape dynamics, indicating that a small fraction of growth values may increase or decrease with increasing radii.

The changes in γ_ρ as a function of the radius are correlated with the clustering structure of the delay space. When the radius is small, all nodes included in the closed balls $B_P(\rho r)$ and $B_P(r)$ are in the same cluster. Since the density of the cluster is large, $|B_P(\rho r)|$ is much larger than $|B_P(r)|$. On the other hand, with increasing radius, the cardinality differences between $B_P(\rho r)$ and $B_P(r)$ become smaller, since the number of nodes covered in $B_P(r)$ becomes large enough to cover nearly one or more clusters and since the number of clusters is small. Therefore, the shape of the growth may form multiple "M" as the radius increases.

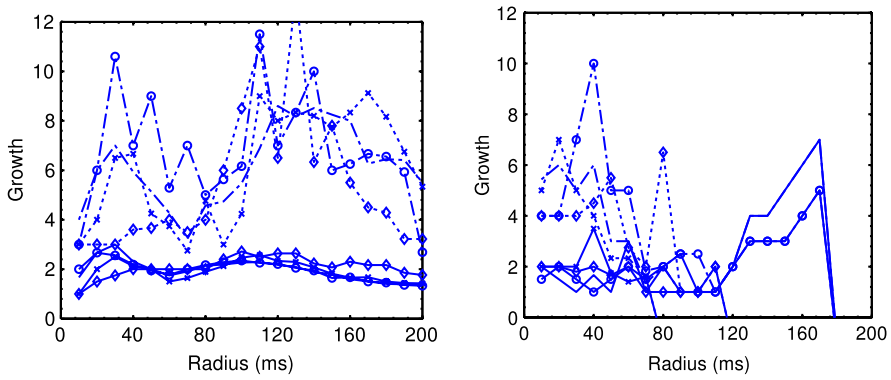
4. DNNL based on the inframetric model

In this section, we prove that it is feasible to design an accurate and fast DNNL algorithm for the inframetric model.

Without loss of generality, assume that each DNNL step needs to locate another node that is β ($\beta \in (0, 1]$) times closer to the target, in order to optimize the delays between the service nodes found and the target.

Definition 3. A DNNL is a system where at each step, a node P_i locates a node P_{i+1} that is β times closer to a target T, which implies that $d_{P_{i+1}T} \leq \beta \times d_{P_iT}$.

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002



(a) DNS3997. (b) Host479.

Fig. 5. The statistics of the median and 90-th percentile growth γ_ρ for $\rho = 3$; – ◇ – denotes median values computed from sampled 20% nodes; – × – denotes median values computed from sampled 50% nodes; –o– denotes median values computed from sampled 75% nodes; - represents median values computed from all nodes; · · · ◇ · · · denotes 90-percentile values computed from sampled 20% nodes; · · · x · · · denotes 90-percentile values computed from sampled 50% nodes; –.o–. denotes 90-percentile values computed from sampled 75% nodes; –. represents 90-percentile values computed from all nodes.

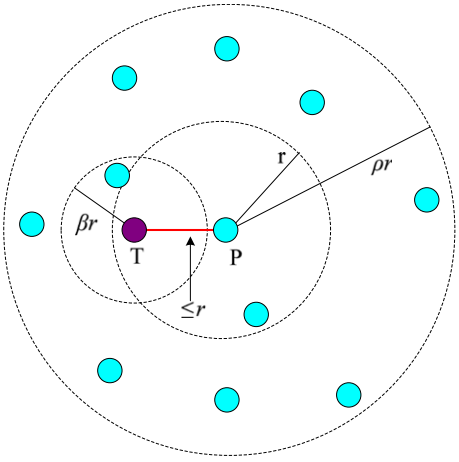


Fig. 6. Sampling nodes closer to a target T from $B_P(\rho r)$ in the ρ-inframetric model with growth γ_ρ.

4.1. How to locate nodes closer to the target

Without loss of generality, assume that a node P needs to locate a node Q that is at least β ($\beta \leq 1$) times closer to a target T, which implies that $d_{QT} \leq \beta \times d_{PT}$. Let $d_{PT} = r$. We can see that node Q must be covered by the ball $B_P(\rho r)$, since $d_{PQ} \leq \rho \max\{d_{PT}, d_{TQ}\} = \rho r$ by the definition of the inframetric model. Fig. 6 shows an example of sampling a node closer to the target T in the closed ball $B_P(\rho r)$.

4.2. Random sampling condition

We analyze the number of samples required to locate a node closer to the target than the current node with high probability (w.h.p).[2] The sampling condition gives the bandwidth cost at each search step.

We first quantify the cardinality differences of balls with identical centers but different radii in the following lemma.

Lemma 4.1. *Given a ρ-inframetric with growth $\gamma_\rho \geq 1$, for any $x \geq \rho$, $r > 0$ and any node P, the cardinality of a ball $B_P(r)$ is at most x^α times smaller than that of the ball $B_P(xr)$, where $\log_\rho \gamma_\rho \leq \alpha \leq 2\log_\rho \gamma_\rho$.*

Proof. First, by recursively calling $\lceil \log_\rho x \rceil$ times the growth definition, until $\frac{x}{\rho^{\lceil \log_\rho x \rceil}} < 1$, we have

$$|B_P(xr)| \leq \gamma_\rho^{\lceil \log_\rho x \rceil} |B_P(r)| = x^\alpha |B_P(r)|, \quad \alpha = \log_x \gamma_\rho \times \lceil \log_\rho x \rceil.$$

Since $x \geq \rho$, $\gamma_\rho > 1$, we can calculate the lower bound of α as:

$$\alpha \geq \log_x \gamma_\rho \times \log_\rho x = \log_\rho \gamma_\rho$$

and the upper bound of α as:

$$\alpha \leq \log_x \gamma_\rho \times (\log_\rho x + 1) = \log_\rho \gamma_\rho + \log_x \gamma_\rho \leq 2\log_\rho \gamma_\rho$$

this concludes the proof. □

Lemma 4.1 states that the cardinality differences of the balls with identical centers and different radii are bounded by x^α, where x is the multiplicative ratio between different radii, and the parameter α lies in a bounded interval.

Fig. 7 shows the values of α as function of the radius r for different multiplicative ratios x. We see that α is mostly below 3 and decreases quickly with increasing values of r or x.

We next show the inclusion relation of balls with different centers, which generalizes the inclusion of balls around a node pair in the *metric space* [35].

Lemma 4.2. *For any pair of nodes p and q, and $d_{pq} \leq r$, then*

$$B_q(r) \subseteq B_p(\rho r).$$

Proof. For any node i in the ball $B_q(r)$, node i satisfies

$$d_{qi} \leq r. \tag{4}$$

By Definition 1, it follows that

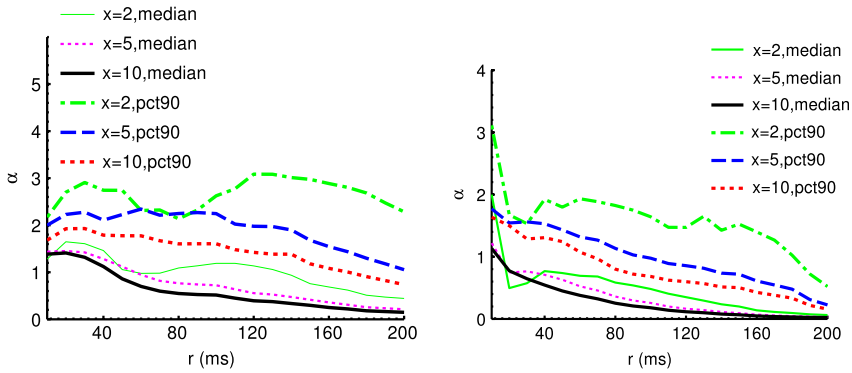
$$d_{pi} \overset{\text{(Eq. (1))}}{\leq} \rho \max\{d_{pq}, d_{qi}\} \leq \rho \max\{r, d_{qi}\} \overset{\text{(Eq. (4))}}{\leq} \rho r \tag{5}$$

which implies that $i \in B_p(\rho r)$, i.e.,

$$B_q(r) \subseteq B_p(\rho r). \quad \square$$

The following theorem tells us how many neighbor nodes a node P needs to sample to find with high probability at least one node lies in the closed ball $B_T(\beta r)$.

[2] An event occurs with high probability if the event occurs with probability at least $1 - N^{-c}$, where N is quite large and $c > 1$ is a positive constant.



(a) DNS3997. (b) Host479.

Fig. 7. α as function of the radius r with varying multiplicative ratio x. We plot the median and 90-th percentile values of α.

Theorem 4.3 (*Sampling Efficiency in the Growth Dimension*)**.** *Given a ρ-inframetric d with growth $\gamma_\rho \geq 1$, a node P and a DNNL target T satisfying $d_{PT} \leq r$. For any $\beta \in (0, 1]$, let N denote the number of servers, and $c > 1$.*

If P selects $O(\ln N)$ nodes uniformly at random with replacement from $B_P(\rho r)$, then with a probability larger or equal than $1 - N^{-c}$ one of sampled nodes will lie in $B_T(\beta r)$.

Proof. From Lemma 4.2, we know that $B_T(\beta r) \subset B_T(r) \subseteq B_P(\rho r)$, and all nodes covered by $B_T(\beta r)$ are also covered by $B_P(\rho r)$. Therefore, we only need to sample enough nodes in $B_P(\rho r)$ in order to sample a node located in $B_T(\beta r)$.

Furthermore, for the pair of nodes P and T satisfying $d_{PT} \leq r$, it follows

$$|B_P(\rho r)| \leq \left|B_T\left(\rho^2 r\right)\right| = \left|B_T\left(\frac{\rho^2}{\beta}\beta r\right)\right|.$$

Since $\rho > 1$ and $\beta \leq 1$, then $\frac{\rho^2}{\beta} \geq \rho^2 \geq \rho$, the preconditions of Lemma 4.1 hold and we can compute the relation between the ball $B_P(\rho r)$ and the ball $B_T(\beta r)$:

$$|B_P(\rho r)| \leq \left|B_T\left(\frac{\rho^2}{\beta}\beta r\right)\right| \leq \left(\frac{\rho^2}{\beta}\right)^{\alpha} |B_T(\beta r)|$$

where $\log_\rho \gamma_\rho \leq \alpha \leq 2\log_\rho \gamma_\rho$. Therefore, the probability of uniformly sampling a node from $B_P(\rho r)$ that lies in the ball $B_T(\beta r)$ is:

$$\frac{|B_T(\beta r)|}{|B_P(\rho r)|} \geq \frac{|B_T(\beta r)|}{\left(\frac{\rho^2}{\beta}\right)^{\alpha} |B_T(\beta r)|} = \frac{1}{\left(\frac{\rho^2}{\beta}\right)^{\alpha}}.$$

Let $\gamma = (\rho^2/\beta)$. Let the number of samples be l. The probability of failing to sample a node in the ball $B_T(\beta r)$ is at most

$$(1 - 1/\gamma^{\alpha})^l.$$

In order to obtain the failure probability to be within N^{-c}, where $c > 1$, i.e., $(1 - 1/\gamma^{\alpha})^l \leq N^{-c}$, the number l of samples must be at least

$$\begin{aligned} l &= -\frac{c}{\ln(1 - 1/\gamma^{\alpha})} \ln N \\ &= -\frac{c}{\ln\left(1 - (\beta/\rho^2)^{\alpha}\right)} \ln N \\ &= O(\ln N). \end{aligned}$$

As a result, with a probability of at least $(1 - N^{-c})$, the current node is able to locate a neighbor that lands in the ball $B_T(\beta r)$. □

4.3. DNNL on the inframetric model

Based on the sampling condition of Theorem 4.3, DNNL should operate as follows (see Algorithm 2):

Let $r = d_{PT}$; sample $O(\ln N)$ neighbors from the closed ball $B_P(\rho r)$ at each intermediate node P; forward the DNNL request to a next-hop node that is at least β times closer to the target than the node P; stop when we cannot find such a next-hop node.

Algorithm 2: The theoretical DNNL algorithm.

```
1 DNNL(P, T)
  input  : current node P, target T
  output: Nearest node to T
2 r ← d_PT;
3 P selects O(ln N) neighbors S_P from the closed ball B_P(ρr);
4 A ← closest neighbor to T from S_P ∪ {P};
5 if d_AT ≤ βr then
6     DNNL(A, T);
7 else
8     return A;
9 end
```

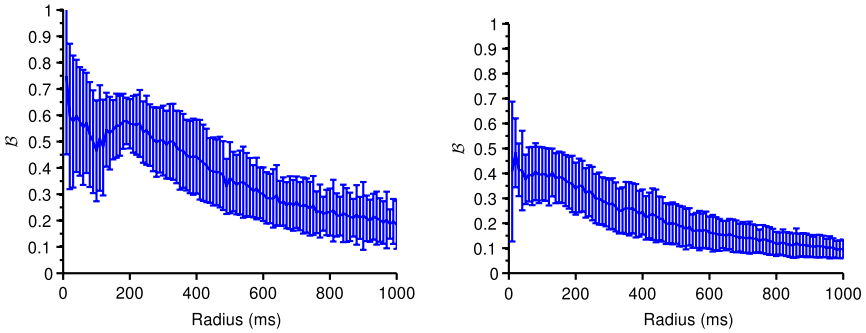
We quantify the closeness between the nearest service node found by Algorithm 2 and the ground-truth nearest service node to the target and the number of search steps needed. For a DNNL request with target T, if the delay between A to T is smaller than ωd_*, the found nearest service node A is a *ω-approximation*, where d_* is the delay between the real nearest service node to T. As a result, the smaller ω, the closer the found node is to the target.

We first analyze the upper bound of the number of hops in Theorem 4.4.

Theorem 4.4. *Algorithm 2 terminates in at most $\log_{\frac{1}{\beta_{\text{real}}}}(\Delta_d)$ steps, where $\beta_{\text{real}} < 1$ denotes the average delay reduction per step and Δ_d is the ratio of the maximum delay to the minimum delay in the delay space.*

Proof. Let l be the number of search steps. Let $d_{\min}$ ($d_{\min} > 0$) be the minimal pairwise delay of all-pair delays over the network delay space. Let the current node be P. Let β_i denote the delay reduction at the i-th step. We can see that $\beta_i \leq \beta$, since we always choose the neighbor that is closest to the target than the current node, the delay reduction by the chosen next-step node may be much better than β. A sufficient condition for terminating the search process after l steps is that:

$$\prod_{i=1}^{l} \beta_i d_{PT} = d_{\min}. \tag{6}$$



(a) DNS3997. (b) Host479.

Fig. 8. The average ratio $\mathcal{B}$ and the confidence interval as a function of the radius of the closed ball $B_T(\beta r)$, where T denotes the target, β denotes the threshold value, r denotes a positive real number.

Moreover, since any delay value does not exceed the maximum delay, we see that $d_{PT} \leq \Delta_d \times d_{\min}$ also holds. As a result, we have

$$\frac{1}{\prod_{i=1}^{l} \beta_i} d_{\min} = d_{PT} \leq \Delta_d \times d_{\min}. \tag{7}$$

Therefore, the number of search steps l is at most

$$\frac{1}{\prod_{i=1}^{l} \beta_i} \leq \Delta_d. \tag{8}$$

Furthermore, let $\beta_{\text{real}} = \frac{\sum_{i=1}^{l} \beta_i}{l}$ be the average delay reduction for each search step. By the inequality of arithmetic and geometric means for nonnegative numbers, we have

$$\beta_{\text{real}}^{l} = \left(\frac{\sum_{i=1}^{l} \beta_i}{l} \right)^{l} \geq \prod_{i=1}^{l} \beta_i. \tag{9}$$

Combining Eqs. (8) and (9), we can see that

$$\frac{1}{\beta_{\text{real}}^{l}} \leq \frac{1}{\prod_{i=1}^{l} \beta_i} \leq \Delta_d. \tag{10}$$

As a result, the number of search steps l is at most

$$l \leq \log_{\beta_{\text{real}}} \left(\frac{1}{\Delta_d} \right) = \log_{\frac{1}{\beta_{\text{real}}}} (\Delta_d). \quad \square \tag{11}$$

During the proof, we assumed the average delay reduction β_{real} is known. We next experimentally show the differences between β_{real} and the threshold β.

For an arbitrary target T, a threshold value β and a positive real number r. We sample the differences between the real-world latencies from the next-hop nodes to the target and the upper bound βr of the latencies required by Algorithm 2. We can see that the smaller the ratios, the larger the differences between β_{real} and β.

Assume that we need to obtain Υ ($\Upsilon = 1000$ by default) samples from the ball $B_T(\beta r)$. The ratio-sample process takes three steps:

1. List the set of nodes in the closed ball $B_T(\beta r)$. Let the number n_s of samples be $n_s = 0$.
2. Sample a node Z uniformly at random from the ball $B_T(\beta r)$.
3. Compute the ratio $\frac{d_{ZT}}{\beta r}$, where d_{ZT} denotes the RTT value from node Z to node T. $n_s = n_s + 1$. If $n_s < \Upsilon$, go to step 2; otherwise, the sampling process stops.

For a given radius r and a delay reduction threshold β, let $\mathcal{B}$ denote the average value for the ratios $\frac{d_{ZT}}{\beta r}$ by using different targets T. We plot $\mathcal{B}$ and the confidence intervals in Fig. 8. The average ratio $\mathcal{B}$ decrease with increasing RTT value r or the latency reduction threshold β, and are below 0.6 in most cases. Therefore, the average latency reduction threshold β_{real} is much smaller than β, which implies that the upper bound of the routing hops $\log_{\frac{1}{\beta_{\text{real}}}} (\Delta_d)$ is much smaller than $\log_{\frac{1}{\beta}} (\Delta_d)$.

We next analyzed the expectation of the average delay reduction per step β_{real}.

Theorem 4.5 (*Expectation of the Average Delay Reduction Per Step*)**.** *The expectation of the average delay reduction β_{real} is smaller than half of the delay reduction threshold β:*

$$E[\beta_{\text{real}}] < \frac{1}{2}\beta. \tag{12}$$

Proof. Let d_T be the minimum RTT from nodes in V to the target T. Assume that the search path has L hops. Let node P be the first node that receives the DNNL request. Let r be the RTT from node P to the target T. Let r_i be the RTT value from the node at the i-th hop to the target, where $i \in [1, L]$. Let $r_0 = r$.

The delay reduction β_i at the i-th hop is lower bounded by $\frac{d_T}{r_{i-1}}$, where $i \in [1, L]$, and is upper bounded by β. As a result, we now write the expectation of the delay reduction at the i-th hop as the integral:

$$\begin{aligned} E[\beta_i] &= \int_{\tau=\frac{d_T}{\beta r_{i-1}}}^{1} (\tau\beta) \Pr[\beta_i = \tau\beta] \, d\tau \\ &< \int_{\tau=\frac{d_T}{\beta r_{i-1}}}^{1} (\tau\beta) \, d\tau \\ &= \beta \int_{\tau=\frac{d_T}{\beta r_{i-1}}}^{1} \tau \, d\tau \\ &= \beta \times \frac{1 - \left(\frac{d_T}{\beta r_{i-1}}\right)^2}{2} \\ &= \frac{1}{2}\beta \left(1 - \left(\frac{d_T}{\beta r_{i-1}}\right)^2\right). \end{aligned}$$

Therefore, the expectation of the average delay reduction value can be calculated as follows:

$$\begin{aligned} E[\beta_{\text{real}}] &= \frac{1}{L}\sum_{i=1}^{L} E[\beta_i] \\ &< \frac{1}{L}\sum_{i=1}^{L}\left(\frac{1}{2}\beta\left(1-\left(\frac{d_T}{\beta r_{i-1}}\right)^2\right)\right) \\ &= \frac{1}{2}\beta\frac{1}{L}\sum_{i=1}^{L}\left(1-\left(\frac{d_T}{\beta r_{i-1}}\right)^2\right) \\ &< \frac{1}{2}\beta. \quad \square \end{aligned}$$

By Theorem 4.4, the search steps of Algorithm 2 is bounded by $\log_{\frac{1}{\beta_{\text{real}}}} \Delta_d$ for any β_{real}, we further theoretically confirm that the search steps is therefore logarithmically related to the ratio Δ_d.

We finally establish the accuracy of the search results in Theorem 4.6.

Theorem 4.6. *Given a target node T, Algorithm 2 has an $\frac{1}{\beta}$-approximation with high probability $1 - N^{-c_2}$, where N denotes the number of servers and $c_2 > 1$.*

Proof. Suppose that P_* is the ground-truth nearest server to the target T. Suppose that a node P forwards the DNNL request to another node Q by Algorithm 2, the *progress* of the DNNL process is calculated as the ratio of $\frac{d_{PT}}{d_{QT}}$, which is at least $\frac{1}{\beta}$ by Theorem 4.3.

First, let p be the probability of finding a neighbor Q that is β times closer to the target at a step. Based on the sampling conditions of Theorem 4.3, $p \geq 1 - N^{-c}$. As a result, the failure probability of l steps is at most

$$\begin{aligned} 1 - p^l &= \left(1 - \left(1 - N^{-c}\right)^l\right) \\ &\approx 1 - e^{-l/N^c} \\ &\approx 1 - \left(1 - l/N^c\right) \\ &\approx \left(N^{-c_2}\right) \end{aligned} \tag{13}$$

due to the Taylor's expansion, where $c_2 = c - \log_N l > 1$ since $l \ll N$ by Theorem 4.4. As a result, the probability of finding a neighbor satisfying the sampling condition in Theorem 4.3 after l steps is at least $1 - N^{-c_2}$, i.e., with high probability.

Second, assume that Algorithm 2 locates a node P_x as the nearest server and has an approximation ratio larger than $\frac{1}{\beta}$ i.e., $d_{P_xT} > \frac{1}{\beta} d_{P_*T}$. We disprove the approximation ratio by contradiction.

Since $\beta \leq 1$, we see that $d_{P_xT} > d_{P_*T}$. As a result, we can locate a new node β times closer to the target than P_x with high probability. As a result, the search process can be continued, which contradicts the fact that the search process stops at node P_x. Therefore, the approximation ratio of the found node must be at most $\frac{1}{\beta}$, which completes the proof. $\square$

4.4. Making the theoretical DNNL algorithm more efficient

Based on our theoretical results we propose several adjustments to make the DNNL algorithm more *efficient*. We can reduce the cost in two complementary ways.

Make β large: The delay reduction threshold β determines when to terminate a DNNL query. Theorem 4.6 shows that setting a higher β value improves the search accuracy. Moreover, setting $\beta < 1$ can lead to local minima caused by the clustering in the delay space. In order to avoid such local minima, we set β to be 1 to continue the DNNL process.

Error-aware hybrid delay measurements: We can use network coordinates to avoid active delay measurements from the sample nodes to the target. We predict delay based on the revised Vivaldi algorithm [21] *TIVVivaldi*$(x_i, e_i, d_{ij}, x_j, e_j)$, where the inputs x_i and x_j denote the coordinates of node i and j, respectively; the inputs e_i and e_j denote the averaged error of node i's and j's coordinates, respectively. The output of *TIVVivaldi* is the updated coordinate x_i and coordinate error e_i of node i.

Using delay estimations alone to find the nearest service nodes may not be reliable, since the delay estimation incurs some inaccuracy due to the embedding distortions of network coordinates. However, the Vivaldi algorithm measures the inaccuracy of a coordinate with the error variable e_i. When e_i or e_j exceeds a threshold, the DNNL algorithm will use active delay probes instead.

4.5. Sample enough neighbors

For the DNNL service to perform well and find the nearest service node to any target, the algorithm must maintain enough neighbors covering different delay ranges in the delay space and each node has to maximize its diversity in the neighbor set.

Our theoretical results need to sample sufficient neighbors from a closed ball centered at the current node P. Unfortunately, since we sample from the whole set of nodes, the samples can be outside the closed ball. As a result, finding a sample in the closed ball may take multiple samples. Moreover, maintaining a closed ball for each DNNL request is both time- and bandwidth-consuming. As a result, we need a light-weight scheme to realize the ball-based neighbor selection process.

4.5.1. Intuitions

Researchers have observed that the delay space contains macroscopic clusters, e.g., Europe, Asia and America [40]. As a result, wide-area nodes are grouped into a small number of clusters in the network delay space and nearby nodes are in the same cluster. If we would like to uniformly sample nodes from a closed ball centered at a node P, we have to sample more nodes in the same cluster as node P.

On the other hand, the concentric rings used by Meridian [13] and OASIS [14] represent the state-of-art data structure to store neighbors. The concentric ring is organized with rings of exponentially increasing radii and each ring contains a number of uniformly sampled nodes that fall into the delay ranges covered by that ring. We can see that the concentric ring for a node P biases in favor of nodes that are close to node P. As a result, selecting nodes from the concentric ring selects more nodes from the same cluster with the current node, which is consistent with the ball-based uniform sampling process.

Moreover, since the exponentially increasing radii focus on inner rings, the concentric ring can cover intra-cluster nodes in multiple inner rings, which helps locate closer nodes with fine granularity.

4.5.2. Search accuracy for the ring based neighbor sampling

Let B_{ui} be the ball $B_u\left(2^i\right)$. Let $S_{ui} = B_{ui} \setminus B_{(u,i-1)}$ be the i-th ring in the concentric ring. Assume that each ring contains $O(\ln N)$ nodes and the nodes at each ring is uniformly sampled from the whole set of nodes that fall into that ring.

For a target T, we prove how many neighbors P needs to sample for a ring to find at least one node that lies in the closed ball $B_T(\beta r)$ w.h.p.

Theorem 4.7 (*Sampling Efficiency in Concentric Rings*). *Given a ρ-inframetric d with growth $\gamma_\rho \geq 1$, a node P, and a DNNL target T. Let $r = d_{PT}$. The size of each ring is $O(\log(N))$. There exists a ring whose number is in $[1, \lceil \log_2(\rho r)\rceil]$ satisfying that selecting all neighbors on that ring will find one node covered by $B_T(\beta r)$ with at least a probability $(1 - N^{-c})$, where N denotes the number of servers, $c > 1$.*

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

Proof. Let

$$r_* = \beta r. \tag{14}$$

We select the minimum positive integer i such that

$$\rho \max\{r, r_*\} = \rho r \le 2^i \tag{15}$$

holds. As a result, the inequality

$$\rho r > 2^{i-1} \tag{16}$$

also holds, because otherwise, $(i-1)$ will become the minimum integer satisfying Eq. (15). Besides, we can see that $i = \lceil \log_2(\rho r) \rceil$.

For a node j from the ball $B_T(r_*)$, i.e.,

$$d_{Tj} \le r_*. \tag{17}$$

By Definition 1, we know that

$$d_{Pj} \le \rho \max\left\{d_{PT}, d_{Tj}\right\} \le \rho \max\{r, r_*\} = \rho r \overset{\text{Eq. (15)}}{\le} 2^i. \tag{18}$$

Therefore, node j is covered by the ball B_{Pi}. As a result, the ball $B_T(r_*)$ is covered by B_{Pi}, i.e.,

$$B_T(r_*) \subseteq B_{Pi}. \tag{19}$$

In other words, in order to obtain a sample from the ball $B_T(r_*)$, we only need to select sufficient nodes from the ball B_{Pi}.

(1) By multiplying two at both sides of Eq. (16), we have

$$2^i < 2\rho r. \tag{20}$$

Therefore, by multiplying ρ at both sides of Eq. (20), it follows that

$$B_T\left(\rho 2^i\right) \subset B_T(\rho(2\rho r)). \tag{21}$$

Moreover, for any node $j \in B_{Pi}$, we know that

$$d_{Tj} \le \rho \max\left\{d_{PT}, d_{Pj}\right\} \le \rho \max\left\{r, 2^i\right\} \overset{\text{Eq. (15)}}{=} \rho 2^i \tag{22}$$

by Eq. (1) from Definition 1. As a result, the ball B_{Pi} is covered by the ball $B_T\left(\rho 2^i\right)$:

$$B_{Pi} \subseteq B_T\left(\rho 2^i\right). \tag{23}$$

Combining Eqs. (21) and (23), we know that B_{Pi} is covered by $B_T(\rho(2\rho r))$:

$$B_{Pi} \subset B_T(\rho(2\rho r)). \tag{24}$$

Since $r = \frac{r_*}{\beta}$ based on Eq. (14), Eq. (24) can be transformed to be:

$$B_{Pi} \subset B_T((2\rho^2/\beta) r_*). \tag{25}$$

By the definition of the growth metric, we calculate the cardinality difference between the ball B_{Pi} and $B_T((2\rho^2/\beta)r_*)$ as follows:

$$|B_{Pi}| < \left(2\rho^2/\beta\right)^\alpha |B_T(r_*)|. \tag{26}$$

As a result, the probability of uniformly sampling a node from B_{Pi} that lies in the ball $B_T(r_*)$ is:

$$\frac{|B_T(r_*)|}{|B_{Pi}|} > \frac{|B_T(r_*)|}{\left(2\rho^2/\beta\right)^\alpha |B_T(r_*)|} = \frac{1}{\left(2\rho^2/\beta\right)^\alpha}. \tag{27}$$

(2) Suppose that the size of the ring is

$$\begin{aligned}&\left(2\rho^2/\beta\right)^\alpha \log\left(N/N^{-c}\right)\\ &= (c+1)(\rho/\beta)^\alpha \log(N)\\ &= O(\log(N)).\end{aligned}$$

By Theorem 4.1 in [13], we can see that some node from a ring S_{Pl}, $l \le i$ lands in the ball $B_T(r_*)$ with a failure probability $p < \left(N^{-c}\right)/N^2 < N^{-c}$. The proof is complete. □

As a result, by selecting neighbors on the rings numbered from 1 to $\lceil \log_2(\rho d_{PT}) \rceil$, we can recursively determine an approximately optimal nearest server to the target T similar to Theorems 4.4 and 4.6.

Corollary 1. *Given a target node T, by recursively selecting neighbors on the rings numbered from 1 to $\lceil \log_2(\rho d_{PT}) \rceil$, the search process has an $\frac{1}{\beta}$-approximation with a probability $1 - N^{-c_2}$, where N denotes the number of servers and $c_2 > 1$. Moreover, the search process terminates in at most $\log_{\frac{1}{\beta_{\text{real}}}} \Delta_d$ steps, where $\beta_{\text{real}} < 1$ denotes the average delay reduction per step and Δ_d is the ratio of the maximum delay to the minimum delay in the delay space.*

Moreover, we empirically found that setting a modest number of neighbors Δ is sufficient to guarantee the search accuracy (8 by default).

4.5.3. Concentric ring based neighbor selection

The ring structure partitions the delay space into annuli with exponentially increasing space. As a result, different annuli contain non-uniform number of nodes. Assume that we know the complete delay matrix, we configure the concentric ring for each node based on the delay matrix: we fill a ring with all nodes that fall into the delay interval of that ring. We then compute the percentage of nodes falling into each ring for each node. From Fig. 9, we see that a few of rings, whose delay range is in the middle portion of the delay distribution, contain most of the nodes. However, only very few nodes are mapped into the innermost and outermost rings, since most RTTs are concentrated in tens or hundreds of milliseconds.

As a result, uniformly Sampling based neighbor discovery for the concentric ring is insufficient, since the number of samples in a ring will be approximately proportional to the percentage of nodes that fall into that ring, but nodes in the rings are non-uniformly distributed as in Fig. 9. As a result, the gossip process will inevitably sample an insufficient number of neighbors for the innermost and outermost rings.

In order to maintain a small number of samples on each ring for scalability, we propose to combine the gossip based uniform sampling and *biased sampling* to select enough neighbors for each rings. For the middle portions of rings, uniform sampling is sufficient; while for the inner and outer portions of rings, we have to explicitly sample enough nodes that fall into these rings.

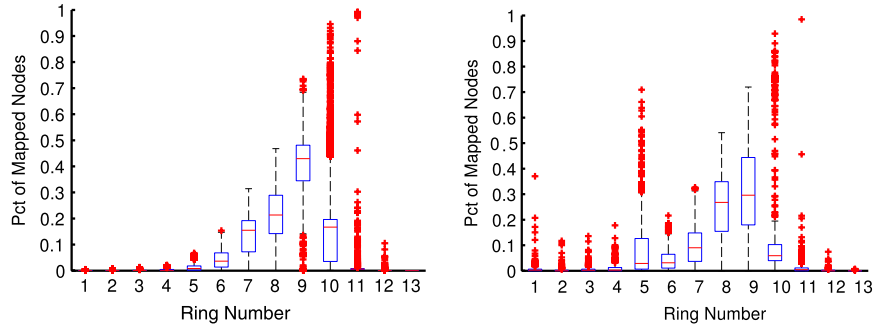
We also note that, setting the number i^* of rings to a small integer, e.g., 9 rings in the Meridian protocol, does not avoid the weakness of the gossip based uniform sampling, since the inner most rings still have too few samples due to the skewed distributions of the percentage of nodes on each ring. Only the outer rings may have more sampled neighbors since the percentage of nodes on these outer rings increases as shown from Fig. 9. However, when the RTT value is quite large, selecting a small i^* will not select enough rings to sample candidate neighbors.

5. Details of HybridNN

We are now ready to present a novel DNNL scheme called *HybridNN* (*Hybrid Nearest* Service *N*ode Search) that applies all the design and optimization principles discussed so far. HybridNN is a distributed recursive search algorithm that terminates the search process when the nearest server to the target is found.

Fig. 10 shows the main steps of HybridNN. Suppose that a node P receives a DNNL request to a target T. Node P first samples candidate neighbors from its concentric ring that are possibly close to the target. Then node P selects the nearest candidate to T based on delay predictions and a small number of direct delay probes. Furthermore, HybridNN also computes a coordinate for the target in order to predict delay to the target. Finally, after finding the node P_* that is closest to the target, node P next determines whether to terminate the DNNL process or forward the request to P_*.

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002



(a) DNS3997. (b) Host479.

Fig. 9. Box plots of the percent of nodes mapped into each ring for each node. We bin all the delay values on the data sets into the concentric ring. The i-th ring contains neighbors whose delay to a node P lie in the interval $(2^{i-1}, 2^i]$.

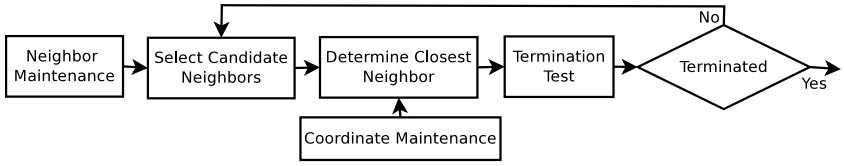


Fig. 10. Finding a closer neighbor to the target using HybridNN at a service node.

5.1. Neighbor maintenance

We first introduce the neighbor management process that includes the neighbor discovery and update procedures.

We use delay estimations to approximate the pairwise delay, in order to reduce the delay measurement cost. Each service node passively maintains its coordinate, by reusing the delay measurements to other service nodes during the neighbor discovery process. Furthermore, each service node also stores its neighbors' coordinates in order to estimate pairwise delay between neighbors in the concentric ring.

5.1.1. Neighbor discovery

As discussed in Section 4.5, we need to sample enough neighbors for each ring. Gossip is a high scalable method for sampling peers for P2P applications [13,47,34,48,49]. We propose to combine the gossip based uniform sampling and a random-walk based bias sampling process to discover new neighbors for the concentric ring. Both sampling processes are quite simple to implement and incur modest communication overhead.

Gossip based neighbor sampling. Our gossip protocol is similar to that used in Meridian. Each node P periodically triggers a gossip event. When the event is triggered, node P selects a node Q from each of its rings as the gossip peers. Second, node P sends a gossip request message to each gossip peer Q. Third, when node Q receives the gossip request message, node Q answers node P a *gossip ACK message* whose payload consists of one node per non-empty ring from Q's concentric ring. Finally, after node P receives the ACK message, node P selects the cached nodes in the message as the candidate neighbors. Then node Q iteratively probes delay to these candidate neighbors and inserts those with successful delay probes into node Q's concentric ring.

Random-walk based neighbor sampling. Each node samples nearby and far-away nodes into its concentric ring by random walks. The random-walk message has a Time to Live (TTL) field indicating its liveness period. The TTL is initialized to a positive integer (20 by default) and decreased by one at each intermediate step. Each node P runs the random-walk sampling procedure. Key steps are as follows:

1. Node P first samples a neighbor Q as the gossip peer. Then node P stores its network coordinate and the TTL value into a message *msg* and sends it to node Q.
2. After node Q receives *msg*, node Q selects K nearest service nodes and K farthest neighbors from Q's concentric ring to node P based on the delay predictions. Then node Q stores the optimal K nearer and K farther neighbors into *msg*, and decrease the TTL value of *msg* by 1. Then node Q sends *msg* to a node selected from a random ring of Q's concentric ring.
3. Node Q_2 tests whether the TTL value of *msg* is 0: if TTL is 0, node Q_2 sends *msg* back to node P; otherwise, node Q_2 runs a similar process as step 2. To improve the diversity of neighbors, node Q_2 further stores P into its concentric ring.

Finally, node P receives the random-walk message and selects the cached 2 K neighbors as the candidate neighbors.

We can see that the gossip- and random-walk-based sampling processes are complementary to each other. Intuitively, since most overlay links are created between nearby nodes in terms of the network delay and only a small number of overlay links are between remote nodes, the overlay is approximately a "small world", which implies that there always exists a small-hop routing path between any node pair with length in terms of the logarithmical function of the system size [50,51].

For example, Theorem 4.6 has bounded the number of search steps in terms of the logarithmical function of the ratio between the maximum and minimum delay values. We also empirically confirm that the search hops for finding an approximately nearest server to any target terminates in around four hops, which is consistent with the efficient routing on the overlay structure.

5.1.2. Neighbor replacement

When the number of neighbors on a ring exceeds Δ, we remove these additional nodes. The replacement process should maximize the diversity in the neighbor set, which translates to better chances of locating nearby nodes for any target. To do that, we use the maximal hypervolume polytope algorithm [13] that preserves those that maximize the diversity of neighbors in a ring.

However, the maximal hypervolume polytope algorithm requires all-pair delay measurements of nodes in a ring, which needs quadric number of delay probes of the nodes of a ring. In order to avoid these delay measurements for better scalability, we use delay predictions to approximate pairwise delay.

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

Algorithm 3: Pruning the size of candidate neighbors.

```
1  chooseCandidates(P, T, M)
   input  : current node P, target T, DNNL query message M
   output: the set of candidate neighbors S
2  S ← ∅;
3  for each neighbor i satisfying d_iP ≤ ρ d_PT do
4      if i's non-empty ring > τ then
5          S ← S ∪ {i};
6      end
7  end
8  S_f ← M.Path;
   // avoid loops
9  S ← S − S_f;
10 return S;
```

Algorithm 4: Maintaining target's network coordinate.

```
1  InitTargetCoord(P, T, M)
   input  : current node P, target T, DNNL query M
2  if M.init == False then
3      Ω_P ← L neighbors selected uniformly at random from P's
       concentric ring;
4      for i ∈ Ω_P do
5          d_iT ← RTTProbe(i, T);
6          [x_T, e_T] ← TIVVivaldi(x_T, e_T, d_iT, x_i, e_i);
7      end
8      M.init ← True;
9  else
10     x_T ← M.x_T;
11     e_T ← M.e_T;
12     d_PT ← directProbe(P, T);
13     [x_T, e_T] ← TIVVivaldi(x_T, e_T, d_PT, x_P, e_P);
14 end
15 M.x_T ← x_T;
16 M.e_T ← e_T;
```

5.2. Select candidate neighbors

As discussed from Section 4.5.3, each node P selects candidate neighbors from the concentric ring for each DNNL request. Let the delay between node P and the target T be d_{PT}. Node P selects neighbors from the rings numbered from $[1, \lceil \log_2(\rho d_{PT}) \rceil]$.

We also prune neighbors that may mislead the DNNL process. First, candidate neighbors that contain too few non-empty rings are more likely to provide no help on continuing the DNNL query. As a result, the search process is more easily to be trapped into a local minima. Accordingly, we remove neighbors that have fewer than τ non-empty rings ($\tau = 4$ by default). We keep up-to-date ring information by reusing the gossip based neighbor discovery process for efficiency. When a node P starts a new round of gossip communication with a node from each ring, node P asks the gossip peer to piggyback the peer's non-empty ring information to node P. The gossip period serves as a trade off between the updating bandwidth cost and the freshness of the non-empty rings.

Second, we remove all neighbors that have received the identical DNNL query from the set of candidate neighbors, in order to avoid the search loops. The pseudo-code is shown in Algorithm 3.

5.3. Coordinate maintenance for target

We compute a network coordinate for the target in order to predict delay from candidate neighbors to the target for better scalability. Suppose that a node P receives a DNNL request. Node P tests whether the coordinate of the target is initialized: if the request message does not contain the target T's coordinate, node P then initializes T's coordinate; otherwise, node P updates T's coordinate. The coordinate computation is based on the *TIVVivaldi* algorithm from Section 4.4.

To initialize T's coordinate, node P randomly selects L (15 by default) neighbors from its concentric ring. Then node P asks these neighbors to probe delay to the target T and to return the measurements to P. Node P next computes a network coordinate based on these neighbors' coordinates and probed delay to T. Then, node P stores target T's coordinate and coordinate error into the DNNL query and forwards to the next-hop node. Algorithm 4 describes the coordinate maintenance.

5.4. Determine closest neighbor

We determine the closest node to the target from candidates found in Section 5.2 based on delay prediction and several direct probes, since the coordinate distances are only approximations of the actual delay:

- We select the top-m nearest service nodes S_c to the target T from the candidate neighbors based on the coordinate distances.

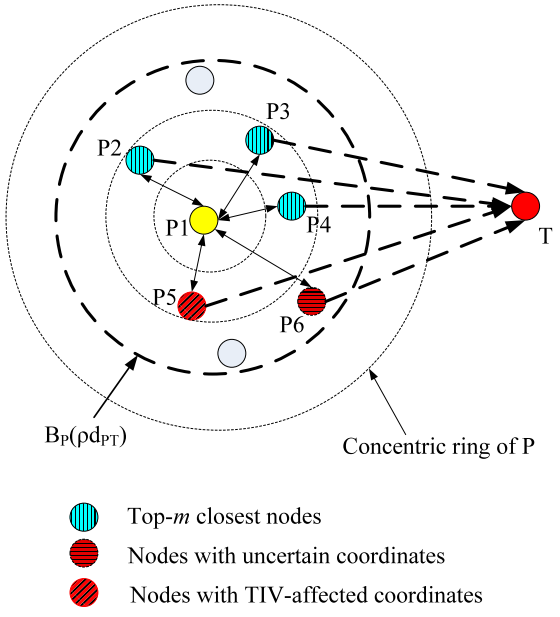


Fig. 11. Selecting candidate neighbors that are close to the target.

- We choose those candidate neighbors S_e whose coordinate errors e_i exceeds the threshold (setting the threshold to be 0.7 performs quite well in practice). If a coordinate has a large coordinate error, this coordinate must have not converged to stable positions. As a result, predicting delay using this coordinate will not be reliable. Therefore, we directly measure delay values from nodes with large coordinate errors to the target.
- We also include all candidate neighbors S_t whose coordinate distance and real delay towards the current node P differs by more than 50 ms, in order to adapt the TIV effects on the coordinates [21].

Fig. 11 summarizes the candidate selection.

Node P next asks the neighbors in the set $S_* = S_c \cup S_e \cup S_t$ to probe the delay to target T, from which node P determines the closest neighbor. Ties are broken by choosing the neighbor

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

Algorithm 5: Detecting the nearest candidate neighbor with delay estimations and direct probes.

```
1  NearestDetector(P, S, x_T, M)
   input  : current node P, candidate neighbors S, target's coordinate
            x_T, DNNL query message M
   output: Nearest candidate neighbor u_1,
   // store the candidate neighbors
2  S_c ← ∅;
3  S_* ← S;
4  while |S_c| == m do // top-m nearest nodes to target T
5      i_m ← arg min_{i∈S_*} ||x_i − x_T||;
6      S_c ← S_c ∪ {i_m};
7      S_* ← S_* − {i_m};
8  end
   // large coordinate error
9  S_c ← S_c ∪ {i | e_i > 0.7, i ∈ S};
   // TIV induced inaccuracy
10 S_c ← S_c ∪ {i | |||x_i − x_P|| − d_iP| > 50ms, i ∈ S};
11 for i ∈ S_c do
12     d_iT ← RTTProbe(i,T);
13     D_S ← D_S ∪ {d_iT};
14 end
15 D_T ← D_S ∪ {d_PT};
16 u_1 ← arg min_{i∈S_c} {d_iT | d_iT ∈ D_T};
17 return u_1;
```

with most accurate coordinate. Algorithm 5 summarizes the above selection criteria.

5.5. Termination test

HybridNN sets the delay reduction threshold $\beta = 1$. Therefore, when the closest neighbor u_1 selected has a larger delay to the target than that of the current node P, node P terminates the DNNL query. Then node P is returned to the target T as the closest service node. Algorithm 6 summarizes the termination test.

Algorithm 6: Determining whether to terminate the DNNL query.

```
1  TerminateTest(P, u_1, T, M)
   input  : current node P, selected nearest candidate u_1, target T,
            DNNL query message M
   output: Nearest node to the target
2  if d_{u_1 T} ≤ d_PT & u_1 ≠ P then
3      M.Path ← M.Path ∪{P};
       // recursive call
4      HybridNN(u_1, T, M);
5  else
6      return u_1;
7  end
```

5.6. Putting it all together

The pseudo-code for HybridNN is given in Algorithm 7.

Algorithm 7: The pseudo-code of HybridNN.

```
1  HybridNN(P, T, M)
   input  : current node P, the target T, DNNL query message M
   Output: Nearest node u_1 to target T
2  S ← chooseCandidates(P, T, M);
3  InitTargetCoord(P, T, M);
4  x_T ← M.x_T;
5  u_1 ← NearestDetector(P, S, x_T, M);
6  u_1 ← TerminateTest(P, u_1, T, M);
7  return u_1;
```

6. Simulation

In this section, we report the results of simulation experiments for two real-world delay data sets described in Section 3.3.

6.1. Experimental setup

We compare HybridNN with several DNNL algorithms.

- *Vivaldi*. We compute the coordinate of each node based on the Vivaldi algorithm [52], and find the nearest service nodes for each requesting node using shortest coordinate distances. The coordinate dimension for Vivaldi is 5. We run a prototype of the Vivaldi system implemented by [52].
- *Meridian*. Meridian [13] finds a nearest server by a distributed search process. We run the prototype of Meridian system implemented by Wong et al. [13]. For the simulation, we set Meridian's β parameter to 0.5 by the default configuration in paper [13].
- *CoordNN*. To quantify the usefulness of delay predictions of HybridNN, we present a DNNL algorithm CoordNN, which is identical with HybridNN except that CoordNN does not issue direct probes when determining the best next-hop neighbors.
- *DirectDN2S*. To evaluate the usefulness of the direct delay probes, we evaluate a DNNL algorithm called DirectDN2S that is identical with HybridNN except that DirectDN2S only issues direct probes to find the best next-hop neighbor without any delay predictions.

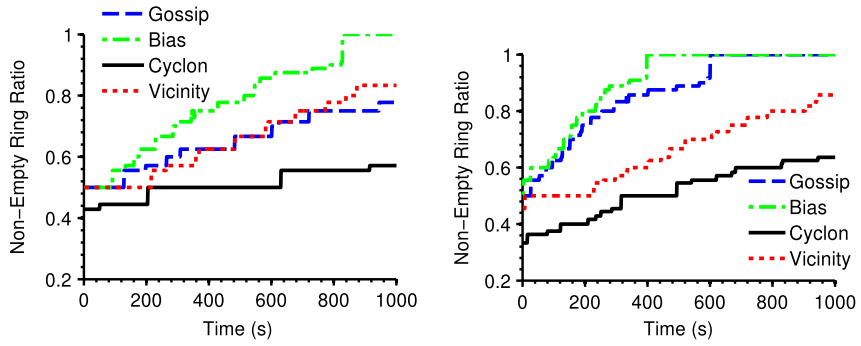
Table 3
Parameter values of HybridNN for simulation.

Parameter	Value
Maximal number of neighbors per ring Δ	8
Delay reduction threshold β	1
Inframetric parameter ρ	3
Coordinate dimension $\|x\|$	5
Sampled neighbors K	10
Number of nearest nodes m	4
Non-empty ring threshold τ	4

For HybridNN, the default configuration is summarized in Table 3. CoordNN and DirectDN2S share identical parameters with HybridNN. We also evaluated the sensitivity of parameters for HybridNN, which is reasonably robust against the parameter choices. The detailed sensitivity results of system parameters for HybridNN can be found in the technical report [36].

We have developed a discrete-time simulator for DNNL. The simulator randomly chooses a set of nodes as service nodes (by default 500) that can receive DNNL queries. Other nodes in the system are clients that can issue DNNL queries to these service nodes. For the Host479 data set, 200 nodes are the service nodes. The DNNL queries are repeated 10,000 times. For each DNNL query, we uniformly select one client as the target machine, and a random service node receiving the query. Besides, the simulation is repeated 5 times by shuffling the set of service nodes to avoid biases in choosing service nodes. For HybridNN, CoordNN, DirectDN2S and Meridian, the inter-gossip events for neighborhood discovery are generated by an exponential distribution with expected value of 1 s. The inter-ring management events are generated by an exponential distribution with expected value of 2 s. For HybridNN, DirectDN2S and CoordNN, the time interval between two random-walk sampling events are generated by an exponential distribution with expected value of 10 s. The inter-DNNL event generation follows an exponential distribution with expected value of 30 s.

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002



(a) DNS3997. (b) Host479.

Fig. 12. The non-empty ring ratio as a function of the simulation time.

The performance metrics for each DNNL query include:

Absolute error: defined as the difference between the estimated nearest service node j and the real nearest service node i to the target T, i.e., $d_{jT} - d_{iT}$.

Search hops: defined as the hop number of the forwarding path of a DNNL query.

6.2. Comparing the performance of neighbor discovery

We first test the quality of rings using the biased neighbor discovery method. Let the *optimal concentric ring* of each node be the one constructed based on the complete delay matrix. We define two performance metrics to analyze the ring quality of concentric rings:

- *Non-empty ring ratio*, defined as the ratio of the number r_M of non-empty rings to the number of non-empty rings r_o by the optimal ring construction, i.e.,

$$\frac{r_M}{r_o}. \tag{28}$$

- *Fullness ratio*, defined as the ratio of the number of found neighbors to the number of neighbors by optimal ring construction for each ring, i.e.,

$$\frac{\sum_{i\in I} \frac{\#_{M_i}}{\#_{o_i}}}{|I|} \tag{29}$$

where I denotes the set of non-empty ring, $\#_{M_i}$ denotes the number of neighbors on the i-th non-empty ring by each sampling method, and $\#_{o_i}$ represents the number of neighbors on the i-th non-empty ring of optimal ring construction.

To see the gains of combining the gossip method with the random walks, we compare our biased neighbor sampling method with existing popular peer sampling methods:

- Gossip, the neighbor discovery method used by Meridian.
- Cyclon [47], a popular peer sampling approach for P2P overlays. In Cyclon, each node periodically exchanges neighbor sets with a random neighbor in a push and pull manner.
- Vicinity [48], a popular biased peer sampling approach for finding semantic clusters on P2P overlays. Vicinity uses the Cyclon to locate random peers on the overlay, but keeps peers that are similar to each other as neighbors in order to form semantic clusters. We define the distance function between two nodes with the pairwise delay value. Since we need to sample nodes for inner and outer rings, we combine two independent Vicinity instances: one for locating K nearest nodes and the other for K farthest nodes.

We configure Cyclon's parameters as in [47] and Vicinity's parameters as in [48]. For a fair comparison, we set the period of triggering the sampling event of Gossip, Cyclon and Vicinity to be identical with that of the bias approach. The number of nodes per ring and the number of rings per concentric ring of all methods are also identical with each other. Also, Vicinity uses the same K value as the bias approach.

Figs. 12 and 13 plot the dynamics of the non-empty ring ratios and the fullness ratios with increasing simulator time. The confidence intervals of the average ratios for all methods are close to the average values and are omitted for brevity. We can see that the gossip based method fails to quickly sample inner and outer rings. On the other hand, the bias approach outperforms other sampling approaches, since the former explicitly samples neighbors from both intra- and inter-clusters by combining gossip and random walks. As a result, the random walks can efficiently fill empty rings and complement the gossip method quite well.

Cyclon is less accurate than the gossip method. This is because the gossip method selects one neighbor from each ring covering different delay ranges, but Cyclon's sampled nodes are agnostic of the rings. Since nodes on the concentric ring are clustered towards a small number of rings, Cyclon typically selects samples covering lower delay ranges than the gossip method.

The Vicinity method is more accurate than the Cyclon method, since Vicinity combines the semantic clustering with Cyclon to simultaneously sample neighbors uniformly and K nearest and farthest nodes. However, we can see that Vicinity is less efficient than the bias method. This is because Vicinity assumes that if P is close to Q and Q is close to R then P is also close to R. Unfortunately, the network delay space is not a metric space and a fraction of triples violates the triangle inequality, which may cause P to be far from R. As a result, Vicinity's peer sampling is impaired when TIV cases happen.

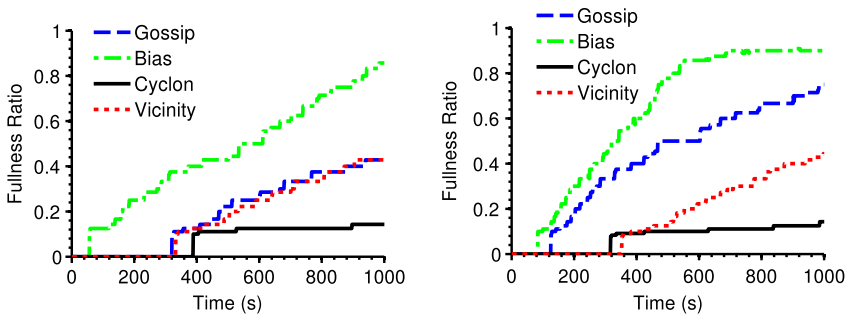
6.3. Comparison

6.3.1. Absolute error

Fig. 14 shows the absolute errors of the different algorithms. DirectDN2S achieves lowest absolute errors except for the Host479 data sets. HybridNN is close to DirectDN2S in terms of reducing absolute errors. However, HybridNN is the most accurate on Host479 data set. Next, CoordNN is worse than both DirectDN2S and HybridNN. The high accuracy of DirectDN2S and HybridNN compared to CoordNN indicates that using direct probes greatly reduces the error of the estimation, while using coordinate distances alone can lead to getting trapped in a local minima.

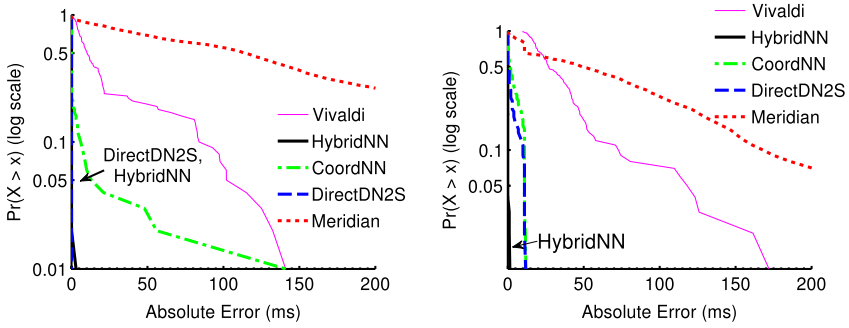
The inaccuracy of DirectDN2S compared to HybridNN on the Host479 data set is rather counter-intuitive. This inaccuracy may be caused by the asymmetry in the delay data sets that misleads

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002



(a) DNS3997. (b) Host479.

Fig. 13. The fullness ratio as a function of the simulation time.



(a) DNS3997. (b) Host479.

Fig. 14. The CCDFs of absolute errors.

the greedy search into a local minima, since DirectDN2S is more accurate than HybridNN on the other three data sets that are all symmetric for pairwise delays [36].

Meridian has the largest absolute error compared to other algorithms including Vivaldi, which implies that the coordinate distances provided by Vivaldi are at least effective if we use it in the centralized approach. The superiority of Vivaldi over Meridian is consistent with the experiments independently performed by Choffnes and Bustamante [53]. The main reason for the inaccuracy of Meridian is the local minima caused by the TIV and clustering in the delay space; while Vivaldi can adapt to TIV based on adaptive coordinate movements.

On the other hand, from Fig. 15 we see that HybridNN is able to bypass the bad local minimum caused by the asymmetry in the delay values. This is because HybridNN does not always choose the neighbor closest to the target as the forwarding node, since HybridNN also incorporates the approximated delay predictions when choosing neighbors. However, when the delay values are symmetric, e.g., on DNS3997 data set, HybridNN can be trapped at worse local minimum than DirectDN2S.

6.3.2. Search hops

We next quantify the distributions of the number of search hops for DNNL algorithms, as shown in Fig. 16. The search hops of most DNNL queries are rather modest for all DNNL algorithms. Meridian has two search hops in about 80% of the cases. While HybridNN and DirectDN2S have no more than three hops in over 80% of the cases. Almost all search processes for Meridian, HybridNN and DirectDN2S have less than six hops. On the other hand, CoordNN needs more search hops than Meridian, HybridNN and the DirectDN2S; even worse, a fraction of search hops of CoordNN are larger than ten.

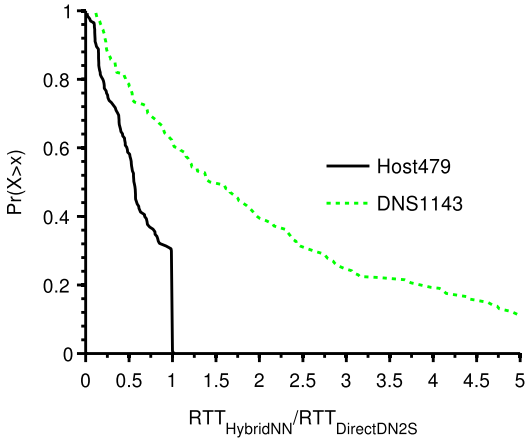


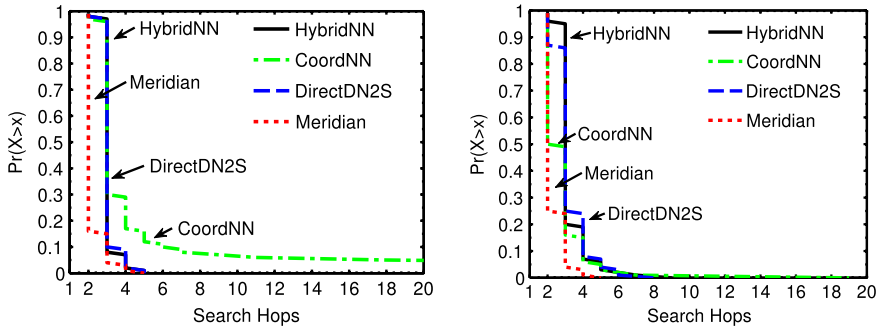
Fig. 15. The latency ratio between the nearest server found by HybridNN and that by DirectDN2S.

6.4. Applying adjustments to improve meridian

Having shown HybridNN is able to outperform Meridian in orders of magnitudes, we next put key components of HybridNN on Meridian and see whether the performance of Meridian is also improved. The variants of Meridian include:

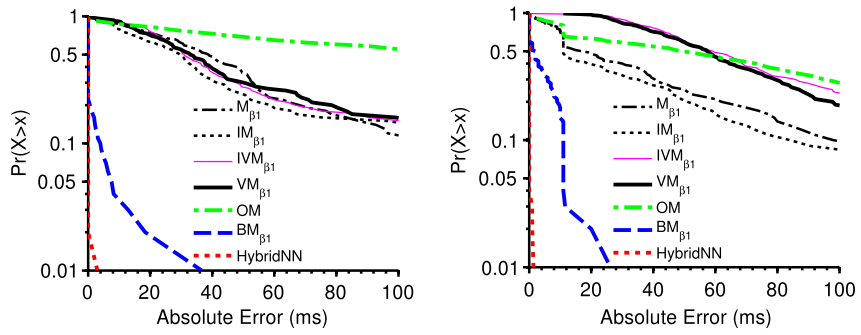
- *OM*, denotes the original Meridian with $\beta = 0.5$ and gossip based neighbor discovery;
- $M_{\beta 1}$, denotes the Meridian with $\beta = 1$ and gossip based neighbor discovery;
- $IM_{\beta 1}$, denotes the Meridian with $\beta = 1$, gossip based neighbor discovery and the inframetric model based candidate selection;
- $IVM_{\beta 1}$ denotes the Meridian with $\beta = 1$, gossip based neighbor discovery and the inframetric model based candidate selection and the latency prediction;

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002



(a) DNS3997. (b) Host479.

Fig. 16. The CCDFs of search hops.



(a) DNS3997. (b) Host479.

Fig. 17. The CCDFs of absolute errors for Meridian and several variants.

- $VM_{\beta1}$, denotes the Meridian with $\beta = 1$, gossip based neighbor discovery and the latency prediction;
- $BM_{\beta1}$ denotes the Meridian with bias neighbor discovery and $\beta = 1$.

Fig. 17 plots the results. We can see that only the bias based neighbor discovery significantly improves Meridian's accuracy, while the other variants just add minor improvements.

First, comparing Fig. 14 with 17, we confirm that setting $\beta = 1$ improves Meridian's accuracy. Intuitively, setting β to larger numbers will increase Meridian's accuracy, since we will select more candidate neighbors at each search step $[(1-\beta)d_{PT}, (1+\beta)d_{PT}] = [0, 2d_{PT}]$. However, the search process will terminate much slower than that for $\beta = 0.5$.

On the other hand, Meridian with $\beta = 1$ also approximates the inframetric model based neighbor sampling process, since the latter selects neighbors from the interval $[0, \rho d_{PT}]$, where ρ denotes the inframetric parameter.

Second, using the inframetric model based neighbor sampling results in similar or better performance with that by setting $\beta = 1$. As a result, even when we do not change the β value, the inframetric model based neighbor selection can significantly improve Meridian's accuracy.

Third, we see that using the bias method for the neighbor discovery significantly reduces the absolute errors compared to Meridian variants that use the gossip method, since the bias method provides more neighbors covering the inner and outer rings.

Finally, replacing the direct measurements with the Vivaldi coordinate based delay estimation decreases the search accuracy. This is because network coordinate may incorrectly predict the ground-truth nearest node to the search at each search step, which decreases the search accuracy accordingly.

7. PlanetLab experiments

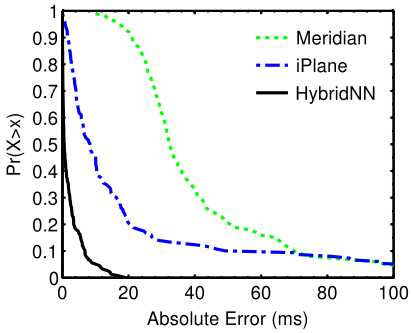
We have implemented a prototype for distributed nearest service node location in Java using the asynchronous communication library. We implemented both, HybridNN and Meridian. The core logic consists of around 5000 lines of codes comprising three main modules: (1) *probe module*, which uses the kernel-level ping for delay measurements, to reduce application level perturbations caused by high loads of PlanetLab nodes; (2) *neighborhood management module*, which finds and maintains neighbors on the concentric rings; (3) *DNNL module*, which implements the distributed nearest service node location of HybridNN and Meridian.

Our objective is to compare the accuracy and efficiency of DNNL queries based on real-world deployments. We choose 173 servers distributed globally on the PlanetLab as the service nodes. Then we select another 412 servers on the PlanetLab as the target machines. Service nodes are sequentially added into the system. We introduce a warm-up period for each newly joined service node in order to stabilize the DNNL algorithm. During the warm-up period, each service node updates its concentric ring, but does not answer DNNL queries. Our experiments last one week in May 2011 between May 5 and May 12.

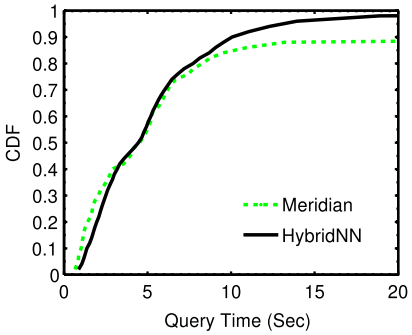
We compare HybridNN with Meridian and iPlane [29]. We choose the same parameter configurations for HybridNN and Meridian as in the Simulation section (Section 6.1). For iPlane, we query iPlane to obtain the delays between service nodes and target machines. Then we compute the nearest service node for each target machine.

Besides, we compute the ground-truth nearest servers using direct probes in order to compare the found nearest servers to the ground-truth nearest servers (denoted as *Direct*). Furthermore, since the pairwise delays between PlanetLab machines dynamically vary, we represent the pairwise long-term delay based on the median value of ping records. Finally, we select the service node

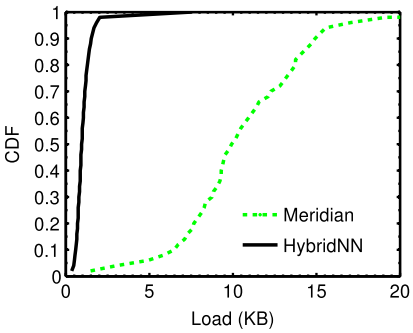
Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002



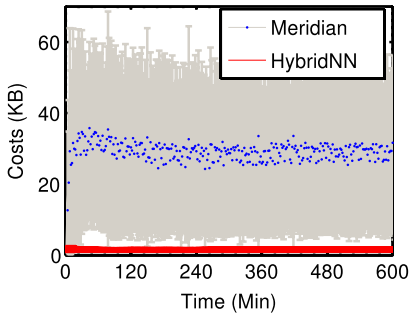
(a) Absolute error.



(b) Query time.



(c) Query load.



(d) Control overhead.

Fig. 18. Performance comparison on PlanetLab.

that has the lowest median delay to the target as the ground-truth nearest server.

7.1. Accuracy

We first compare the accuracy of different methods with the absolute error metric defined in Section 6.1. The results are shown in Fig. 18(a). HybridNN has significantly lower absolute errors than Meridian; iPlane is similar with HybridNN, but incurs slightly higher errors. For instance, iPlane also has around 3% of DNNL queries with absolute errors above 100 ms. The inaccuracy of iPlane is caused by the mismatch of the estimated routing paths and the real-world ones. The inaccuracy of Meridian shows that Meridian is easily trapped at local minimum far away from the optimal solution.

7.2. Completion time

We next evaluate the completion time of individual DNNL queries for HybridNN and Meridian. The completion time of a DNNL query equals the sum of the search delay at each step, which then equals the slowest responses from all candidate neighbors.

Empirically, we have found that both HybridNN and Meridian complete DNNL queries within three search hops, which is consistent with the simulation results in Fig. 16. However, the overall query time for DNNL searches depends on not only the number of search hops, but also the completion time of the message exchanges and delay probes.

Fig. 18(b) plots the distributions of query time of HybridNN and Meridian. Around 85% of the DNNL queries in HybridNN are similar with those of Meridian. Therefore, query time for HybridNN and Meridian are similar in most cases.

However, we also see that in Meridian around 20% of the queries take much more time to answer, and 10% have query time larger than 15 s; while the hybrid measurement approach of HybridNN can avoid large query latencies. Since Meridian requires direct probes from all candidate neighbors to targets, some candidate neighbors that are far from the target take much longer time to send the response to the requesting node. As a result, the total search period is prolonged by these nodes.

On the other hand, HybridNN avoids most direct probes by delay predictions. Moreover, since Vivaldi coordinates have converged after the warm-up period, the candidate neighbors are mainly the top-m nodes whose coordinates are nearest to the target. As a result, collecting delays from these candidate neighbors to the target complete quickly. Therefore, some of HybridNN's search processes has lower search times than Meridian.

7.3. Query overhead

We define the load of a DNNL query as the total size of the transmitted packets during the DNNL process. We plot the Cumulative Distribution Function (CDFs) of the loads for HybridNN and Meridian in Fig. 18(c). The load of HybridNN is significantly lower than that of Meridian. In more than 95% of the cases the load of HybridNN is less than 2 kB, while in more than 50% of the cases the load of Meridian is more than 10 kB, which is due to the large size of the candidate neighbor set for DNNL queries. Therefore, the delay estimation of HybridNN substantially reduces the query overhead.

7.4. Control overhead

We measure the bandwidth overhead of the neighborhood management in HybridNN and Meridian for each service node every two minutes, as shown in Fig. 18(d). The maintenance overhead of Meridian includes both, the gossip process and the ring maintenance costs, while the maintenance of HybridNN includes the gossip messages and random-walk sampling messages. The average maintenance overhead of HybridNN is around 2 kB per minute, and for Meridian is over 20 kB per minute. Since the time

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

interval of ring maintenance for both, HybridNN and Meridian is identical, the all-pair probes between nodes in the same ring are the main cause of the control overhead in Meridian. On the other hand, as HybridNN uses the coordinate distances to update the rings, it does not need to do all-pair probes between nodes in a ring.

8. Conclusion

We have addressed the problem of designing an accurate and efficient DNNL algorithm in a principled and comprehensive way. We first show that the Internet delay space is not a metric space and propose to use the inframetric model that allows for violations of the triangle inequality and delay asymmetries. We then use the inframetric as a foundation to design new DNNL algorithms with strong theoretical guarantees concerning search overhead and accuracy of the search results. Finally, we apply all the insights to design a new DNNL algorithm called HybridNN, which locates nearest service nodes accurately and efficiently. HybridNN contains several techniques to improve its accuracy and efficiency: (i) It maximizes the diversity in the neighbor set by discovering neighbors within each delay range through a light-weight neighbor sampling process. (ii) It reduces the measurement costs for locating closer servers. HybridNN combines network coordinate based delay estimation and direct probes for fast and efficient nearest service node selection. (iii) Finally, HybridNN terminates the search process conservatively in order to obtain better approximations of nearest service nodes.

We also evaluate how each of the techniques used in HybridNN can improve the search accuracy of Meridian. We see that it is the biased neighbor discovery brings the biggest improvement in accuracy.

Extensive simulation and a prototype deployment on the PlanetLab confirm the efficiency and effectiveness of HybridNN.

Acknowledgments

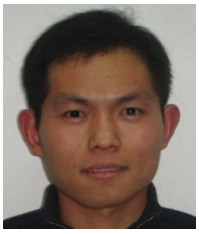
We would like to thank the reviewers for their numerous and very constructive comments. This work was supported by the National Grand Fundamental Research 973 Program of China (Grant No. 2011CB302601), the National High Technology Research and Development 863 Program of China (Grant No. 2013AA010206), the National Natural Science Foundation of China (Grant No. 60873215), the Natural Science Foundation for Distinguished Young Scholars of Hunan Province (Grant No. S2010J5050), Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 200899980003) and the Collaborative Project FIGARO supported by the European Commission under the 7th Framework Program (Grant No. 258378).

References

[1] U. Schwiegelshohn, R.M. Badia, M. Bubak, M. Danelutto, S. Dustdar, F. Gagliardi, A. Geiger, L. Hluchy, D. Kranzlmüller, E. Laure, T. Priol, A. Reinefeld, M. Resch, A. Reuter, O. Rienhoff, T. Rüter, P. Sloot, D. Talia, K. Ullmann, R. Yahyapour, G. von Voigt, Perspectives on grid computing, Future Generation Computer Systems 26 (2010) 1104–1115.
[2] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, S. Haridi, Peer-to-peer resource discovery in grids: models and systems, Future Generation Computer Systems 23 (2007) 864–878.
[3] W. Yu, S. Chellappan, D. Xuan, P2P/grid-based overlay architecture to support VoIP services in large-scale IP networks, Future Generation Computer Systems 21 (2005) 209–219.
[4] P. Morillo, S. Rueda, J.M. Orduña, J. Duato, Ensuring the performance and scalability of peer-to-peer distributed virtual environments, Future Generation Computer Systems 26 (2010) 905–915.
[5] R. Rodrigues, P. Druschel, Peer-to-peer systems, Communications of the ACM 53 (2010) 72–82.
[6] Microsoft, Office live workspace, 2011. http://workspace.officelive.com/zh-hk/.
[7] Google, Google Maps, 2011. http://maps.google.com/.
[8] M.S. Artigas, P.G. López, eSciGrid: a P2P-based e-science grid for scalable and efficient data sharing, Future Generation Computer Systems 26 (2010) 704–719.
[9] G. Mateescu, W. Gentzsch, C.J. Ribbens, Hybrid computing—where HPC meets grid and cloud computing, Future Generation Computer Systems 27 (2011) 440–453.
[10] F. Agboma, A. Liotta, QoE-aware QoS management, in: Proc. of MoMM 2008, ACM, New York, NY, USA, 2008, pp. 111–116.
[11] R. Krishnan, H.V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, J. Gao, Moving beyond end-to-end path information to optimize CDN performance, in: Proc. of IMC 2009.
[12] A.-J. Su, D.R. Choffnes, A. Kuzmanovic, F.E. Bustamante, Drafting behind Akamai (travelocity-based detouring), in: Proc. of SIGCOMM 2006, pp. 435–446.
[13] B. Wong, A. Slivkins, E.G. Sirer, Meridian: a lightweight network location service without virtual coordinates, in: Proc. of SIGCOMM 2005, pp. 85–96.
[14] M.J. Freedman, K. Lakshminarayanan, D. Mazières, OASIS: anycast for any service, in: Proc. of NSDI 2006.
[15] V. Vishnumurthy, P. Francis, On the difficulty of finding the nearest peer in P2P systems, in: Proc. of IMC 2008, pp. 9–14.
[16] C. Lumezanu, R. Baden, N. Spring, B. Bhattacharjee, Triangle inequality variations in the internet, in: Proc. of IMC 2009, pp. 177–183.
[17] P. Fraigniaud, E. Lebhar, L. Viennot, The inframetric model for the internet, in: Proc. of INFOCOM 2008, pp. 1085–1093.
[18] G.R. Hjaltason, H. Samet, Index-driven similarity search in metric spaces (survey article), ACM Transactions on Database Systems 28 (2003) 517–580.
[19] K.L. Clarkson, Nearest-neighbor searching and metric space dimensions, in: G. Shakhnarovich, T. Darrell, P. Indyk (Eds.), Nearest-Neighbor Methods for Learning and Vision: Theory and Practice, MIT Press, 2006, pp. 15–59.
[20] E. Chávez, G. Navarro, R. Baeza-Yates, J.L. Marroquín, Searching in metric spaces, ACM Computing Surveys 33 (2001) 273–321.
[21] G. Wang, B. Zhang, T.S.E. Ng, Towards network triangle inequality violation aware distributed systems, in: Proc. of IMC 2007, pp. 175–188.
[22] R.L. Carter, M.E. Crovella, Server selection using dynamic path characterization in wide-area networks, in: Proc. of INFOCOM 1997, pp. 1014–1021.
[23] R.L. Carter, M.E. Crovella, On the network impact of dynamic server selection, Computer Networks 31 (1999) 2529–2558.
[24] J.D. Guyton, M.F. Schwartz, Locating nearby copies of replicated internet servers, in: Proc. of SIGCOMM 1995, pp. 288–298.
[25] S.M. Hotz, Routing information organization to support scalable interdomain routing with heterogeneous path requirements, Ph.D. Thesis, Computer Science Department, University of Southern California, Los Angeles, California, 1994.
[26] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, Topologically-aware overlay construction and server selection, in: Proc. of INFOCOM 2002, vol. 3, pp. 1190–1199.
[27] P. Sharma, Z. Xu, S. Banerjee, S.-J. Lee, Estimating network proximity and latency, Computer Communication Review 36 (2006) 39–50.
[28] A.-J. Su, D. Choffnes, F.E. Bustamante, A. Kuzmanovic, Relative network positioning via CDN redirections, in: Proc. of ICDCS 2008, pp. 377–386.
[29] H.V. Madhyastha, E. Katz-Bassett, T.E. Anderson, A. Krishnamurthy, A. Venkataramani, iPlane nano: path prediction for peer-to-peer applications, in: Proc. of NSDI 2009, pp. 137–152.
[30] H.V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T.E. Anderson, A. Krishnamurthy, A. Venkataramani, iPlane: an information plane for distributed services, in: Proc. of OSDI 2006, pp. 367–380.
[31] S. Banerjee, C. Kommareddy, B. Bhattacharjee, Scalable peer finding on the internet, in: Proc. of Global Internet Symposium 2002.
[32] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, A. Akella, On the treeness of internet latency and bandwidth, in: Proc. of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, ACM, Seattle, WA, USA, 2009, pp. 61–72.
[33] M. Waldvogel, R. Rinaldi, Efficient topology-aware overlay network, ACM Computer Communication Review 33 (2003) 101–106.
[34] A.-M. Kermarrec, M. van Steen, Gossiping in distributed systems, ACM SIGOPS Operating Systems Review 41 (2007) 2–7.
[35] D.R. Karger, M. Ruhl, Finding nearest neighbors in growth-restricted metrics, in: Proc. of STOC 2002, pp. 741–750.
[36] Y. Fu, Y. Wang, E. Biersack, HybridNN: supporting network location service on generalized delay metrics for latency sensitive applications, 2011. http://arxiv.org/abs/1108.1928.
[37] M. Costa, M. Castro, A.I.T. Rowstron, P.B. Key, PIC: practical internet coordinates for distance estimation, in: Proc. of ICDCS 2004, pp. 178–187.
[38] P. Wendell, J.W. Jiang, M.J. Freedman, J. Rexford, DONAR: decentralized server selection for cloud services, in: Proc. of SIGCOMM 2010, pp. 231–242.

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002

[39] F. Dabek, R. Cox, M.F. Kaashoek, R. Morris, Vivaldi: a decentralized network coordinate system, in: Proc. of SIGCOMM 2004, pp. 15–26.
[40] B. Zhang, T.S.E. Ng, A. Nandi, R.H. Riedi, P. Druschel, G. Wang, Measurement-based analysis, modeling, and synthesis of the internet delay space, IEEE/ACM Transactions on Networking 18 (2010) 229–242.
[41] K.P. Gummadi, S. Saroiu, S.D. Gribble, King: estimating latency between arbitrary internet end hosts, in: Proc. of IMW 2002, pp. 5–18.
[42] D.R. Choffnes, M. Sanchez, F.E. Bustamante, Network positioning from the edge—an empirical study of the effectiveness of network positioning in P2P systems, in: Proc. of INFOCOM 2010, pp. 291–295.
[43] Y. Schwartz, Y. Shavitt, U. Weinsberg, A measurement study of the origins of end-to-end delay variations, in: Proc. of PAM 2010, pp. 21–30.
[44] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang, IDMaps: a global internet host distance estimation service, IEEE/ACM Transactions on Networking 9 (2001) 525–540.
[45] H. Zheng, E.K. Lua, M. Pias, T.G. Griffin, Internet routing policies and round-trip-times, in: Proc. of PAM 2005, pp. 236–250.
[46] Y. Zhang, N.G. Duffield, On the constancy of internet path properties, in: Proc. of IMW 2001, pp. 197–211.
[47] S. Voulgaris, D. Gavidia, M. van Steen, CYCLON: inexpensive membership management for unstructured P2P overlays, Journal of Network and Systems Management 13 (2005) 197–217.
[48] S. Voulgaris, E. Riviere, A.-M. Kermarrec, M. van Steen, Sub-2-Sub: self-organizing content-based publish subscribe for dynamic large scale collaborative networks, in: Proc. of IPTPS 2006.
[49] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, M. van Steen, Gossip-based peer sampling, ACM Transactions on Computer Systems 25 (2007).
[50] A. Slivkins, Distance estimation and object location via rings of neighbors, Distributed Computing 19 (2007) 313–333.
[51] A. Slivkins, Towards fast decentralized construction of locality-aware overlay networks, in: Proc. of PODC 2007, pp. 89–98.
[52] J. Ledlie, P. Gardner, M.I. Seltzer, Network coordinates in the wild, in: Proc. of NSDI 2007.
[53] D.R. Choffnes, F.E. Bustamante, Pitfalls for testbed evaluations of internet systems, ACM SIGCOMM Computer Communication Review 40 (2010) 43–50.



Yongquan Fu received the B.S. degree in computer science and technology from the School of Computer of Shandong University, China, in 2005, and received the M.S. degree in Computer Science and technology from the School of Computer Science of National University of Defense Technology, China, in 2008. He is currently a Ph.D. candidate in the School of Computer Science of National University of Defense Technology. He is a student member of CCF and ACM. His current research interests lie in the areas of network measurement, Peer-to-Peer networks and distributed system.

Yijie Wang received the Ph.D. degree from the National University of Defense Technology, China in 1998. She was a recipient of the National Excellent Doctoral Dissertation (2001), a recipient of Fok Ying Tong Education Foundation Award for Young Teachers (2006) and a recipient of the Natural Science Foundation for Distinguished Young Scholars of Hunan Province (2010). Now she is a Professor in the National Key Laboratory for Parallel and Distributed Processing, National University of Defense Technology. Her research interests include network computing, massive data processing, parallel and distributed processing.

Ernst Biersack studied computer science at the Technische Universitä München and at the University of North Carolina at Chapel Hill. He received the Dipl. Infom. (M.S.) and Dr. rer. nat. (Ph.D.) degrees in computer science from the Technische Universität München, Munich, Germany, and the Habilitation à Diriger des Recherches from the University of Nice, France. From March 1989 to February 1992, he was a Member of Technical Staff with the Computer Communications Research District of Bell Communications Research, Morristown, US. Since March 1992, he has been a Professor in telecommunications at Eurecom, Sophia Antipolis, France. His current research is on peer-to peer systems and network tomography.

Please cite this article in press as: Y. Fu, et al., HybridNN: An accurate and scalable network location service based on the inframetric model, Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.12.002