A Lightweight Network Proximity Service Based On Neighborhood Models

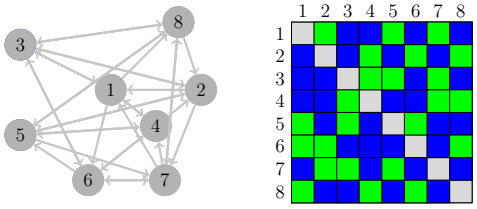
Yongjun Liao, Wei Du and Guy Leduc
Research Unit in Networking (RUN), University of Liège, Belgium

***Abstract*—This paper proposes a network proximity service based on the neighborhood models used in recommender systems. Unlike previous approaches, our service infers network proximity without trying to recover the latency between network nodes. By asking each node to probe a number of landmark nodes which can be servers at Google, Yahoo and Facebook, etc., a simple proximity measure is computed and allows the direct ranking and rating of network nodes by their proximity to a target node. The service is thus lightweight and can be easily deployed in e.g. P2P and CDN applications. Simulations on existing datasets and experiments with a deployment over PlanetLab showed that our service achieves an accurate proximity inference that is comparable to state-of-the-art latency prediction approaches, while being much simpler.**

I. Introduction

The knowledge of network proximity is critical to achieve Quality-of-Service (QoS) guarantees for end users. For example, in Peer-to-Peer (P2P) applications and Content Distribution Networks (CDNs), it is desired to direct user requests to a peer or a server that is nearby and thus with a connection of small latencies. In this paper, network latency refers only to round-trip time (RTT).

To exploit the proximity information in large distributed systems, a practical challenge is the efficient acquisition because active probing of network latency for all paths in a large network is infeasible due to the quadratic complexity. This issue has been well studied, resulted in a rich line of research on network proximity inference based on latency measurements on a few paths [1], [2], [3]. In particular, a recent progress is that the inference of network latencies can be cast as a matrix completion problem whereby a partially observed matrix is to be completed [4], [5], [6]. Here, the matrix contains latencies between network nodes with some of them known and the others unknown, shown in Figure 1(a). We then observed a similarity between network inference and the problem of *recommender systems* which studies the prediction of preferences of users to items [7], shown in Figure 1(b). If we consider network latency as a preference measure between nodes, then peer and server selection is just a recommendation task. This observation enables us to leverage the rapid advances in machine learning and investigate the applicability of various solutions to recommender systems for network inference. Our previous studies have shown that a class of matrix factorization techniques are suitable and achieved good results that are known to be acceptable for recommendation tasks [6].



(a) A matrix completion view of latency inference. In the matrix, the blue entries contain measured path latencies, represented by directed edges in the graph, and the green entries are missing.

	item1	item2	item3	item4	item5
user1	5	3	4	1	?
user2	5	3	4	1	5
user3	5	?	4	1	5
user4	1	3	2	5	1
user5	4	?	4	4	4

(b) An example of recommender system.

Fig. 1. Connection between network inference and recommender systems.

Alternatively, neighborhood models are also widely used in recommender systems which exploit the similarities between users and between items [8]. For example, two users are considered similar if they rate a set of items similarly. Meanwhile, two items are considered similar if they are given similar ratings by a set of users. Thus, two kinds of recommendations can be made to a user: liked items by similar users and items that are similar to the liked items by that user. In this paper, we develop a lightweight network proximity service based on neighborhood models whereby, **if two nodes have similar proximity to a number of common nodes, the two nodes are likely to be close to each other**. Different from previous work, our approach infers proximity without recovering network latencies. A simple proximity measure is computed and can be exploited for ranking and rating network paths which is useful in e.g. P2P and CDN applications for peer and server selection. The approach is highly scalable and involves only the latency measurement by *PING* from each node to a small number of pre-selected landmark nodes which can be servers at Google, Yahoo and Facebook, etc. Simulations on existing

This work was supported by the EU under project FP7-ICT mPlane.
Yongjun Liao now is with LIP Lab, IXXI, ENS de Lyon, France and Wei Du is with CITI Lab, INSA-Lyon, France.

datasets and experiments with a deployment on a real network, namely PlanetLab, showed that our approach provided accurate proximity services that are comparable to state-of-the-art latency inference approaches, while being much simpler.

The rest of the paper is organized as follows. Section II summarizes related work on network proximity inference. Section III introduces our network proximity services based on neighborhood models. Section IV and V describe results of both simulations and the deployment on PlanetLab. Section VI gives the conclusion.

II. Related Work

Most related work inferred proximity by estimating latencies between network nodes. For example, GNP [1] inferred RTTs by embedding network nodes into a metric space and Vivaldi [2] decentralized the inference based on Euclidean embedding by removing the landmarks in GNP. Likewise, IDES [3] and DMFSGD [4] solved the inference problem by matrix factorization in a centralized and decentralized manner respectively. We refer interested readers to [9] for a survey of network latency prediction. In general, existing approaches employ a common framework that fits a prediction model by using some optimization scheme and a small number of latency measurements. For the decentralized processing in networks, message exchanges are performed between network nodes [2], [4].

While interesting, the most popular application for latency prediction has always been peer and server selection in P2P networks and CDNs. In such a recommendation task, the absolute latency values are only exploited to rank network nodes by their proximity to a target node. Thus, it is useful to develop simpler approaches to do ranking directly without trying to recover latency. This paper describes a novel approach based on the idea in neighborhood models for recommender systems which allows just that. The biggest advantage of our approach is its simplicity which, unlike previous latency prediction approaches, requires no optimization of any kind. By asking each node to probe a number of landmark nodes, a simple proximity measure is computed and allows the ranking of network nodes without recovering latencies. Note that our service is different from other landmark-based systems such as GNP and IDES in that we need no control over the landmark nodes. In addition, this paper also studies the impacts of the dynamic measurement and of the selection of the landmarks, showing that our proximity service is stable and insensitive to the latency dynamics and to the random selection of the landmarks.

III. Network Proximity Service

A. Measuring Network Proximity

Instead of measuring latencies between network nodes, we require each node to probe the latencies to a number of landmark nodes and the latency measurements are put in a feature vector which is attached to each node, illustrated in Figure 2. Let $v_i = [v_{i1}, \ldots, v_{ik}]^T$ be the feature vector of node i, where v_{ij} is the latency between node i and landmark j and

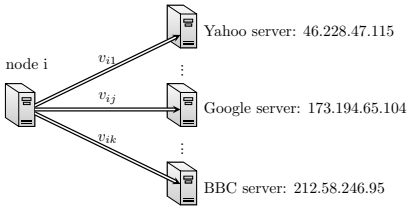


Fig. 2. Network proximity service.

k is the number of landmarks. The landmark nodes can be any IP addresses on the Internet such as content servers in Google, Yahoo and Facebook or news and university web servers which stay alive stably and respond to the PING measurement. Note that anycast IP addresses such as Google public DNS servers at $8.8.8.8$ and $8.8.4.4$ need to be avoided.

A similarity/dissimilarity measure between the feature vectors of two nodes can then be computed using e.g. the correlation, the L2 or L1 norm and the cosine of the included angle, etc., among which the L1 norm is chosen due to its robustness to noisy and outlier measurements [10], given by

$$p_{ij} = \frac{1}{k} \sum_{l=1}^{k} |v_{il} - v_{jl}|. \tag{1}$$

p_{ij} is a proximity measure, although it does not inform us about the actual latency between i and j. Intuitively, when p_{ij} is small, i.e. i and j have similar latencies to the landmarks, chances are that i and j are close to each other. This is exactly the idea in neighborhood models for recommender systems.

Thus, the proximity measure carries information that can be exploited to rank network nodes. For example, if $p_{ij} < p_{il}$, then we can guess that i is closer to j than to l, i.e. the latency between i and j is smaller than that between i and l. Obviously, proximity inference based on p_{ij} is less accurate than based on measured latencies. However, a proximity service based on active probing would require $O(n^2)$ measurements for a network of n nodes which does not scale well. In contrast, our new service reduces the measurement overhead from $O(n^2)$ to $O(kn)$, with k likely to be independent of n. In addition, comparing to state-of-the-art latency prediction approaches, our proximity service is much simpler and requires no computation for learning a prediction model such as Euclidean embedding for Vivaldi [2] and matrix factorization for DMFSGD [4]. We will show in Section IV and V that, on ranking network nodes according to the proximity, our service achieved comparable accuracies to Vivaldi and DMFSGD.

Note that it is possible that a node has a poor connection to all landmarks, i.e. $\min(v_i)$ is larger than a threshold. We consider such cases as that the node has a poor connection to the entire Internet, due probably to a poor Internet access link, and turn the proximity measure between that node and any other node in the network to a large constant. This prevents nodes with poor Internet connections from being labeled as close to each other.

B. Landmark Selection

It is easy to see that the proximity measure depends on both the locations and the numbers of the landmark nodes. In practice, we can check the suitability of the landmarks by computing statistics such as variance of the latencies. There are two considerations.

- For node i, v_i is more informative about its location if some latencies in v_i are small and some are large, i.e. i is close to some landmarks and far away from some others. Thus, the lack of variance in v_i is generally a good indicator of the poor choice of the landmarks for node i. If many nodes have small variance in their feature vectors, a likely reason is that many landmarks are in the same region and close to each other.
- For each landmark, we can also construct a feature vector containing latencies from each node to the landmark. Let $v_i^L = [v_{1i}, \ldots, v_{ni}]^T$ be the feature vector for landmark i. Similarly, the lack of variance in v_i^L is also a good indicator that landmark i is not suitable to serve as a landmark for the nodes in the network, which happens when the landmark has an anycast IP address[1].

Essentially, we can construct a matrix containing latencies between nodes and landmarks, denoted by $V = [v_1, \ldots, v_n]$, and it is desired that the variance in both the rows and the columns in V are large. If we have a set of landmarks from which we choose a few, we can design a selection algorithm that maximizes the variances in V. Empirically, we found that if the landmarks are well distributed all over the world, the variances in V are generally large enough and a good network proximity service can be achieved using a small number of randomly selected landmarks, shown in Section IV.

We performed a test on PlanetLab on June 2nd, 2014 that from 592 live nodes we pinged an anycast Google public DNS server at 8.8.8.8, a unicast Google Web server at 173.194.65.104 and a unicast DNS server at 209.210.172.9 and received latencies to all three servers from 310 nodes, shown in Figure 3. We can see that both the mean and the variance of the latencies to 8.8.8.8 are much smaller, which is expected as 8.8.8.8 corresponds to 28 servers at different locations. Note that among all the latencies to 8.8.8.8, only 12 are larger than 100ms and the latencies from all 11 Chinese nodes are larger than 180ms[2]. This suggests that even an anycast IP may provide useful proximity information about nodes in particular areas. For example, a node is unlikely to be in China if its latency to 8.8.8.8 is smaller than 100ms. By choosing the right landmarks, we may design a location classifier that pinpoints a network node to a small region.

[1]Another possible reason is that most nodes in the network are close to each other. This is a trivial situation which we rule out in this paper.

[2]The only other node with a latency to 8.8.8.8 greater than 100ms is limplanetlab-2.univ-reunion.fr. The mean latency is 213ms and the variance is 4ms. The node is in the University of La Réunion located in a French island in the Indian Ocean.

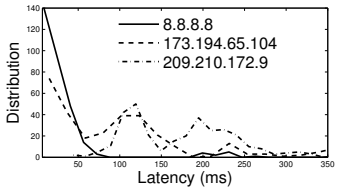


Fig. 3. Distributions of RTT to an anycast Google public DNS server at 8.8.8.8, a unicast Google Web server at 173.194.65.104 and a unicast DNS server at 209.210.172.9.

C. Service Architecture and Applications

Our network proximity service is thus simple, scalable and lightweight. The only overhead is k latency measurements to the pre-selected landmarks at each node. For distributed systems with a centralized control, a tracker node is deployed to which each node registers its feature vector. The tracker node calculates the proximity between nodes and recommends nearby peers or servers for a node. In a fully decentralized architecture with no central node to gather information, each node keeps its own feature vector and searches for nearby peers and servers using a gossip protocol [11].

Potential applications include peer selection for P2P applications and server selection for CDNs. To calculate the proximity between users and servers in a CDN scenario, both users and servers have to probe the landmarks to get feature vectors, which is difficult to assume in the CDN case. Instead, given the symmetry of the RTT measurements, we can probe the users and the servers from the landmarks, assuming that the proximity service has its own landmarks. The feature vectors can then be built by fetching the measurements from the landmarks. Besides, the proximity service can also be exploited to cluster network nodes.

IV. Simulations on Existing Datasets

We performed simulations on the following datasets:

- **Meridian** contains static RTTs between 2500 DNS servers obtained from the Meridian project [12].
- **P2PSim** contains static RTTs between 1740 DNS servers obtained from the P2PSim project [13].
- **Harvard** contains dynamic RTTs between 226 PlanetLab nodes collected in 4 hours [14].

In these datasets, as we only have RTTs between nodes in the networks, we randomly select a number of nodes as landmarks and the feature vector of each node consists of RTTs to those selected nodes. For the Harvard dataset, we extracted the static RTTs by computing the mean RTT between each pair of nodes and used them in the first two subsections for evaluating the ranking and rating accuracy. The dynamic RTTs in the Harvard dataset are used to evaluate the stability of the proximity measure only in the last subsection.

A. Ranking of Network Proximity

We first evaluate the ranking of network proximity using the Spearman's rank correlation coefficient which is defined as the Pearson correlation coefficient between the ranked variables

[15]. For two sequences X and Y, the raw data X_i and Y_i are converted to the ranks x_i and y_i in the sequence, and the Spearman's rank correlation coefficient is computed as

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 (y_i - \bar{y})^2}}. \quad (2)$$

ρ is between 1 and -1, and the larger it is, the more the ranks of X and Y are positively correlated. Thus, we evaluate the ranking accuracy by calculating the Spearman's rank correlation coefficient between the proximity measures by our service and the true latencies between network nodes. In other words, the two sequences X and Y are our proximity measures and the true latencies, and we wish a high ρ value so that their rankings match each other.

As the landmarks are randomly selected from all nodes in the network, we are interested in the impacts of the number of landmarks on the accuracy of proximity inference. We tested $k = 10$, 20, 30 and 60 respectively, with 10 runs of random landmark selection for each k, and the mean rank correlation coefficients and the standard deviations for each dataset are shown in Table I. It can be seen that the rank correlation improves with the increase of the landmark number k and that the ranking results are stable and not sensitive to the random selection of the landmarks in the network. Overall, ranking of network proximity is more accurate on the Harvard dataset than on the other two, due probably to its small size of the network, i.e. 226 nodes.

We then compare our proximity service with two popular latency prediction approaches, namely *Vivaldi* [2] and *DMFSGD* [4], [5], [6]. The former predicts RTTs based on Euclidean embedding and the latter does so based on matrix factorization. Both employ the same architecture that each node probes and exchanges messages with k randomly selected neighboring nodes in the network. In contrast, our service based on neighborhood models only probes k landmarks, with no requirement of message exchanges between nodes. To make the comparison fair, we set $k = 32$ so that all methods have the same measurement overhead. Note that $k = 32$ is the default setting in Vivaldi and DMFSGD. We ran the simulations for 10 times with random landmark and neighbor selection for our service, DMFSGD and Vivaldi respectively. The rank correlation for DMFSGD and Vivaldi is calculated by comparing the ranks of the predicted RTTs and of the true RTTs using eq. 2. The mean rank correlation coefficient and its standard deviation for each method is shown in Table II. It can be seen that our service achieved comparable results with DMFSGD and Vivaldi.

B. Rating of Network Proximity

We then evaluate the rating of network proximity that turns a proximity measure into an ordinal number in the range of $\{1, 5\}$. Ordinal rating is a loose version of ranking that labels a measure as rank 1 if it is among the top 20 percent smallest, as rank 2 if between top 20 and top 40 percent, and so on. While less informative, the advantage of rating over ranking is that rating measures are more stable over time. Thus, we compare the ratings of the proximity measures returned by our service with the ratings of the true latencies, evaluated by using the same criterion, *Root Mean Square Error* (RMSE), as in [6], given by

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}{n}}. \quad (3)$$

The smaller the RMSE, the better. We also compare our service with DMFSGD and Vivaldi. DMFSGD can directly predict ratings of RTTs, as described in [6]. For Vivaldi, we turn the predicted RTTs into ratings.

As above, we ran the simulations for 10 times with random landmark and neighbor selection for our service, DMFSGD and Vivaldi respectively, and calculated the mean RMSE and its standard deviation for each method, shown in Table III. It can be seen that on Merdian and P2PSim, DMFSGD is the best, while on Harvard, our service based on neighborhood models is the best. Overall, on the rating performance, our service achieved results at least comparable to Vivaldi.

Table IV shows the confusion matrices by our service. In these matrices, each column represents the predicted ratings, while each row represents the actual ratings. Thus, the diagonal entries represent the percentage of the correct prediction, and the off-diagonal entries represent the percentage of "confusions" or mis-ratings. For example, the entry at $(2, 2)$ represents the percentage of the rating-2 paths which are correctly predicted as rating-2, and the entry at $(2, 3)$ represents the percentage of the rating-2 paths which are wrongly predicted as rating-3, i.e. the confusions from rating-2 to rating-3. It can be seen that while there are mis-ratings, most of them have a small error of $|x_{ij} - \hat{x}_{ij}| = 1$, marked as shaded entries in the confusion matrices. This means that the mis-ratings are under control. For example, a rating-5 path may be wrongly predicted as 4, but seldom as 3, 2 or 1, since the entries at $(5, 3)$, $(5, 2)$ and $(5, 1)$ in all confusion matrices are small.

C. Stability over Latency Dynamics

We further evaluate the impact of the latency dynamics on the stability of our service using the Harvard dataset which contains $2,492,546$ dynamic RTTs with timestamps collected in 4 hours from PlanetLab [14]. In the dataset, about 94.0% of the paths between pairs of nodes are measured between 40 and 60 times.

In the experiment, we assume that each node probes the landmarks once and the first RTT measurement from each node to each landmark is put in the feature vector of the node. Thus, the proximity measures between nodes are computed using the latency information acquired in the beginning of the simulation which are not updated over time. We then evaluate how the ranking and rating accuracy is affected by the dynamics of the true RTTs in the dataset. To this end, we ran the simulations with the RTTs between nodes updated using the timestamps in the dataset and computed the Spearman's rank correlation coefficient ρ and the RMSE over time (in every three minutes) between the proximity measures and

TABLE I
IMPACT OF k ON RANKING ACCURACY.

	ρ	mean	std
Meridian	k=10	0.784	0.017
	k=20	0.802	0.010
	k=30	0.809	0.008
	k=60	0.819	0.003

	ρ	mean	std
P2PSim	k=10	0.786	0.038
	k=20	0.813	0.026
	k=30	0.820	0.016
	k=60	0.828	0.011

	ρ	mean	std
Harvard	k=10	0.942	0.020
	k=20	0.948	0.008
	k=30	0.952	0.006
	k=60	0.952	0.003

TABLE II
COMPARISON OF RANKING ACCURACY.

	ρ	mean	std
Merdian	Neighborhood	0.811	0.007
	DMFSGD	0.823	0.001
	Vivaldi	0.807	0.002

	ρ	mean	std
P2PSim	Neighborhood	0.819	0.020
	DMFSGD	0.889	0.002
	Vivaldi	0.834	0.002

	ρ	mean	std
Harvard	Neighborhood	0.952	0.006
	DMFSGD	0.910	0.003
	Vivaldi	0.868	0.002

TABLE III
COMPARISON OF RATING ACCURACY.

	RMSE	mean	std
Meridian	Neighborhood	0.943	0.014
	DMFSGD	0.860	0.003
	Vivaldi	0.945	0.003

	RMSE	mean	std
P2PSim	Neighborhood	0.922	0.058
	DMFSGD	0.667	0.004
	Vivaldi	0.879	0.003

	RMSE	mean	std
Harvard	Neighborhood	0.570	0.028
	DMFSGD	0.601	0.009
	Vivaldi	0.704	0.007

TABLE IV
CONFUSION MATRICES.

Meridian	1	2	3	4	5
1	**81%**	14%	4%	1%	0%
2	13%	**58%**	20%	7%	2%
3	3%	21%	**38%**	26%	12%
4	1%	5%	27%	**39%**	27%
5	1%	2%	12%	27%	**58%**

P2PSim	1	2	3	4	5
1	**54%**	26%	10%	10%	0%
2	40%	**37%**	16%	6%	1%
3	6%	34%	**49%**	10%	1%
4	0%	3%	24%	**54%**	18%
5	0%	0%	0%	20%	**79%**

Harvard	1	2	3	4	5
1	**77%**	21%	2%	0%	0%
2	23%	**57%**	19%	1%	0%
3	0%	21%	**64%**	13%	2%
4	0%	3%	15%	**70%**	15%
5	0%	0%	0%	17%	**83%**

the true, updated RTTs (turned into rating when computing the RMSE), shown as the curve of ρ *for Neighborhood* and of *RMSE for Neighborhood* respectively in Figure 4. In addition, we also calculated the ranking correlation ρ over time between the RTTs in the beginning of the simulation (the first measurement between each pair of nodes) and the true, updated RTTs. This measure reflects the dynamics of the RTT measurements in the dataset, shown as the ρ *for dynamic RTT* curve in Figure 4. It can be seen that the RTTs in the dataset are fairly stable, with the rank correlation over time never smaller than 0.95. In such cases, our service is able to provide accurate proximity inference, i.e. a rank correlation around 0.9 and a RMSE around 0.7, without updating the feature vector of each node for 4 hours. Note that if we do update the latencies in the feature vector of each node by a running mean, i.e. the mean of 10 most recent measurements, we only improve the accuracy slightly by less than 5%, at the cost of more probes to the landmarks.

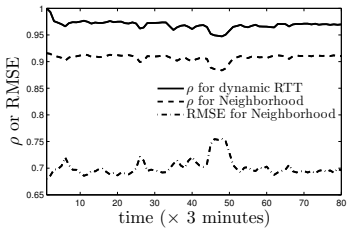


Fig. 4. Impact of latency dynamics on proximity inference. The curve of ρ for dynamic RTT represents the rank correlation over time between the RTTs in the beginning of the simulation and the updated RTTs. The curve of ρ for Neighborhood represents the rank correlation between the non-updated proximity measures and the true, updated RTTs, and that of RMSE for Neighborhood represents the RMSE between the ratings of the non-updated proximity measures and of the true, updated RTTs.

V. DEPLOYMENT ON PLANETLAB

We deployed our network proximity service on PlanetLab to test the performance when using landmarks other than those in the network.

A. Experimental Setup

We first identified the IPs that can be used as landmarks. The first landmark set, so called the *DNS* set, contains the DNS servers in the P2PSim dataset in which we found 792 out of 1740 alive and reachable. The second landmark set, so called the *WEB* set, contains the web servers at Google, Yahoo and Facebook. To extract the IPs, we pinged from each PlanetLab node to www.google.com, www.yahoo.com and www.facebook.com. As these service providers direct user requests to nearby servers, the PINGs returned us the IPs of 147 Google, 11 Yahoo, 69 Facebook web servers. Unlike the DNS set the IPs in which are largely distinct from each other, the WEB set contains similar IPs such as 74.125.237.17 and 74.125.237.18. Thus, we clustered the IPs in the WEB set by

their first 24 bits, resulted in 58 Google, 11 Yahoo and 31 Facebook IP clusters. In the experiments on the DNS set, the k landmarks are randomly selected from all 792 DNS servers, whereas on the WEB set, we first select randomly k IP clusters and each node then selects randomly an IP from each selected IP cluster.

The deployment of our service is simple and involves only the RTT measurement by PING from PlanetLab nodes to landmark nodes. For the purpose of evaluation, we also collected RTTs between PlanetLab nodes. During our experiment period between June 8th and June 12th, 2014, we were able to reach between 306 and 327 PlanetLab nodes. We confirm that the RTTs on PlanetLab are fairly stable, as shown in Section IV-C.

B. Comparison with DMFSGD and Vivaldi

As above, we compared our service based on neighborhood models with DMFSGD and Vivaldi, with $k = 32$ for all methods. For our service, we tested random landmark selection from the DNS and WEB set as well as from the PlanetLab nodes, called *Neighborhood DNS*, *Neighborhood WEB* and *Neighborhood* respectively. For each method, 10 runs of random landmark/neighbor selection were carried out using measurements collected at some time in our experiment period. We calculated the means and the standard deviations of the ranking correlation and the RMSE by comparing the results of each method with the true RTTs, shown in Table V. Note that the experiments were repeated using measurements at different times and the results were found consistent due to the stability of the RTTs on PlanetLab. It can be seen that, while worse than DMFSGD and Vivaldi, our service still achieved decent accuracies on ranking and rating of network proximity. It is worth noting that the accurate RTT prediction by DMFSGD and Vivaldi comes at the cost of 10 to 20 message exchanges between each pair of neighboring nodes. In contrast, our service is the most lightweight, with each node probing a few pre-selected landmarks only once in the beginning of the service and with no computation required.

Note that the small standard deviations in Table V show the insensitivity of our service to random landmark selection when deployed on real networks such as PlanetLab. Thus, our experiments in this and previous section suggest that empirically, random landmark selection is sufficient as long as the landmark nodes are well distributed all over the world. In such situations, each latency measurement from a node to a landmark provides useful information about the location of the node.

TABLE V
COMPARISON OF RANKING AND RATING ACCURACY ON PLANETLAB.

ρ	mean	std
Neighborhood DNS	0.894	0.027
Neighborhood WEB	0.886	0.029
Neighborhood	0.903	0.021
DMFSGD	0.936	0.013
Vivaldi	0.939	0.014
RMSE	mean	std
Neighborhood DNS	0.709	0.034
Neighborhood WEB	0.722	0.038
Neighborhood	0.701	0.044
DMFSGD	0.622	0.020
Vivaldi	0.641	0.022

VI. CONCLUSION

This paper presents a lightweight network proximity service that labels nodes with similar proximity to landmark nodes as being close to each other. The service allows the ranking and rating of network proximity which can be exploited for peer and server selection in P2P and CDN applications. Comparing to state-of-the-art latency prediction approaches such as DMFSGD and Vivaldi, the biggest advantage of our service is its simplicity. The only overhead in our service is a small number of PING measurements, and there is neither computation nor message exchange between network nodes required. Extensive simulations and real deployments show that our service can achieve accurate and stable proximity inference with a few randomly selected landmarks. Unlike many other landmark-based systems such as GNP and IDES, the landmarks in our service can be any Internet servers over which we have no experimental control.

REFERENCES

[1] T. S. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proc. of IEEE INFOCOM*, 2002, pp. 170–179.

[2] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. of ACM SIGCOMM*, Portland, OR, USA, Aug. 2004, pp. 15–26.

[3] Y. Mao, L. Saul, and J. M. Smith, "IDES: An Internet distance estimation service for large networks," *IEEE Journal On Selected Areas in Communications*, vol. 24, no. 12, pp. 2273–2284, Dec. 2006.

[4] Y. Liao, W. Du, P. Geurts, and G. Leduc, "DMFSGD: A decentralized matrix factorization algorithm for network distance prediction," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1511–1524, oct 2013.

[5] ——, "Decentralized prediction of end-to-end network performance classes," in *Proc. of CoNEXT*, Tokyo, Japan, 2011.

[6] W. Du, Y. Liao, N.Tao, P. Geurts, X.Fu, and G. Leduc, "Rating network paths for locality-aware overlay construction and routing," *IEEE/ACM Transactions on Networking*, vol. 23, no. 5, oct 2015.

[7] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. of the International World Wide Web Conference - WWW10*, Hong Kong, May 2001.

[9] B. Donnet, B. Gueye, and M. A. Kaafar, "A survey on network coordinates systems, design, and security," *IEEE Communication Surveys and Tutorial*, vol. 12, no. 4, pp. 488–503, 2010.

[10] C. Hennig and M. Kutlukaya, "Some thoughts about the design of loss functions," *REVSTAT–Statistical Journal*, vol. 5, no. 1, pp. 19–39, 2007.

[11] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys and Tutorials*, vol. 7, pp. 72–93, 2005.

[12] B. Wong, A. Slivkins, and E. Sirer, "Meridian: A lightweight network location service without virtual coordinates," in *Proc. of ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, Aug. 2005, pp. 85–96.

[13] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary Internet end hosts," in *Proc. of the ACM/SIGCOMM Internet Measurement Workshop*, Marseille, France, Nov. 2002, pp. 5–18.

[14] J. Ledlie, P. Gardner, and M. I. Seltzer, "Network coordinates in the wild," in *Proc. of USENIX Symposium on Networked Systems Design and Implementation*, Cambridge, Apr. 2007, pp. 22–35.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NJ, USA: Springer-Verlag, 2006.