Improving Prediction Accuracy of Matrix Factorization Based Network Coordinate Systems

Yang Chen[1], Peng Sun[2], Xiaoming Fu[1], Tianyin Xu[1,3]
[1]Institute of Computer Science, University of Goettingen, Germany
[2]Department of Electronic Engineering, Tsinghua University, China
[3]State Key Lab. for Novel Software and Technology, Nanjing University, China
E-mail: {yang.chen, fu, tianyin.xu}@cs.uni-goettingen.de, sp06@mails.tsinghua.edu.cn

***Abstract*—Network Coordinate (NC) systems provide a lightweight and useful way for scalable Internet distance prediction while serving as an important component in many Peer-to-Peer applications. Most of the existing NC systems utilize the Euclidean distance model, which is largely impaired by the persistent occurrence of Triangle Inequality Violation (TIV) in the Internet. Recently, matrix factorization (MF) based NC systems, which can completely remove the TIV constraint, provide an alternative approach towards better prediction accuracy. Phoenix, an NC system based on the MF model, well explores the advantage of the MF model and becomes the most accurate NC system so far. However, through experimental study, we find that the prediction accuracy of Phoenix for short links is significantly worse compared with the overall prediction accuracy. Based on the observation, we propose a new NC system, named *Pancake*, aiming at reducing the high prediction error for short links. By introducing a two-level architecture, Pancake achieves much higher prediction accuracy than other selected existing NC systems. Through extensive experiments, we demonstrate that Pancake reduces the 90th percentile relative error by up to 25.37% from Phoenix. Moreover, Pancake converges very fast and is robust to different dimension values. For further insights, we study the performance of Pancake using a dynamic data set in addition to the widely used aggregate data sets. With varying RTTs over time, Pancake outperforms other NC systems consistently.**

I. Introduction

Network Coordinate (NC) systems provide a lightweight and useful way for scalable Internet distance prediction. In an NC system, a set of coordinates is assigned to each host, based on which the distance (a.k.a., end-to-end round-trip time) between any two hosts can be calculated with a predefined distance function. For an N-host network, the NC system is able to predict the distances of all the $N \times N$ host pairs with only $O(N)$ measurement overhead. So far, NC systems serve as critical components in many large-scale Peer-to-Peer applications, e.g., application layer multicast [31] and anycast [27], server selection [31], multi-player online games [1], Azureus BitTorrent [9], PeerWise Overlay [24], and compact routing [2].

Most of the existing NC systems use the Euclidean distance model. In this model, all the N hosts are embedded into d-dimensional Euclidean space R^d ($d \ll N$). The Euclidean distance model is simple for the implementation of NC systems. However, the Euclidean distance based NC systems bear the assumption that the predicted distances among any three hosts satisfy the *triangle inequality* [32]. Unfortunately, according to a number of literatures [9], [11], [14], [28], [32], Triangle Inequality Violation (TIV) persistently occurs in the real Internet. As a result, the Internet distance cannot be predicted accurately with Euclidean distance based NC systems.

To overcome the problem of TIV, matrix factorization (MF) based NC systems such as IDES [17], Phoenix [3] and DMF [13] have been developed. According to the simulation study in [13], although there is already no TIV constraint in both IDES and DMF, neither of them has better prediction accuracy than Vivaldi [7], the most widely used NC system. Phoenix [3] introduces the concept of *weight* in the NC calculation to distinguish between the referred NCs with high errors and low errors. This approach better explores the advantage of the matrix factorization model and thus achieves the highest prediction accuracy so far. However, through our experimental study, we find that the prediction accuracy of Phoenix for short links[1] is significantly worse than its prediction accuracy for other links. We hope to take this feature of Phoenix as an opportunity for improving the prediction accuracy of MF based NC systems.

In this paper, we choose Phoenix NC as our basic reference NC algorithm since it is the most accurate NC so far. we first conduct experimental study on prediction accuracy for short links. After grouping all the hosts into different locality-based clusters, we employ an independent Phoenix NC algorithm for each cluster and find that the prediction accuracy of intra-cluster links is substantially improved compared with relying on a global Phoenix NC algorithm. Based on this observation, we design an effective two-level hierarchical NC system, called *Pancake*. Pancake uses a lightweight and decentralized approach called *binning* [26] to group all the hosts into different clusters. In each cluster, an independent local Phoenix NC algorithm is used to predict the distances of intra-cluster links. For the distances of inter-cluster links, Pancake still adopts the global Phoenix NC algorithm to do prediction. According to our extensive evaluation, compared with Phoenix, Pancake reduces the 90th percentile relative error (NPRE) by up to 25.37%. Compared with Vivaldi [7],

[1]In this paper, we define an end-to-end link with distance less than or equals to $50ms$ as a short link.

978-1-4244-7116-4/10/$26.00 ©2010 IEEE

the most widely used NC system, Pancake even reduces the NPRE by up to 45.65%. Compared with Pharos [4], another two-level NC system based on Vivaldi, Pancake reduces the NPRE by up to 31.51%. In addition, Pancake converges very fast and is robust to different values of dimensions.

Specifically, our contributions are as follows:

1) Through experimental study, we demonstrate that employing MF based NC algorithms such as Phoenix in local clusters achieves better prediction accuracy for intra-cluster links than merely relying on global NC algorithms.
2) We design a two-level NC system, called Pancake, which can significantly improve the prediction accuracy of MF based NC. Moreover, Pancake is fully compatible with existing deployments and its extra overhead is negligible.
3) We conduct extensive evaluation based on widely used real Internet data sets to demonstrate the performance gain of Pancake. For further insights, our evaluation includes the dynamic data set used in [16] which reflects the RTT variations over time for all end-to-end links. To the best of our knowledge, this is first work to consider RTT variations while studying the performance of NC systems.

The rest of the paper is organized as follows. Section II reviews the background and related work. In Section III we demonstrate that the prediction accuracy of Phoenix can be improved by running an independent local Phoenix in each cluster. Based on this observation, Section IV describes the design and implementation of the Pancake NC system. We evaluate the performance of Pancake and compare it with other existing NC systems in Section V. Section VI concludes this paper.

II. Background and Related Work

Suppose there are N Internet hosts in the network. Let M be the $N \times N$ Internet distance matrix among these N nodes. Thus $M(i,j)$ denotes the measured distance between host i and host j. In NC systems, through $O(N)$ measurements, each host can obtain one (or multiple) d-dimensional vector(s) called coordinate(s). With the coordinates of host i and host j, the distance between them can be calculated instead of performing direct end-to-end measurements. Let $M^E(i,j)$ denote the predicted distance between host i and host j. NC systems seek to minimize the following objective function (F) in a decentralized way:

$$F = \Sigma_i \Sigma_j (M(i,j) - M^E(i,j))^2 \tag{1}$$

Most existing NC systems (e.g., Vivaldi [7], GNP [18], NPS [19], ICS [15]) use the Euclidean distance model. Among these NC systems, Vivaldi [7] becomes the most widely used one because of its simplicity and decentralized feature. To date, Vivaldi has been used in many well-known Internet applications, such as Azureus BitTorrent (BitTorrent System) [9], PeerWise Overlay [24], SBON [20] (Stream Based Overlay Network), Census [6] (a platform for building large-scale distributed application), DS^2 (Internet delay space synthesizer) [29].

However, Vivaldi and other Euclidean distance based NC systems severely suffer from the widely existing TIV in the Internet. As demonstrated in [32], any three hosts with TIV cannot be embedded into Euclidean space without sacrificing some level of accuracy, since the distances among them in Euclidean space have to obey triangle inequality. However, TIV is natural and persistent in the Internet [32]. Therefore the existence of TIV causes a serious problem for any Euclidean distance based NC system [11], [28]. In other words, TIV is an irremediable problem hurting the prediction accuracy of Euclidean distance based NC systems.

To overcome the problem of TIV in the Internet, matrix factorization (MF) based NC systems are introduced. The key idea of the MF model is that a large matrix can be approximated by the product of two smaller matrices with methods such as Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF). The TIV constraint no longer exists in this model. IDES [17] is the first MF based NC system. Recently, DMF [13] is proposed to achieve a fully decentralized architecture. However, according to the study in [13], both IDES and DMF cannot achieve better prediction accuracy than Vivaldi, the most widely used Euclidean distance based NC system. To further explore the advantage of the MF model, a decentralized and more accurate NC system called Phoenix was proposed in [3]. By introducing the concept and calculation policy of *weight*, the impact of the inaccurate NC can be largely reduced and the overall prediction accuracy can be improved accordingly. Internet trace based simulation shows that Phoenix achieves higher prediction accuracy than other existing NC systems. In this paper, we choose Phoenix as the basic reference NC for our study and design a simple yet effective NC system which can achieve even higher prediction accuracy.

There are some other approaches for Internet distance prediction. Htrae [1] is a new proposal for Internet distance prediction in online gaming. It applies geographic information in bootstrapping the NC, in order to shorten the convergence time. However, a third-party service is required to maintain timely updates of the mapping between geographic information and dynamically assigned IP addresses. To satisfy the accuracy requirement, the additional overhead could be significant, sacrificing the benefits of the light-weight feature of NC systems. Thus, this approach is not chosen for our study.

The prediction accuracy of an NC system is often denoted by the relative error (RE) of the predicted distance over the measured distance. RE of the distance between host i and host j is defined as [7], [11], [17]–[19]:

$$RE = \frac{\mid M^E(i,j) - M(i,j) \mid}{min(M^E(i,j), M(i,j))} \tag{2}$$

where smaller RE indicates higher prediction accuracy. When the predicted distance equals to the measured distance, the RE value will be zero.

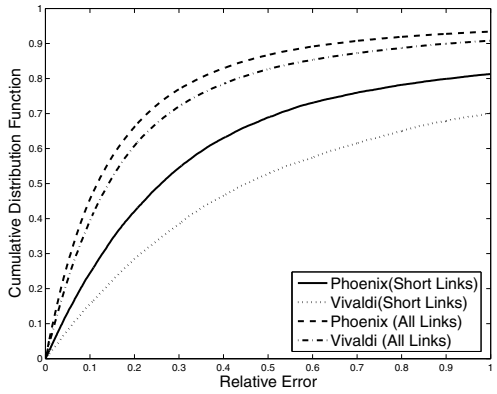


Fig. 1. Relative Error of Phoenix/Vivaldi, CDF for all links and short links

III. Study on Prediction Accuracy for Short Links

A. Prediction Accuracy for Short Links

Accurate distance prediction for short links is a challenging problem for the designers of NC systems. According to [4], the most widely used NC system, Vivaldi, suffers from the inaccurate prediction for short links. In this subsection, we study the prediction accuracy for short links in MF based NC system using Phoenix NC as a representative.

In the experimental study, we conduct a simulation based on a measured Internet distance data set called PlanetLab data set. The data set includes the round-trip times (RTTs) among 169 PlanetLab hosts and it has been used in [3], [17], [31].

Fig. 1 shows that both Phoenix and Vivaldi performs badly in predicting short links. As in [17]–[19], we pay more attention to 90th percentile relative error (NPRE) since it can guarantee 90% of the links have lower RE values than it. Smaller NPRE value indicates higher overall prediction accuracy. According to our experimental study, in terms of all links, the NPRE of Phoenix is 0.67 while that of Vivaldi is 0.92. However, when considering short links, the NPRE of Phoenix increases to 3.37 while that of Vivaldi being 5.20. In other words, although Phoenix is still able to provide much better prediction than Vivaldi for short links, its prediction accuracy for short links is significantly worse than its overall performance. In other words, if we can reduce the prediction error of short links without increasing the prediction error of other links, the overall prediction accuracy will be definitely increased.

B. Enhancement of Prediction Accuracy for Intra-cluster Links

Assume that we have a set of N hosts called S, and a subset of S called S_1 ($S_1 \subset S$). If the distance between two hosts which both belong to S_1, is to be predicted, there are two methods of using the NC algorithm. One simple method is to first run a global NC algorithm for all the N hosts, then use this globally valid NC for prediction. Alternatively, we can run an independent local NC algorithm within S_1 to predict the distance between these two hosts. Lua *et al.* [14] have found that, if a local Vivaldi NC system is deployed in S_1 for the distance prediction instead of using a global one, the prediction error will be largely reduced. However, since MF based NC systems (e.g., Phoenix) have completely different NC calculation mechanisms with Euclidean distance based NC ones, the performance difference of these two methods have not been studied yet.

TABLE I
Median distances among the nodes of the intra and inter clusters (PlanetLab Data Set)

	C_1	C_2	C_3	C_4	C_5
C_1	39.63	79.63	54.06	160.16	54.41
C_2	79.63	51.74	98.21	181.65	66.14
C_3	54.06	98.21	28.06	122.06	85.33
C_4	160.16	181.65	122.06	53.65	190.63
C_5	54.41	66.14	85.33	190.63	27.16

In this paper, we compare these two methods by employing Phoenix NC. In other words, we are going to conduct an experimental study to discuss, for a certain cluster, whether the deployment of a dedicated Phoenix NC system just for that cluster would outperform a global Phoenix NC system in predicting the distances of intra-cluster links.

For a certain data set with N hosts called S, we would first group these N hosts into u clusters in a decentralized way. We use the *binning* scheme [26] for this clustering process. The detailed process includes the following two steps:

1) The u hosts are selected as *anchors* among all N hosts, which will guide the decentralized clustering. The rest are called ordinary hosts. Each anchor represents a cluster and the cluster represented by the i-th$(1 \leq i \leq u)$ anchor is denoted as C_i. These u anchors are chosen randomly and are widely distributed, i.e., the RTTs among anchors are above a pre-defined threshold $R = 40ms$.
2) For each ordinary host, it will measure its distance to every anchor and join the cluster represented by the nearest anchor. Thus, all hosts can be grouped into u clusters in a decentralized way.

For each cluster, an independent local Phoenix NC is deployed. Then each host in this cluster can get its local Phoenix NC. This local NC can only be used for the distance prediction between any two hosts within the same cluster. At the same time, a global Phoenix is deployed for the whole S. Then each host in S can obtain its global NC, which can be used for the distance prediction between any two hosts in the network. For all intra-cluster links, the local NC system and the global one are used respectively to predict their distances. Then the relative error of these two methods are calculated and compared.

We set u as 5 to group the 169 PlanetLab hosts into 5 clusters. Table I shows the median distances in terms of millisecond of intra-cluster links and inter-cluster links. The median value of intra-cluster links are much smaller than the median value of inter-cluster links. Also, we have found that

TABLE II
NPRE OF GLOBAL/ LOCAL PHOENIX (PLANETLAB DATA SET)

NC System \ Cluster	C_1	C_2	C_3	C_4	C_5
Host Amount	16	3	63	32	50
Global Phoenix	75.01	433.26	1.17	3.14	7.08
Local Phoenix	6.70	12.53	0.82	0.74	0.58

74.25% of the short links are intra-cluster links. Therefore, if we can predict the distances of intra-cluster links with higher prediction accuracy, it will lead to more accurate prediction for short links.

Table II shows the NPRE of the global NC system and the local one in each cluster, respectively. We found that, in all the clusters, NPRE of the local NC system is much smaller than that of the global NC system. The result provides us an encouraging way of lowering the prediction error for short links.

IV. SYSTEM DESIGN OF PANCAKE

A. Architecture Overview

According to our experimental study in Section III, we observe that, for Phoenix NC system, the prediction accuracy for short links is significantly worse than that for all the links. Fortunately, we have also found that, if the hosts are grouped into clusters based on locality, running independent local Phoenix NC in each cluster would significantly improve the prediction accuracy for the intra-cluster links compared to running a global Phoenix.

Based on these two observations, we design a two-level hierarchical NC system, named as *Pancake*. In Pancake, all the hosts form two levels for coordinate calculation, namely *global overlay* and *local cluster*. Each local cluster runs an independent local Phoenix NC algorithm, and a global Phoenix NC is maintained using global overlay.

In Pancake, each host must join in both the global overlay and one local cluster to have NCs in different levels. To join the global overlay, hosts follow the process designed in [3], i.e., selecting m other hosts randomly as its global reference hosts. To join local cluster, similar to the process in Section III, the binning scheme is conducted to form all the clusters. A small fix set of hosts called *anchors* are selected to guide the grouping process. After obtaining its distances to all the anchors by performing ICMP PING, a new Pancake host can join the cluster represented by the closest anchor. It will also select m other hosts from its local cluster randomly as its local reference hosts. Afterwards, the NC can be updated periodically under the policy proposed in Section IV-B.

For each Pancake host i, its global NC is composed of two d-dimensional vectors, one is *global outgoing vector* GX_i and the other is *global incoming vector* GY_i. Accordingly, its local NC is composed of *local outgoing vector* LX_i and *local incoming vector* LY_i. Four d-dimensional vectors are assigned to each host in total. For any two hosts in Pancake NC system,

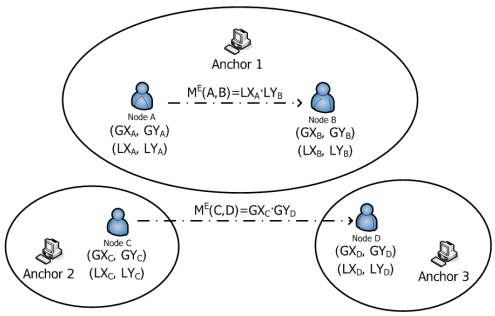


Fig. 2. Hierarchical Distance Prediction of Pancake

namely host i and host j, the distance between them can be predicted in a hierarchical way as follows.

$$M^E(i,j) = \begin{cases} LX_i \cdot LY_j & C_i = C_j \\ GX_i \cdot GY_j & C_i \neq C_j \end{cases} \quad (3)$$

The detailed prediction process is in a bottom-up fashion. If two hosts belong to the same cluster, their local NCs are used for the distance prediction. In contrast, if they belong to different clusters, their global NCs are used. Let us provide an example of the hierarchical distance prediction of Pancake. As shown in Fig. 2, both host A and host B belongs to cluster C_1, thus $M^E(A,B)$ is calculated by the dot product of LX_A and LY_B. In contrast, host C belongs to cluster C_2 and host D belongs to cluster C_3, therefore $M^E(C,D)$ is calculated by the dot product of GX_C and GY_D.

The simple architecture of Pancake implies one advantage, i.e., it is compatible with Phoenix. For every host where the Phoenix client has been deployed, it just needs to use classic Phoenix NC algorithm to join global overlay and local cluster, then calculates and updates its global and local coordinates periodically. Therefore, it is easy for a host to benefit from Pancake NC system without deploying another NC client.

B. The Pancake Algorithm

Algorithm 1 shows the process of how a new host H_{new} joins the Pancake NC system. Firstly, H_{new} measures the distances to all the anchors. Afterwards, it joins the cluster which is represented by the closest anchor. Then H_{new} will start its NC calculation, by alternately updating its global NC and local NC.

In Pancake, we maintain almost the same measurement overhead as Phoenix [2]. For a Pancake host, only one of its two NCs will be updated in each NC update round. In the odd rounds, H_{new} measures its RTTs to its reference hosts in the global overlay as well as retrieving the global incoming vectors and global outgoing vectors of these global reference hosts. Then its global NC can be calculated and updated. Correspondingly, in the even rounds, H_{new} performs measurement to its local reference hosts in the local cluster and update its local NC. We will study the detailed convergence behavior of Pancake in Section V-D.

[2]We will study the extra measurement overhead in SectionIV-C

Algorithm 1 The Pancake Algorithm

$Nearest\_Anchor\_Distance = \infty$
for i in $Anchors$ **do**
  $d(i) = Measure\ Distance\ to\ i$
  if $Nearest\_Anchor\_Distance > d(i)$ **then**
    $Nearest\_Anchor\_Distance = d(i)$
    $Nearest\_Anchor = i$
  end if
end for
$Connect\_to\_Rendezvous\_Point(RP)$
$Get\_Reference\_Host\_Candidates(RP)$
$Join\_Global\_Overlay()$
$Join\_Local\_Cluster(Nearest\_Anchor)$
while forever **do**
  //Measurement to reference hosts in global overlay
  $Get(d_G(\cdot), GX(\cdot), GY(\cdot))$
  //Update the Global NC
  $[GX_i, GY_i] = Phoenix\_Update(d_G(\cdot), GX(\cdot), GY(\cdot))$
  $Wait(Update\_Interval)$
  //Measurement to reference hosts in local cluster
  $Get(d_L(\cdot), LX(\cdot), LY(\cdot))$
  //Update the Local NC
  $[LX_i, LY_i] = Phoenix\_Update(d_L(\cdot), LX(\cdot), LY(\cdot))$
  $Wait(Update\_Interval)$
end while

C. Extra Measurement Overhead

Compared with Phoenix or any other single level NC systems, the extra measurement overhead introduced by Pancake is the measurement from each host to the anchors. For studying the introduced overhead, there are two aspects. One is from the ordinary hosts' perspective, i.e., Pancake should not increase the measurement overhead much for each ordinary host. The other one is from the anchors' perspective, i.e., the anchors should be able to serve large amount of the hosts in the system.

For each ordinary host, it just need to measure its RTTs to every anchor once per hour. Compared with the measurement overhead for the NC calculation, it is negligible.

From the anchors' perspective, unlike landmarks in GNP [18], ICS [15] or Hybrid GNP [30] which limit the scalability if they become communication bottlenecks [5], anchors require neither performing active measurements nor periodically information updating. Thus we do not need to deploy any extra software/client on the anchors, which makes the selection of anchors more flexible. Also, since the anchors just need to be able to reply the ICMP PING passively, this causes very light load to the anchors. According to the study of [26], if there are one million ordinary hosts in the system, each host need 10 PINGs to obtain a good sample of the distance to a certain anchor, and each host refresh its cluster information once per hour, the load of each anchor is approximately 2700 PINGs per second. And [26] demonstrates that such loads are easily handled by the modern midrange PCs. In other words, the binning scheme is quite manageable for systems with one million hosts or even more.

TABLE III
90TH RELATIVE ERROR (NPRE) OF PANCAKE, PHOENIX, PHAROS AND VIVALDI

Data Set \ NC System	Pancake	Phoenix	Vivaldi	Pharos
PlanetLab	0.50	0.67	0.92	0.73
King	0.33	0.41	0.50	0.44

V. PERFORMANCE EVALUATION

A. Settings

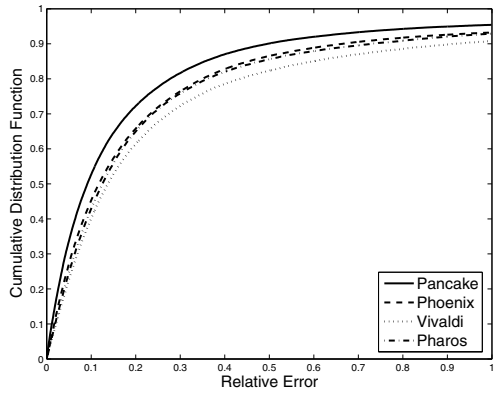
In this section, we evaluate the performance of the Pancake NC system. In the perspective of relative error in distance prediction, Pancake is compared with three typical NC systems. One is Phoenix [3] which is the basic NC system for Pancake. Another one is Vivaldi [7], the most widely used NC system so far. There are some other 2-level hierarchical NC systems proposed in the literatures (e.g., Hybrid GNP [30], Pharos [4]). We do not evaluate Hybrid GNP [30] here since it is a centralized approach (the NC calculation of the whole system relies on a fixed set of landmarks), which cannot scale well. Consequently, it has never been used in any large scale Internet application. Pharos [4] is another two-level NC system which utilizes Vivaldi as its basic NC system in each level. It has been used as distance prediction module in Proxima [27], an application layer anycast system. We choose Pharos as the third NC system to compare with Pancake.

As in [3], we use the default parameters for Phoenix and Vivaldi. All the NC systems use 8-dimensional NC. In Phoenix and Pancake, a host measures 32 reference hosts in one update round. In both Pancake and Pharos, five randomly selected anchors are used. Ten independent runs are conducted on each data set and the average results are reported.

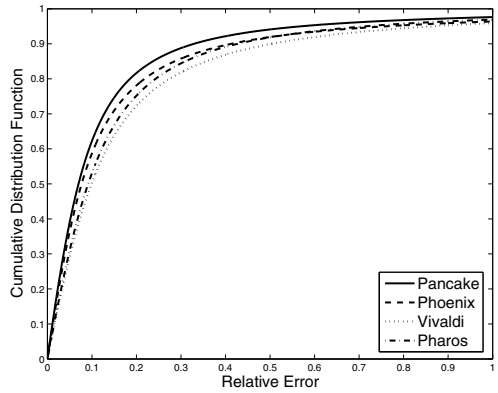
Our simulation is based on two widely used Internet distance data sets. Besides PlanetLab data set that mentioned above, we also use King data set. This data set is derived from the end-to-end RTTs among about 2000 Internet DNS servers, which are collected by the P2PSim project [12] using King tool [8]. As in [10], [11], we remove partial measurements to derive a 462×462 complete and square distance matrix.

B. Relative Error of Distance Prediction

Fig. 3 shows the comparison among these four NC systems using the two representative data sets. According to the simulation results, we can find that Pancake achieves much higher prediction accuracy than other existing NC systems. Table III shows the NPRE values of the four NC systems on two different data sets. Compared with Phoenix, Pancake reduces the NPRE by between 19.51% (King data set) and 25.37% (PlanetLab data set). Compared with Pharos, Pancake reduces the NPRE by between 25.00% (King data set) and 31.51% (PlanetLab data set). Compared with Vivaldi, Pancake reduces the NPRE by between 34.00% (King data set) and

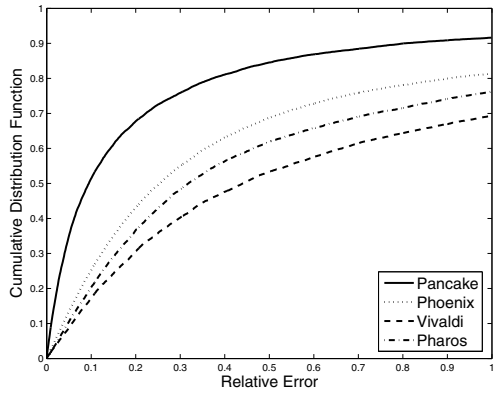


(a) PlanetLab

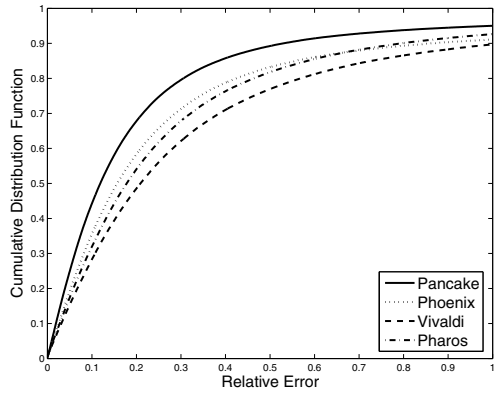


(b) King

Fig. 3. CDF of Relative Error (Dimension = 8)



(a) PlanetLab



(b) King

Fig. 4. CDF of Relative Error for Short Links (Dimension = 8)

45.65% (PlanetLab data set). Therefore, Pancake improves the prediction accuracy of Phoenix even further.

Fig. 4 demonstrates the comparison of the prediction accuracy for short links between Pancake and the other three NC systems. Obviously, Pancake achieves much better prediction accuracy than other NC systems for short links in all three data sets. This result confirms our intuition of designing Pancake.

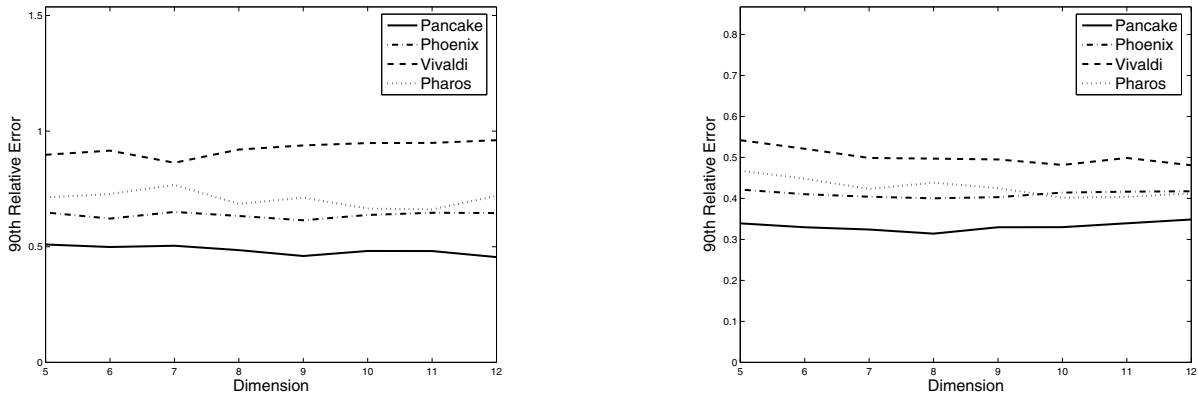
C. Different Values of Dimension

In the study of prediction accuracy, the dimension of every NC system is fixed to 8. In this subsection, the value of the dimension of NC systems is varied from 5 to 12 (higher dimension is rarely used in NC systems). Fig. 5 shows the average 90th percentile relative error of all the four NC systems with different NC dimension values. We can see that Pancake consistently outperforms all the other related NC systems. Therefore, Pancake is robust to different dimension settings.

D. Convergence Behavior of Pancake

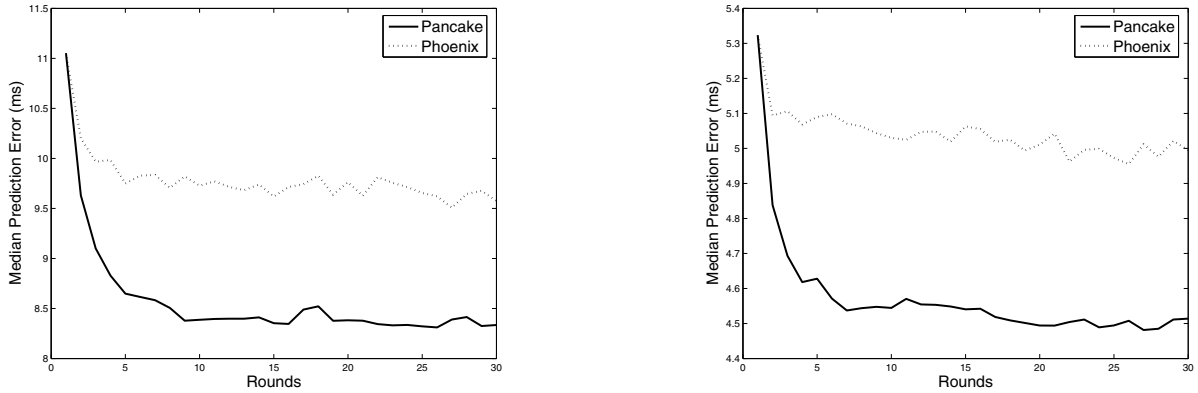
Similar to Vivaldi, Phoenix or any other simulation-based NC system [9], a Pancake host updates its NC periodically and the accuracy of its NC grows until it reaches a stabilized status, i.e., the prediction accuracy stops increasing. Same as previous literatures [7], [21], we call this procedure as a *convergence procedure*. Besides the final stabilized prediction accuracy, for a new joined host, how fast its NC can reach its stabilized status is also critical. If it takes a new host a lot of measurements to reach its stabilized status, it still can not be used effectively in real Internet applications. A good NC system should converge fast.

In this subsection, we study the convergence property of Pancake and compare it with Phoenix. We introduce the scenario used in [21]. For a data set with N hosts, we introduce 20 new hosts into the network after the remaining $N - 20$ hosts have stabilized. Then we can see the evolution of the NC convergence behavior of these 20 hosts. The metric *median prediction error*, defined as $median_{i,j}(\|D^E(i,j) - D(i,j)\|)$, is introduced in [7] to evaluate the convergence behavior of the NC systems. In Phoenix, in each update round, its NC will be updated once. To guarantee a fair comparison, in each update round, as described in Section IV-B, Pancake will only update either its global NC or its local NC, instead of updating both of them. Therefore in each update round, the measurement overhead of Pancake is the same as Phoenix.



(a) PlanetLab (b) King

Fig. 5. 90th Percentile Relative Error under Different Values of Dimension



(a) PlanetLab (b) King

Fig. 6. Convergence Behavior of Pancake

According to Fig. 6, we can see after the first round, Pancake converges even faster than Phoenix. Also, the stabilized prediction error of Pancake is much smaller than Phoenix. Basically Pancake will converge in less than 10 update rounds.

E. Evaluation through Dynamic Data Set

In the previous NC related work, people only use aggregate data sets which combine measurements taken at different times for simulation. In these data sets, the final RTT between two hosts is obtained by taking the median [7] or minimum value of measured RTTs [23], [29], [32] over long periods of time (days or even weeks). Obviously, if we aggregate all the different RTT values of a certain end-to-end link, the variation will not be considered at all. However, the variation of RTTs is not negligible. For example, in "king200-allpairs-1h" data set used in [16], 12% of the pairs have standard deviations of more than $100ms$. Therefore these aggregate data set will reveal some characters of the Internet delay structure. For example, Luzmezanu *et al.*, [16] demonstrate that the number of TIVs varies over time and the TIVs present during the measurement are not necessarily preserved by relying on aggregate data sets. Instead of using aggregate data sets, they have collected data sets which preserve the dynamic properties of RTTs. However, there is still no study about using dynamic data sets to evaluate an NC system so far.

We use the "king200-allpairs-1h" data set for our evaluation since it has the smallest time granularity and also because it exhibits the greatest variability. This data set includes 200 hosts and all pairs of these hosts are measured at least once within every hour. The duration of the data collection is 44 hours. We compare Pancake with Phoenix, Vivaldi and Pharos using this dynamic RTT data set. This result shows the performance of each NC system under dynamic RTTs.

Fig. 7 shows the average 90th percentile relative error of all the four NC systems in different time intervals. We can find the Pancake and Phoenix performs much better than Pharos and Vivaldi. And Pancake performs the best. The average NPRE value of Pancake, Phoenix, Pharos and Vivaldi is 0.42, 0.49, 0.82, 0.98, respectively. Precisely, Pancake reduces the average NPRE from Phoenix 14.29%. Also, Pancake reduces the average NPRE from Pharos by 48.78% and the NPRE from Vivaldi by 57.14%. These results provide more convincing evidence about the performance gain of Pancake in addition to the aggregate data sets.

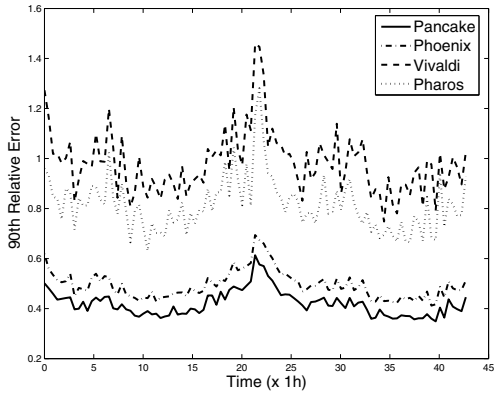


Fig. 7. Relative Error for the "K200-allpairs-1h" Data Set

VI. Conclusion and Future Work

In this paper, we focus on further improving the prediction accuracy of matrix factorization based NC systems. Our study is based on Phoenix, the most accurate NC system so far. Our experimental study demonstrates that running local Phoenix NC algorithm in a cluster can largely improve the prediction accuracy for short links. Based on this observation, we design and implement a two-level hierarchical NC system, named Pancake. This simple yet effective NC system can make better prediction while introducing negligible measurement overhead for each host. Moreover, Pancake is fully compatible with existing deployments. According to our extensive experimental study, by improving the prediction accuracy for short links, Pancake performs better in terms of prediction accuracy than Phoenix. Pancake can reduce the NPRE by up to 25.37% compared with Phoenix. As a two-level NC system, Pancake also outperforms Pharos in terms of a reduced NPRE of up to 31.51%. Moreover, Pancake is robust to different values of the dimension. Besides the aggregate data sets, we use a dynamic data set which reflects the variations of RTTs to evaluate the performance of NC for the first time. Our experimental results demonstrate that even with varying RTTs over time, Pancake outperforms other NC systems consistently.

As our future work, we will integrate Pancake into a global application layer anycast system to serve as the basic distance prediction module for locality-aware Peer-to-Peer applications. Also, we will incorporate security mechanism into the Pancake deployment to avoid the malicious attacks.

Acknowledgment

Thanks to Cristian Lumezanu for providing the dynamic RTT data sets in his ACM IMC'09 paper.

References

[1] S. Agarwal and J. Lorch. Matchmaking for Online Games and Other Latency-Sensitive P2P Systems. In Proc. of ACM SIGCOMM, 2009.

[2] I. Abraham and D. Malkhi. Compact Routing on Euclidian Metrics. In Proc. of ACM PODC, 2004.

[3] Y. Chen, X. Wang, et al. Phoenix: Towards an Accurate, Practical and Decentralized Network Coordinate System. In Proc. of IFIP Networking, 2009.

[4] Y. Chen, Y. Xiong, et al. Pharos: A Decentralized and Hierarchical Network Coordinate System for Internet Distance Prediction. In Proc. of IEEE GLOBECOM, 2007.

[5] M. Costa, M. Castro, A. Rowstron, et al. PIC: Practical Internet Coordinates for Distance Estimation. In Proc. of IEEE ICDCS, 2004.

[6] J. Cowling, D. R. K. Ports, et al. Census: Location-Aware Membership Management for Large-Scale Distributed Systems. In USENIX ATC 2009.

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

[8] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In Proc. of the 2nd Usenix/ACM SIGCOMM Internet Measurement Workshop (IMW), 2002.

[9] J. Ledlie, P. Gardner, and M. Seltzer. Network Coordinates in the Wild. In Proc. of NSDI, 2007.

[10] S. Lee, S. Sahu, and D. Saha. Design and Evaluation of a Network Distance Based Planning Service. In Proc. of IEEE GLOBECOM, 2006.

[11] S. Lee, Z. Zhang, et al. On Suitability of Euclidean Embedding of Internet Hosts. In Proc. of ACM SIGMETRICS, 2006.

[12] P2PSim: A Simulator for Peer-to-Peer Protocols. http://pdos.csail.mit.edu/p2psim/

[13] Y. Liao, P. Geurts and G. Leduc. Network Distance Prediction Based on Decentralized Matrix Factorization. In Proc. of IFIP Networking, 2010.

[14] E. K. Lua, T. Griffin, et al. On the Accuracy of Embeddings for Internet Coordinate Systems. In Proc. of ACM IMC, October 2005.

[15] H. Lim, J. C. Hou, and C. H. Choi. Constructing Internet Coordinate System Based on Delay Measurement. IEEE/ACM Transactions on Networking, 2005, 13(3): 513-525.

[16] C. Lumezanu, R. Baden, et al. Triangle Inequality Variations in the Internet. In Proc. of ACM IMC, 2009.

[17] Y. Mao, L. Saul, and J. M. Smith. IDES: An Internet Distance Estimation Service for Large Network. IEEE Journal on Selected Areas in Communications (JSAC), 2006, 24(12): 2273-2284.

[18] T. S. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In Proc. of IEEE INFOCOM, 2002.

[19] T. S. E. Ng and H. Zhang. A Network Positioning System for the Internet. In Proc. of USENIX ATC, 2004.

[20] P. Pietzuch and J. Ledlie. Network-aware operator placement for stream-processing systems. In Proc. of IEEE International Conference on Data Engineering (ICDE), 2006.

[21] M. Sherr, M. Blaze, and B. T. Loo. Veracity: Practical Secure Network Coordinates via Vote-based Agreements. In Proc. of USENIX ATC, 2009.

[22] G. Wang and T. S. E. Ng. Distributed Algorithms for Stable and Secure Network Coordinates. In Proc. of ACM IMC, 2008.

[23] B. Wong, A. Slivkins, and E.G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In Proc. of ACM SIGCOMM, 2005.

[24] C. Lumezanu, R. Baden, et al. Symbiotic Relationships in Internet Routing Overlays. In Proc. of NSDI, 2009

[25] P. Pietzuch, J. Ledlie, et al. Network-Aware Overlays with Network Coordinates, In Proc. of International Workshop Dynamic Distributed Systems (IWDDS), 2006.

[26] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In Proc. of IEEE INFOCOM, 2002.

[27] G. Wang, Y. Chen, et al. Proxima: Towards Lightweight and Flexible Anycast Service. In Proc. of IEEE INFOCOM Student Workshop, 2009.

[28] G. Wang, B. Zhang, and T. S. E. Ng. Towards Network Triangle Inequality Violation Aware Distributed Systems. In Proc. of ACM IMC, 2007.

[29] B. Zhang, T. S. E. Ng, et al. Measurement-Based Analysis, Modeling, and Synthesis of the Internet Delay Space. In Proc. of ACM IMC, 2006.

[30] R. Zhang, Y. C. Hu, X. Lin, et al. A Hierarchical Approach to Internet Distance Prediction. In Proc. of IEEE ICDCS, 2006.

[31] R. Zhang, C. Tang, et al. Impact of the Inaccuracy of Distance Prediction Algorithms on Internet Applications: an Analytical and Comparative Study. In Proc. of IEEE INFOCOM, 2006.

[32] H. Zheng, E. K. Lua, et al. Internet Routing Policies and Round-Trip-Times. In Proc. of PAM, 2005.