Phoenix: A Weight-Based Network Coordinate System Using Matrix Factorization

Yang Chen, *Member, IEEE,* Xiao Wang, Cong Shi, Eng Keong Lua, *Member, IEEE,* Xiaoming Fu, *Senior Member, IEEE,* Beixing Deng, and Xing Li

***Abstract*—Network coordinate (NC) systems provide a lightweight and scalable way for predicting the distances, i.e., round-trip latencies among Internet hosts. Most existing NC systems embed hosts into a low dimensional Euclidean space. Unfortunately, the persistent occurrence of Triangle Inequality Violation (TIV) on the Internet largely limits the distance prediction accuracy of those NC systems. Some alternative systems aim at handling the persistent TIV, however, they only achieve comparable prediction accuracy with Euclidean distance based NC systems. In this paper, we propose an NC system, so-called Phoenix, which is based on the matrix factorization model. Phoenix introduces a *weight* to each reference NC and trusts the NCs with higher weight values more than the others. The weight-based mechanism can substantially reduce the impact of the error propagation. Using the representative aggregate data sets and the newly measured dynamic data set collected from the Internet, our simulations show that Phoenix achieves significantly higher prediction accuracy than other NC systems. We also show that Phoenix quickly converges to steady state, performs well under host churn, handles the drift of the NCs successfully by using regularization, and is robust against measurement anomalies. Phoenix achieves a scalable yet accurate end-to-end distances monitoring. In addition, we study how well an NC system can characterize the TIV property on the Internet by introducing two new quantitative metrics, so-called RE_{RPL} and AE_{RPL}. We show that Phoenix is able to characterize TIV better than other existing NC systems.**

***Index Terms*—Peer to peer computing, Internet topology, network coordinate system, triangle inequality violation, network monitoring.**

I. Introduction

A number of latency-conscious Internet applications require an efficient, scalable and light-weight service to predict the distance (a.k.a. Round Trip Time or RTT) between any two Internet hosts without explicit measurements. Examples include network monitoring [36], application layer multicast [25], [51], locality-aware server selection [51], distributed query optimization [33], BitTorrent-based file sharing [38], network modeling [49], compact routing [1], multiplayer online games [2], etc. Clearly, for a network with N hosts, active probing for all pairwise link distances which requires $O(N^2)$ measurements is not a scalable solution. Such measurements would impose heavy load on all the hosts. Hence, the research community has developed various network coordinate (NC) systems [11] that predict the distances of all N^2 end-to-end links, by performing only $O(N)$ measurements. NC systems reduce the active probing overhead significantly, which is especially beneficial to large-scale distributed applications. Besides the aforementioned applications, recently the application venues of NC systems have been expanded to cloud services [44] and social graph analysis [7], [52].

Every NC system relies on a *basic model*, which defines the relationship between the coordinates and the predicted distances. Most of the existing NC systems (such as GNP [31], Vivaldi [10], ICS [23], VL [41], NPS [30], PIC [8]), rely on the *Euclidean distance model*. These NC systems assign each host with a coordinate to represent its position in the Euclidean space, and predict distances between any two hosts by their coordinates. Unfortunately, such systems have a common drawback, i.e., the predicted distances among every three hosts have to satisfy *triangle inequality*, a condition that does not always hold in measured distances. Several existing studies [16], [19], [24], [42], [53] reported the wide existence of Triangle Inequality Violations (TIV) on the Internet. The wide existence of TIV leads to limited prediction accuracy of Euclidean distance model based NC systems. Consequently, the NC systems based on this model, let it be Vivaldi, PIC, NPS, or GNP, achieve similarly low prediction accuracy.

To overcome the TIV problem, which cannot be handled well by using Euclidean distance model based NC systems, the definition of *matrix factorization* based NC system is introduced in [29]. The key idea of matrix factorization based systems is that a large distance matrix can be approximately factorized into two smaller matrices by methods such as Singular Value Decomposition (SVD) or Non-negative Matrix Factorization (NMF) [18]. Thanks to the linear dependence among the rows in Internet distance matrices [23], [29], [41], such methods achieve accurate approximations. Different from Euclidean distance based NC systems, the predicted distances obtained by matrix factorization based NC systems do not have to satisfy the triangle inequality. This offers an opportunity to

Manuscript received October 17, 2010; revised April 12, 2011. The associate editor coordinating the review of this paper and approving it for publication was L. Deri.

This work was supported in part by the National Science Foundation of China (No. 60473087, No. 60703052, No. 60850003), and the National Basic Research Program of China (No. 2007CB310806). The preliminary result of this paper was published in the Proc. of the 8th International IFIP-TC6 Networking Conference (Networking'09).

Y. Chen and X. Fu are with the Institute of Computer Science, Georg-August-University of Goettingen, Germany (e-mail: {yang.chen, fu}@cs.uni-goettingen.de).

X. Wang is with the Department of Computer Science and Engineering, University of Washington, USA.

C. Shi is with the College of Computing, Georgia Institute of Technology, USA.

E. K. Lua is with the College of Engineering, Carnegie Mellon University, USA.

B. Deng and X. Li are with the Department of Electronic Engineering, Tsinghua University, China.

Digital Object Identifier 10.1109/TNSM.2011.110911.100079

overcome the accuracy barrier introduced by TIV. Prior to this work, IDES [29] is the first matrix factorization model based NC system. Recently, another matrix factorization model based NC system, so-called DMF [22], is proposed using a decentralized architecture. We discuss and evaluate these existing systems in detail in Section II-C and Section II-D. According to our study, these systems cannot achieve higher prediction accuracy than Euclidean distance model based NC systems. This result is because of the propagation of the prediction error. In both IDES and DMF, every host refers to a set of reference NCs for calculating its own NC. Concretely, all reference NCs have equal impact in the calculation. Once the inaccurate NCs are referred, their errors will be propagated to the hosts which refer to them, and could be propagated even further by misleading the NCs of these hosts. Handling this issue is the main goal of our work. The intuition behind our approach is as follows: the more accurate the reference NC is, the higher confidence (i.e., weight) should be assigned to this NC, and vice versa. Therefore, the impact of relatively inaccurate reference NCs is reduced by lower assigned weights. We will examine the effectiveness of our approach and present the detailed NC system design which can significantly improve the prediction accuracy over existing approaches.

The key contributions of this paper are threefold.

- We introduce a *weight*-based mechanism for the calculation of the matrix factorization based NC. The weight concept is introduced here to distinguish the reference NCs with high errors and low errors. Base on this mechanism, a fully decentralized NC system using matrix factorization, so-called Phoenix, is proposed.
- We evaluate Phoenix extensively from two angles, i.e., overall prediction accuracy and practicality. By studying the cumulative distribution function (CDF) of relative errors, we show that Phoenix improves the prediction accuracy significantly over existing approaches. For practicality, we show that Phoenix can handle the drifts of the NCs well by utilizing regularization, while dealing with host churn using a low sampling rate. Moreover, it achieves fast convergence and robustness against measurement anomalies. We also evaluate Phoenix in a scalable network monitoring scenario, using a newly collected dynamic data set [27] which captured the RTT variations over time. According to our evaluation, Phoenix can reduce the monitoring cost by 83.92% while maintaining the 90th percentile relative error (NPRE) as 0.56. Phoenix reduces the average NPRE from Vivaldi by 35.63% and from IDES by 58.82% during the data collection period.
- We study how well an NC system can characterize the wide existence of TIV on the Internet by introducing two new quantitative metrics, so-called RE_{RPL} and AE_{RPL}. We find that RE_{RPL} and AE_{RPL} of Phoenix are lower than the other selected NC systems, which means that Phoenix can characterize the TIV property better than other existing approaches.

The rest of this paper is organized as follows. In Section II, we review and evaluate the related work. Section III presents our design of Phoenix. We evaluate the performance of Phoenix and compare it with two existing approaches through extensive simulations in Section IV. In Section V, we perform analysis of TIV characterization in NC systems. We conclude the whole paper in Section VI.

II. Related Work

A. Network Coordinate (NC) Systems – An Overview

In NC systems, the predicted distance between two hosts is calculated based on their corresponding NCs. The difference of various NC systems primarily relies on the different basic models they use. They can be classified into 3 categories: Euclidean distance model, matrix factorization model [29], and Hyperbolic space model [28]. Table I lists some existing NC systems and their corresponding models. Most systems use the Euclidean distance model. Concretely, some of them (such as GNP [31], VL [41], ICS [23]) are centralized, requiring a fixed set of dedicated *landmarks* to orient the NC calculation of the whole system. These dedicated landmarks are the bottleneck of the entire system. In contrast, decentralized NC systems such as Vivaldi [10], NPS [30], PIC [8] are scalable and work well for large-scale applications. To our knowledge, Vivaldi is the most widely used NC system due to its clean and decentralized architecture. It has been deployed in many well-known Internet systems, such as Peerwise Overlay [26], Census [9], Bamboo DHT [35], Stream-Based Overlay Network (SBON) [33] and Azureus BitTorrent [38].

There are some approaches to improve the prediction accuracy of these NC systems. They can be classified into two classes. The approaches in the first class still utilize the basic Euclidean distance model. Adding extra elements to existing NC systems may provide a potential means for modeling the Internet distance properties, i.e., adding heights to *Vivaldi* [16], [38], or assisted with heuristic neighbor selection, i.e., *dynamic neighbor Vivaldi* [42]. We will evaluate these approaches in detail in Section II-D and demonstrate that these approaches do not improve the prediction accuracy much from basic Vivaldi. The approaches in the second class utilize a completely different basic model. For example, matrix factorization based NC systems [22], [29] are designed to provide the possibility to overcome the TIV problem. We choose a well known matrix factorization based NC system, so-called IDES [29], to compare with our solution. Hyperbolic Vivaldi [28] is not chosen here for our comparison, since it only outperforms Vivaldi when being used to estimate distances between closer nodes but underestimates large latencies ($>$ 100ms).

Besides the systems listed in Table I, there are some other ways to improve the accuracy of NC systems. One way is utilizing some third-party information. Htrae [2] uses geographic information to bootstrap the NC calculation, thus shortening the convergence time. However, it requires a third-party entity to keep timely updates about the mapping between geographic information and dynamically assigned IP addresses. Depending on the accuracy requirement, the additional overhead may be significant, thus sacrificing the benefits of light-weightiness of NC systems. Also, the accuracy of the geographic information is determined by the third-party service and there is no means for the NC system itself to verify or enhance the accuracy of this information. Another

TABLE I: Overview of existing NC systems

System	Centralized/Decentralized	Basic Model	Applications
GNP [31]	Centralized	Euclidean distance	Google CDN [39]
Vivaldi [10]	Decentralized	Euclidean distance	SBON [33], Azureus BitTorrent [38], Bamboo DHT [35], Network Modelling [49], Peerwise Overlay [26], Census [9]
ICS[23]	Centralized	Euclidean distance	-
VL[41]	Centralized	Euclidean distance	-
NPS[30]	Decentralized	Euclidean distance	-
PIC[8]	Decentralized	Euclidean distance	Pastry[8]
IDES[29]	Centralized (with relaxation option)	Matrix factorization	-
DMF[22]	Decentralized	Matrix factorization	-
Hyperbolic Vivaldi[28]	Decentralized	Hyperbolic space	-

way is to introduce hierarchy in NC systems, for example, hierarchical GNP [50], Pharos [5], hierarchical Vivaldi [14], Toread [54] use a two-level hierarchy to improve the prediction accuracy of the short links of GNP or Vivaldi. However, a dedicated grouping infrastructure should be introduced and operated, and each host needs to maintain two different NCs (one for global level and one for local level) instead of just calculating one NC for single layer NC systems. In this paper, our discussion focuses on the NC system relying on the basic $O(N)$ measurements.[1]

Suppose there are N Internet hosts in the system. Let D be the $N \times N$ distance matrix of them, while $D(i,j)$ represents the measured distance between host H_i and host H_j and $D^E(i,j)$ denotes the predicted distance between them. NC systems seek to minimize the following objective function (F) in a scalable way, where F denotes the sum of the squares of the absolute prediction error of all N^2 links.

$$F = \sum_{i=1}^{N}\sum_{j=1}^{N}(D(i,j) - D^E(i,j))^2 \quad (1)$$

The prediction accuracy of an NC system is often evaluated by the relative error (RE) of predicted distance over the measured distance on the Internet. Relative Error (RE) of the end-to-end link between host H_i and host H_j is defined as [10], [19], [20], [29]–[31]:

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

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

B. Euclidean Distance Based NC Systems and Triangle Inequality Violation (TIV)

The Euclidean distance model is the most widely used basic model in designing NC systems. Basically, an Euclidean distance based NC system embeds N hosts into a d-dimensional Euclidean space R^d. Defining x_i as the NC of host H_i, we have $x_i = (r_1^i, r_2^i, ..., r_d^i), r_k^i \in R, 1 \leq k \leq d$. Then x_i and x_j can be used to predict the RTT between host H_i and host H_j. Accordingly, $D^E(i,j)$ is defined as $D^E(i,j) = \|x_i - x_j\| = \sqrt{\sum_{1\leq k\leq d}(r_k^i - r_k^j)^2}$.

For any three hosts (namely A, B and C), let us consider the triangle ABC. Suppose AB is the longest edge of the triangle. If $D(A,B) > D(A,C) + D(C,B)$, ABC is called a TIV, due to the violation of the triangle inequality. As demonstrated in [53], any three hosts with TIV cannot be embedded into the Euclidean space within some level of accuracy, for the distances among them in Euclidean space must obey triangle inequality. However, the existence of TIV is natural and persistent on the Internet [53], which causes a serious problem for Euclidean distance model based NC systems [19], [42]. In other words, TIV can substantially impair the prediction accuracy of Euclidean distance model based NC systems. [21] proposes a systematic approach to detect TIV in NC systems. [43] and [26] utilize the TIV information for the malicious hosts detection and overlay routing. However, none of these works has proposed any solution to improve the prediction accuracy of NC systems. [42] proposes a TIV alert mechanism that can help an NC system to identify edges with severe TIVs and tries to improve Vivaldi by performing TIV aware neighbor selection. We will evaluate this approach in detail in Section II-D.

To substantially improve the prediction accuracy of Euclidean distance based NC systems, the wide existence of TIV on the Internet needs to be carefully considered. In other words, the basic model should not require the predicted Internet distances among each three hosts to satisfy the triangle inequality. Matrix factorization model is one desirable model for characterizing Internet distance matrix without the constraints of TIV. In the next subsection, we will discuss existing matrix factorization based NC systems.

C. Matrix Factorization Based NC Systems

Different literatures [23], [29], [41] are interested in the linear dependence among the rows in Internet distance matrix. Intuitively, two nearby hosts would have similar distances to

[1]We proposed a two-level hierarchy NC system, so-called Pancake, which utilizes Phoenix as its basic NC in [4], this approach improves the prediction accuracy of Phoenix further, but it is out of the scope of this paper.

all the other hosts in the system, the corresponding two rows in the distance matrix will be nearly the same. Moreover, there might be several rows in the distance matrix can approximately be expressed as a linear combination of other rows. In [41], by studying a couple of collected Internet distance matrices, in each of them the first n ($n << N$) singular values are significantly larger than the rest. This shows the linear dependence among the rows and motivates the factorization of Internet distance matrix, i.e., for a system with N Internet hosts, the $N \times N$ Internet distance matrix D can be factorized into two smaller matrices. $D \approx XY^T$ where X and Y are $N \times d$ matrices ($d << N$). As mentioned in [29], this matrix factorization is essentially a problem of linear dimensionality reduction. Y stores d basic vectors and X stores the linear coefficients. Then $D^E = XY^T$ becomes the corresponding predicted Internet distance matrix of D. Therefore, in matrix factorization based NC systems, Eq.(1) can be rewritten as the following

$$F = \sum_{i=1}^{N}\sum_{j=1}^{N}(D(i,j) - \vec{X_i} \cdot \vec{Y_j})^2 \quad (3)$$

Eq.(3) shows the overall minimization object of the matrix factorization based NC systems, which is going to be solved in a scalable way. In these systems, two d-dimensional row vectors are assigned to each host. For host i, $\vec{X_i}$ is the $outgoing\ vector$ and $\vec{Y_i}$ is the $incoming\ vector$. The predicted distance from host H_i to host H_j is simply the dot product between the outgoing vector of host i and the incoming vector of host j, defined as $D^E(i,j) = \vec{X_i} \cdot \vec{Y_j} = \sum_{k=1}^{d} X(i,k) \cdot Y(j,k)$.

Besides our approach, there are two other matrix factorization model based NC systems, IDES and DMF. IDES [29] is the first known matrix factorization based NC system. The basic architecture of IDES is a centralized NC system. A small set of dedicated landmark hosts are deployed to orient the NC calculation of all other hosts. To improve the scalability, a *relaxation* architecture is proposed. In this architecture, any host with computed NCs can serve as landmarks for other hosts. DMF [22] is a recently proposed matrix factorization based NC system. Compared with IDES, it has a fully decentralized architecture. There are two major issues with these two systems, and unlike Vivaldi, they have rarely been used by real Internet applications.

Overall prediction accuracy is the most important issue. According to the evaluation results in [22], DMF performs similarly to basic Vivaldi. According to our evaluation in Section II-D, IDES also performs comparably to basic Vivaldi. In other words, although the TIV constraint no longer exists in the matrix factorization model, comparing with basic Vivaldi, these two NC systems do not obtain performance gain regarding prediction accuracy. The propagation of the prediction error plays a big negative role in this result. For a certain host, it uses the NCs of reference hosts and its distances to reference hosts to calculate its own NC. In current NC calculation policy of both IDES and DMF, all reference NCs have equal impact for calculating the final NC. However, the accuracy of reference NCs can differ very much, some reference NCs are more accurate while some are inaccurate. Once the inaccurate NCs are referenced, their errors will mislead the hosts referring to them, and the impaired NCs could propagate the prediction errors even further. To handle this, we introduce a weight-based model to distinguish different reference NCs and thus reduce the negative impact of the inaccurate reference NCs in the new NC calculation. We will describe the details of our model in Section III-C and Section III-D.

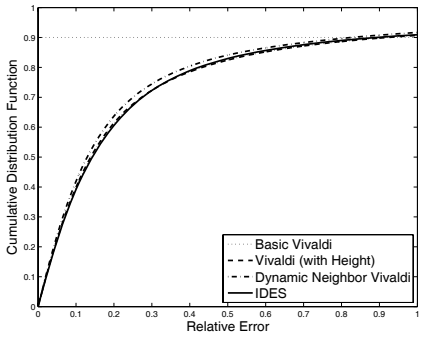
Another issue is that DMF and the basic implementation of IDES produce negative predicted distances, which is not realistic since the distance stands for RTT on the Internet. Negative predicted distance is harmful to many practical applications as they are based on the assumption of non-negative distances. Even when the percentage of negative distances is relatively small, their impact on the whole application can be severe. Let us consider a example in NC based overlay multicast [51]. A typical NC application in overlay multicast is the construction of Minimum Spanning Tree (MST) using NC predicted distance [51]. If we use Dijkstra algorithm to construct MST, even one negative distance can destroy the entire algorithm. As shown in Table I, none of the existing NC applications is based on NC systems which may produce negative distances. To be compatible with existing applications, in our new NC system proposed in this paper, it never gives negative value for the predicted distances.

IDES has an alternative non-negative option to guarantee the predicted distance to be positive. [29] denotes that their simulation results did not reveal any significant difference between this option and the basic implementation of IDES from the perspective of RE. In this paper, we select this alternative non-negative IDES to compare with other NC systems. Since DMF produces some negative predicted distances and achieves very similar prediction accuracy as basic Vivaldi [22], we do not choose it for detailed evaluation in this paper.

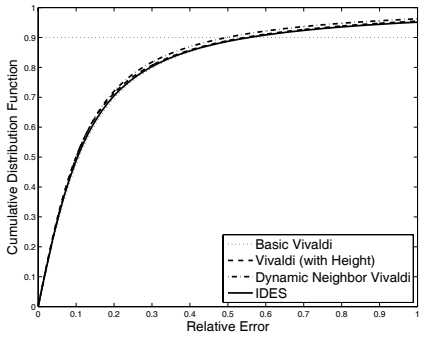
D. Prediction Accuracy of Existing Approaches

In this subsection, we study the prediction accuracy of selected existing approaches, i.e., Vivaldi and IDES. Besides the basic Vivaldi algorithm, there are two well known alternative implementations of Vivaldi. The first approach called Vivaldi (with height) introduces a *height* element to each coordinate, in which the height models the distance which takes packets to travel the access link from the host to the core Internet[10]. Vivaldi (with height) is implemented and deployed in Pyxida [16], which is the most widely deployed Vivaldi implementation [38]. The second approach is *dynamic neighbor Vivaldi* [42]. This approach focuses on refining the neighbor set of each host by removing some neighbors which may cause severe TIVs. According to the evaluation in [42], this approach performs better than basic Vivaldi in closest neighbor selection. However, besides this specified application, the overall prediction accuracy of dynamic neighbor Vivaldi has not been studied. In all, we will evaluate the prediction accuracy of four NC implementations, i.e., basic Vivaldi, Vivaldi (with height), dynamic neighbor Vivaldi, and IDES.

There are several Internet distance data sets used in related papers. In this paper, we focus on two representative and widely used data sets. The first data set is the PL data set, which includes the RTTs among 169 PlanetLab hosts all over



(a) PL data set



(b) King data set

Fig. 1: CDF of relative error of basic Vivaldi & Vivaldi (with height) & dynamic neighbor Vivaldi & IDES.

the world. PL data set has been used in [4], [29], [51]. The second data set is King data set which includes the RTTs among 1740 Internet DNS servers, which are collected by the P2PSim project. These DNS servers were obtained from an Internet scale Gnutella network trace. King data set has been used in [5], [10], [42], [43], [50].

In our experiments, we evaluate the four NC implementations on these two data sets. As in [29], all the four NC systems use 8-dimensional coordinates. In Vivaldi, each host has m neighbors, and the default values in [10] are used for all other parameters. Likewise, there are m randomly selected landmarks in IDES. As in [10], [42], we set m as 32. Ten runs are performed on each data set and the average results are reported.

In Fig. 1, we show the CDF of Relative Error of the four NC implementations on these two data sets. As in [29]–[31], more attention is paid to the *90th Percentile Relative Error (NPRE)* since it guarantees 90% of the links have lower RE values than it. Smaller NPRE value indicates higher overall prediction accuracy. From Fig. 1 we can see in both PL data set and King data set, these four implementations perform comparably. Concretely, for PL data set, the NPRE of basic Vivaldi, Vivaldi (with height), dynamic neighbor Vivaldi and IDES is 0.92, 0.92, 0.83 and 0.89, respectively. For King data set, the NPRE of basic Vivaldi, Vivaldi (with height), dynamic neighbor Vivaldi, and IDES is 0.56, 0.54, 0.50 and 0.56, respectively.

According to the simulation in this subsection, we find that the current approaches aiming at improving the prediction accuracy of the traditional Euclidean distance based NC systems, only achieve comparable prediction accuracy with the basic Vivaldi. Among the four implementations, dynamic neighbor Vivaldi performs the best, while it could only reduce the NPRE from basic Vivaldi by up to 10.71%. This motivates us to design a new NC system with significantly higher prediction accuracy. In the following sections, we will introduce the design and implementation of our solution and demonstrate the remarkable improvement over these existing approaches.

III. Phoenix Design and Implementation

A. System Overview

In this section, we present the design and implementation of Phoenix. Phoenix maps each host to a pair of d-dimensional row vectors, namely an incoming vector and an outgoing vector (we refer to the two vectors as coordinates). The predicted distance from host H_i to host H_j is simply the dot product between the outgoing vector of host H_i and the incoming vector of host H_j. For each host, Phoenix ensures all the elements in its two vectors to be non-negative, which guarantees the non-negativity of predicted distances.

As a decentralized NC system, Phoenix system does not require a dedicated set of hosts to serve the whole system. We treat all the hosts equally. Any host with calculated NC can serve as a *reference host* to orient the new host to participate in. Thus, Phoenix is suitable for large scale applications since the communication and computation overhead will be distributed evenly among all hosts in the system. After a new host joins the system, it can pick any host with calculated NC as one of its referenced hosts. Let S be the set of hosts whose NCs have been calculated. When a new host H_{new} joins the system, it randomly selects m reference hosts from set S and starts its NC update procedure. In every round, H_{new} measures its RTTs to these m hosts and retrieves the incoming and outgoing vectors of these m hosts. Then its NC can be calculated and updated periodically. We will discuss the detailed NC calculation policy in Section III-B and Section III-C.

B. NC Calculation of Early Hosts

In Phoenix, NC calculation is different for the first m hosts. We define N as the number of hosts in our system. If $N \leq m$, the new host H_{new} will be considered as one of the *early hosts*. In such case when the scale of the system is very small, a centralized architecture will not hurt scalability. Thus, we let these early hosts probe each other to obtain the $N \times N$ distance matrix. By applying NMF [18], two $N \times d$ matrices, namely X and Y are obtained, while ensuring the non-negativity of the elements in X and Y. For host H_i, its outgoing vector $\vec{X_i}$ is the ith row of matrix X, its incoming vector $\vec{Y_i}$ is the ith row of matrix Y. The measurement and NC calculation procedure will repeat periodically to reflect the latest node participation and RTT values.

Different from the landmarks in GNP [31], ICS [23] and VL [41], which are pre-deployed and will serve as landmarks permanently, early hosts in Phoenix will become ordinary hosts once $N > m$. This means that every host in Phoenix

has the same probability to be selected as a reference host to calculate the NC of another host when $N > m$. Our design of NC calculation allows early hosts to initialize the system without hurting the decentralization of Phoenix.

C. Weight-based NC Calculation for Ordinary Hosts

We introduce a weight-based mechanism for the NC calculation of ordinary hosts in Phoenix. This model plays a major role in the improvement of the overall prediction accuracy over existing NC systems. In NC systems without a weight, the reference hosts with inaccurate NCs will have equal impact in NC calculation as those with accurate NCs. In contrast, if we distinguish the accurate and inaccurate reference NCs, and reduce the negative impact of the inaccurate reference NCs in NC calculation, we can avoid the error propagation and obtain more accurate NCs. In our design, we are aiming at identifying the inaccurate NCs effectively and assigning suitable weight for each reference NC.

Every ordinary host H_{new} randomly selects m hosts in the swarm as its reference hosts namely $R_1, R_2, ..., R_m$. Suppose the outgoing vectors of these m reference hosts are $\vec{X}_1, \vec{X}_2, ..., \vec{X}_m$, while the incoming vectors are $\vec{Y}_1, \vec{Y}_2, ..., \vec{Y}_m$. Suppose D_i^{out} is the distance from the host H_{new} to reference host R_i, and D_i^{in} is the distance from reference host R_i to the host H_{new}. Ideally, we would like the outgoing vector $\vec{X}_{new}$ and the incoming vector $\vec{Y}_{new}$ to satisfy $D_i^{out} = \vec{X}_{new} \cdot \vec{Y}_i$ $(1 \leq i \leq m)$ and $D_i^{in} = \vec{X}_i \cdot \vec{Y}_{new}$ $(1 \leq i \leq m)$ while enforcing all the elements in these two vectors to be non-negative.

Similar to [6], when H_{new} joins the system, we assign bootstrap NCs to it as calculated by Eq.(4) and Eq.(5). Unlike Vivaldi [10], which started all the coordinates from the zero point, we assign a roughly accurate bootstrap NC to each host. This can shorten the convergence procedure. In calculating the bootstrap NC, we treat all the reference NCs equally. At that phase H_{new} has no knowledge of the accuracy of the reference NCs so that the weight-based mechanism cannot be applied temporarily.

$$\vec{X}_{new} = arg \min_{U \in R^{+d}} \sum_{i=1}^{m} (D_i^{out} - \vec{U} \cdot \vec{Y}_i)^2 \tag{4}$$

$$\vec{Y}_{new} = arg \min_{U \in R^{+d}} \sum_{i=1}^{m} (D_i^{in} - \vec{X}_i \cdot \vec{U})^2 \tag{5}$$

Right after bootstrapping, we refine the NC calculation policy to further improve the prediction accuracy. We use a weight-based non-negative least squares module to calculate the NC of each ordinary host. We define w_{X_i} and w_{Y_i} as the weights of vector $\vec{X}_i$ and $\vec{Y}_i$ respectively, where $0 \leq w_{X_i} \leq 1$ and $0 \leq w_{Y_i} \leq 1$. Detailed weight assignment policy will be introduced in the next subsection. We calculate $\vec{X}_{new}$ and $\vec{Y}_{new}$ with the weight-based non-negative least squares error as Eq.(6)-(7). Comparing with the NC calculation in the bootstrapping, we assign different weights to the squared error of different reference NCs.

$$\vec{X}_{new} = arg \min_{U \in R^{+d}} \sum_{i=1}^{m} w_{Y_i} (D_i^{out} - \vec{U} \cdot \vec{Y}_i)^2 \tag{6}$$

$$\vec{Y}_{new} = arg \min_{U \in R^{+d}} \sum_{i=1}^{m} w_{X_i} (D_i^{in} - \vec{X}_i \cdot \vec{U})^2 \tag{7}$$

We rewrite Eq.(6)-(7) in the form of a non-negative least squares constraints problem as Eq.(8)-(9).

$$\vec{X}_{new} = arg \min_{U \in R^{+d}} \sum_{i=1}^{m} (\sqrt{w_{Y_i}} D_i^{out} - \vec{U} \cdot \sqrt{w_{Y_i}} \vec{Y}_i)^2 \tag{8}$$

$$\vec{Y}_{new} = arg \min_{U \in R^{+d}} \sum_{i=1}^{m} (\sqrt{w_{X_i}} D_i^{in} - \sqrt{w_{X_i}} \vec{X}_i \cdot \vec{U})^2 \tag{9}$$

Eq.(4)-(5) and Eq.(8)-(9) are non-negative least squares (NNLS) problems [15] of the form $\min_{x>0} \| Ax - b \|^2$, where $A \in R^{m \times d}, b \in R^{m \times 1}$ are given. To increase the numerical stability, we apply the regularization in NNLS of the form $\min_{x>0} \| Ax - b \|^2 + \lambda \| x \|^2 \quad (\lambda \geq 0)$ where λ is the regularization parameter [3]. Thus, Phoenix gives preference to solutions with smaller Euclidean norms, i.e., smaller $\| x \|$. Similar regularization is used in DMF NC system [22] without enforcing $x > 0$. According to the test using our simulator written in Matlab 7.7, while setting m as 32, by average each host requires only 0.0012 second for the calculation of either $\vec{X}_{new}$ or $\vec{Y}_{new}$ (performed on a computer with Pentium 4 Dual-Core 2.33GHz CPU, 3GB RAM). Comparing with the length of the NC update interval, which is on the order of hundreds of seconds, the NC calculation time is negligible.

D. Link-based Weight Assignment for Reference NCs

For each ordinary host H_{new}, the weight calculation is involved after the bootstrap NC is obtained. While considering reference host R_j $(1 \leq j \leq m)$, we look at the absolute prediction error from R_j to H_{new} denoted as Eq.(10). Likewise, we also obtain the absolute prediction error from H_{new} to R_j as Eq.(11).

$$E_{X_j, Y_{new}} = \mid \vec{X}_j \cdot \vec{Y}_{new} - D(R_j, H_{new}) \mid \tag{10}$$

$$E_{X_{new}, Y_j} = \mid \vec{X}_{new} \cdot \vec{Y}_j - D(H_{new}, R_j) \mid \tag{11}$$

Since the overall objective function F is the sum of the squares of the absolute prediction errors of all links, intuitively, host H_{new} regards the reference NCs with lower absolute prediction errors as more accurate ones, and trust them more. We relate the value of a weight to the absolute prediction error between H_{new} and each of its reference NCs. According to Eq.(6)-(7), we minimize the weighted sum of squared prediction errors. As a result, the accurate reference NCs can play a more significant role in the final calculated $\vec{X}_{new}$ and $\vec{Y}_{new}$. In this subsection, we propose the detailed weight calculation policy of each reference NC based on this intuition. We use two thresholds A and B $(A > B)$ of prediction errors to handle it. For reference hosts with a prediction error lower than or equal to B, we give them full trust where the weight is 1; for those with a prediction error higher than A, we give them zero trust where the weight is 0; and for those with a prediction error in $(B, A]$, we give them conditional trust where the weight is the square of the ratio of B to the prediction error. To balance the trust, we seek

for appropriate thresholds. As median value is steady and widely used in statistics studies, we relate thresholds to it. The median values of the error of $E_{X_j,Y_{new}}(1 \leq j \leq m)$ and $E_{X_{new},Y_j}(1 \leq j \leq m)$ are

$$\begin{aligned} M_X &= median_j(E_{X_j,Y_{new}}) \\ M_Y &= median_j(E_{X_{new},Y_j}) \end{aligned} \tag{12}$$

For the outgoing vectors of reference hosts, we define the thresholds as $B = M_X$ and $A = C \cdot M_X$, where C is a constant. Phoenix sets C to 10 in its implementation. Thus, we obtain the expression of the weight assignment for the outgoing vectors of the reference hosts in Eq.(13). Similarly, we also obtain the expression of the weight assignment for the incoming vectors of the reference hosts in Eq.(14).

$$w_{X_j} = \begin{cases} 1, & \text{if } E_{X_j,Y_{new}} \leq M_X; \\ (M_X/E_{X_j,Y_{new}})^2, & \text{if } M_X < E_{X_j,Y_{new}} < C \cdot M_X; \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

$$w_{Y_j} = \begin{cases} 1, & \text{if } E_{X_{new},Y_j} \leq M_Y; \\ (M_Y/E_{X_{new},Y_j})^2, & \text{if } M_Y < E_{X_{new},Y_j} < C \cdot M_Y; \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Vivaldi also distinguishes different reference hosts by using the weight. Each Vivaldi host i has an *error estimate coefficient* so-called e_i. This coefficient is used by other hosts to decide the confidence of host i and it is maintained and updated by host i itself. Therefore, all the other hosts have the same confidence for a certain host i. Although the convergence time in Vivaldi is shortened by using this *host-based* weight assignment policy, the stabilized prediction accuracy cannot be improved [10]. Differently, as shown in Eq.(10)-(14), Phoenix associates weights with links rather than hosts. Such *link-based* weight assignment is more flexible because different hosts can have different confidence for the same certain host i. The prediction error of each measured link, which is part of the objective function F defined in Eq.(3), can be reflected individually in the weight calculation. Therefore, we can handle the error propagation well. According to our evaluation in IV-B, our link-based weight assignment policy plays an essential role in the improvement of overall prediction accuracy. Moreover, the advantages of our link-based weight assignment not only lie in the accurate characterization of prediction errors, but also lie in the communication overhead. Like in IDES and DMF, one Phoenix host retrieves the NCs of the reference hosts and measures its RTTs to these reference hosts in each round. Calculating the weights of its reference NCs only needs this existing information and thus introduces no extra communication overhead. In contrast, if one Vivaldi host refers to one of its reference hosts to update its own NC, it not only needs to retrieve the $d-dimensional$ NC, but also needs to retrieve the e_i. Such extra transmission takes place in every update round and introduces $1/d$ extra communication overhead. In short, the weight-based mechanism of Phoenix exploits known information to improve the prediction accuracy without introducing extra communication overhead which contradicts with the rationale of NC systems - moving network communication to endhost computation.

E. Workflow of Phoenix

For a new ordinary host, Algorithm 1 shows the procedure of its NC calculation and update. Similar to other P2P schemes, a new Phoenix host first contacts the Rendezvous Point (RP) once for bootstrapping. After obtaining a list of existing hosts from RP as its reference host candidates, this new host contacts them and randomly selects m hosts out of them as its reference hosts. In each round, the new host measures its RTTs to its reference hosts and retrieves their NCs before the calculation of its own NC. Due to the distributed nature of the NC applications, every host can join or leave the system at any time. Therefore, a host may need to find some other online hosts to replace the left hosts in its reference host list. To actively fetch candidates for reference hosts, Phoenix utilizes a distributed scheme, so-called Peer Exchange (PEX) [46], which is used in BitTorrent. Instead of contacting the RP again, hosts gossip with each other for their knowledge of existing hosts. While an ordinary host selects another existing host as one of its reference hosts, these two hosts exchange their reference host lists. Afterwards, they just need to update the information of newly added or removed reference hosts with each other. As mentioned in [13], such information can be simply piggybacked by the NC update. Therefore, every host in the system keeps an active view of its two-hop reference hosts without contacting the RP again and takes it as the list of candidate reference hosts. We will discuss the impact of churn in Phoenix in Section IV-E. Notably, in the first round, the initial outgoing vector $\vec{X}_{new}$ and the incoming vector $\vec{Y}_{new}$ (bootstrap NC) will be calculated using non-negative least squares (Line 7 - Line 9 in Algorithm 1) without introducing weight. After obtaining the bootstrap NC, the outgoing vector $\vec{X}_{new}$ and the incoming vector $\vec{Y}_{new}$ can be calculated and updated using the weighted NC calculation model periodically in each round.

IV. Performance Evaluation

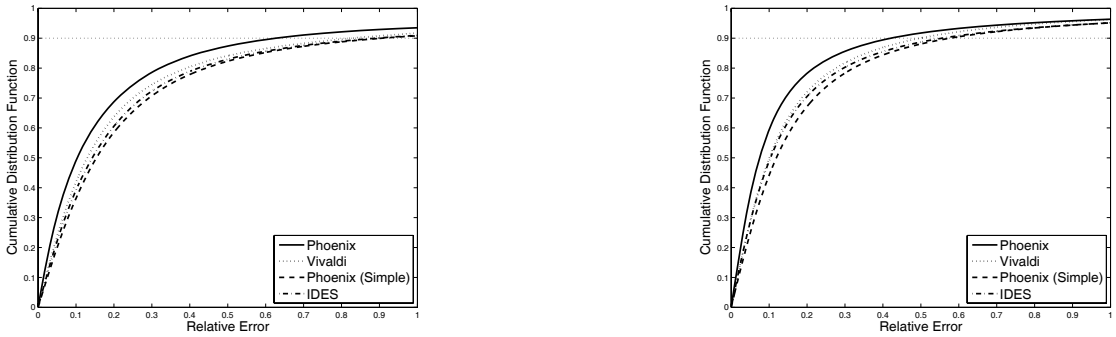
A. Experiment Setup

In our experiment, we compare Phoenix with IDES and Vivaldi. We use the dynamic neighbor Vivaldi to represent a series of Vivaldi implementations, because according to our evaluation in Section II-D it achieves the highest prediction accuracy among the three different evaluated Vivaldi implementations. For IDES and Vivaldi, we use the parameter setting described in Section II-D. Correspondingly, Phoenix uses 8-dimensional coordinates and each host has 32 reference hosts. The regularization parameter λ is set as 2 in Phoenix. We conduct ten runs on each data set and report on the average results.

Our evaluation focuses on three different aspects of NC systems. The first aspect is prediction accuracy, which is the basic function of NC systems. We evaluate this in Section IV-B. The second aspect is the practicality which includes handling the drift of the NCs (Section IV-C), the convergence behavior (Section IV-D), the impact of host churn (Section IV-E), and the robustness against measurement anomalies (Section IV-F). Finally, we evaluate Phoenix in a scalable network monitoring scenario in Section IV-G.

Algorithm 1 Phoenix Algorithm

$Connect\_to\_Rendezvous\_Point(RP)$
$Get\_Reference\_Host\_Candidates(RP)$
$Connect\_to\_Reference\_Hosts()$
$round = 1$
while forever **do**
  $Get(d(\cdot), X(\cdot), Y(\cdot))$
  if $round = 1$ **then**
    $[X_{new}, Y_{new}] = Bootstrap\_NC\_Calculation(d(\cdot), X(\cdot), Y(\cdot))$
  end if
  $[w_X(\cdot), w_Y(\cdot)] = Weight\_Calculation(d(\cdot), X(\cdot), Y(\cdot), X_{new}, Y_{new})$
  $[X_{new}, Y_{new}] = NC\_Calculation(d(\cdot), X(\cdot), Y(\cdot), w_X(\cdot), w_Y(\cdot))$
  Wait(Update_Interval);
  round = round + 1
end while



(a) PL data set

(b) King data set

Fig. 2: CDF of relative error.

B. Evaluation Results on Prediction Accuracy

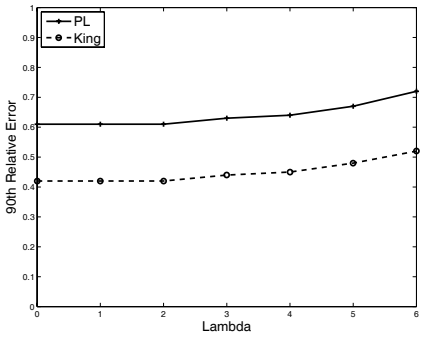
In this subsection, we compare IDES, Vivaldi and Phoenix regarding their prediction accuracy. In order to investigate the contribution of the weight in Phoenix to the prediction accuracy, we introduce a simplified implementation of Phoenix called Phoenix(Simple). The only difference between Phoenix(Simple) and standard Phoenix lies in the NC calculation procedure. Phoenix(Simple) does not use weighted model. It simply sets every weight value as 1 in Eq.(6) and Eq.(7). The comparison between Phoenix and Phoenix(Simple) demonstrates the effectiveness of the weight. In total, we have four different implementations of NC systems in this comparison.

Fig. 2 shows the CDF of relative errors of these four NC implementations using two different data sets. In both of these two data sets, Phoenix achieves much higher prediction accuracy than all other existing NCs. Specifically, for PL data set, the NPRE values of Phoenix, Phoenix(Simple), IDES and Vivaldi are 0.63, 0.91, 0.89 and 0.83, respectively. For King data set, the NPRE values of Phoenix, Phoenix(Simple), IDES and Vivaldi are 0.42, 0.58, 0.56 and 0.50, respectively. Compared with IDES, the representative matrix factorization model based NC, Phoenix reduces the NPRE by between 25.00% (King data set) and 29.21% (PL data set). Therefore, we can see after introducing weight to reduce the impact of the error propagation, the prediction accuracy increases largely. Compared with dynamic neighbor Vivaldi, Phoenix reduces the NPRE by between 16.00% (King data set) and 24.10% (PL data set). Therefore, Phoenix can improve the Vivaldi implementation which uses a TIV aware neighbor selection mechanism even further by employing weight-based mechanism in matrix factorization model. In short, Phoenix achieves significantly lower relative error than existing approaches.

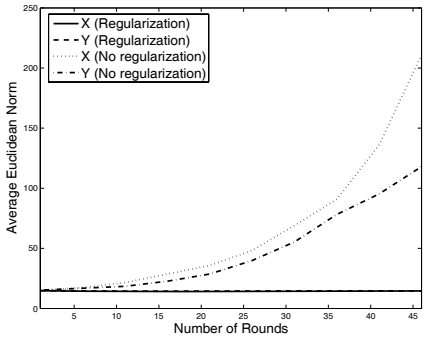
Interestingly, we find that the Phoenix(Simple) performs slightly worse than Vivaldi and IDES. However, compared with Phoenix(Simple), Phoenix reduces the NPRE by between 27.59% (King data set) and 30.77% (PL data set). Therefore, we conclude that the weight-based mechanism is the major factor of the improvement of prediction accuracy.

C. Impact of Regularization

We study the impact of regularization from two aspects. First of all, we have to select a suitable regularization parameter λ to make sure the overall prediction accuracy will not be reduced by introducing the regularization. We study the relationship between the λ value and the overall prediction accuracy using PL data set and King data set by varying the λ value from 0 to 6. Fig. 3(a) shows the average NPRE of Phoenix with different λ values. We find that the average NPRE of Phoenix changes very little when $\lambda <= 2$. When $\lambda > 2$, the NPRE of Phoenix increases as λ increases. We set λ as 2 in order to maintain the accurate prediction, while preventing the Euclidean norms of the NCs from becoming too large.

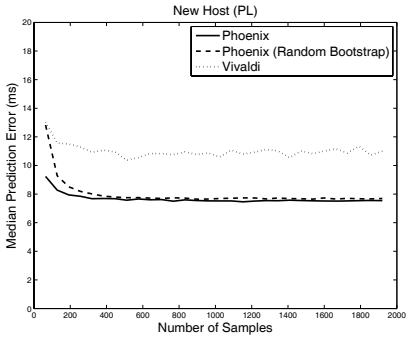


(a) Impact of λ

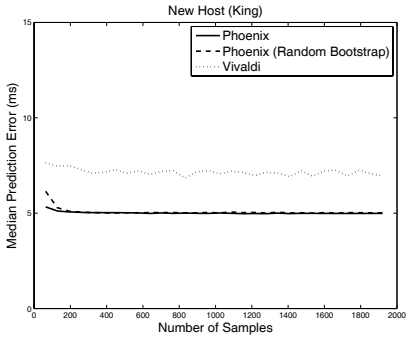


(b) Evolution of average Euclidean norm (PL)

Fig. 3: Impact of regularization.



(a) PL data set



(b) King data set

Fig. 4: Convergence behavior of Phoenix.

The other aspect is the effectiveness for handling the drift of the NCs. In Vivaldi, the centroid of the NCs moves continuously and repeatedly in a certain direction away from the origin [16], [17]. Although the NC calculation only relies on the relative distance information, avoiding the drift is still necessary because the corresponding applications always need to know the potential range of the elements in NCs. As an example in [16], Hilbert's space filling curve can be used to map NCs into one single dimension. Therefore, a priori estimation of the maximum range of the elements in NCs may fill up is required.

Using PL data set, we study the average Euclidean norm of all the outgoing vectors which is defined as $\frac{1}{N}\sum_{i=1}^{N} \parallel \vec{X}_i \parallel$ and average Euclidean norm of all the incoming vectors which is defined as $\frac{1}{N}\sum_{i=1}^{N} \parallel \vec{Y}_i \parallel$. According to Fig. 3(b), we can see after setting the regularization parameter λ as 2, the average Euclidean norm of outgoing vectors and incoming vectors stay stably. In contrast, there is a rapid drift of NCs while setting λ as 0. Similar phenomenon can be observed using King data set. Therefore the usage of regularization can avoid the drift successfully.

D. Analysis of Convergence Behavior

Similar to Vivaldi and any other decentralized NC system, a Phoenix host updates its NC periodically and improves the accuracy of its NC until a steady state is reached, i.e., the accuracy stops increasing. Same as previous literatures [10], [37], we call this procedure as a *convergence procedure*. Besides the final stabilized prediction accuracy, how many measurements are required for the NCs to reach their steady state is also critical. If it takes a new host a lot of measurements and time to reach the steady state, it still cannot be used practically in real Internet applications. A good NC system should converge fast.

In this subsection, we study the convergence property of Phoenix. We apply the dynamic scenario which is used in [37]. For a data set with N hosts, we let the NCs of $N - 40$ randomly selected hosts stabilize first and then introduce the other 40 hosts into the network. Then we can see the evolution of the NC convergence behavior of these 40 new hosts. This reflects how fast our newly injected hosts will converge in a stabilized system. The metric *median prediction error*, defined as $median_{i,j}(\mid D^E(i,j) - D(i,j) \mid)$, is introduced in [10] to evaluate the convergence behavior of the NC systems.

We compare Phoenix with Vivaldi in terms of the number of samples required for convergence. For a Vivaldi host, in each update round it probes one of its neighbors and retrieves the NC of this neighbor. We regard this process as one sample. For a Phoenix host, it has 32 reference hosts and each reference host has an incoming vector and an outgoing vector, thus each update round needs 64 samples. Also, to demonstrate the effectiveness of bootstrap NC calculation, i.e., Line 7 - Line 9 in Algorithm 1, we evaluate a modified Phoenix which uses random bootstrap NCs. We have done the comparison using both PL data set and King data set, and have drawn similar conclusions. According to Fig. 4, we can find that Phoenix

converges very fast, basically requires less than 200 samples. Also, the stabilized median prediction error of Phoenix is much smaller than Vivaldi. The comparison between standard Phoenix and modified Phoenix that uses random bootstrap NCs indicates that the bootstrap NC calculation can make the convergence procedure faster.

E. Analysis of Churn

Handling churn is another important issue for an NC system to become practical, since the hosts can join or leave the system at any time. In Phoenix, a new host needs to conduct measurements to its reference hosts periodically to update its NC. However, part of its reference hosts may leave the system at any time, even before it converges. Therefore every host has to select for some other existing hosts to replace the left reference hosts and some hosts in the system which have not stabilized their NCs yet can be selected [13], [16]. Such selection may lead to lower overall prediction accuracy. In this subsection, we study the performance of Phoenix while considering host churn.

To simulate the churn behavior, we use the model proposed in [12]. We set the session times with PDF $f(x) = ab^a/(x + b)^{a+1}$ with exponent $a = 1.5$. This model is a standard Pareto distribution, shifted b minutes (without the shift, a host would be ensured to have a minimum session time for at least b minutes). The mean session time t_{Avg} of this model can be calculated as $t_{Avg} = ab/(a-1) - b = 2b$. In [12] the authors set b as 15 to tune t_{Avg} as 30 minutes. To evaluate Phoenix using shorter mean session time, we also set b as 5 and 10 to tune t_{Avg} as 10 minutes and 20 minutes. For each of these three scenarios, we vary the length of the NC update interval from 100 seconds to 1000 seconds. Larger update interval (lower sampling rate) means longer convergence time for every single host, which brings more barriers to Phoenix from becoming accurate.

According to Fig. 5, we can see for smaller t_{Avg} (10 minutes), the NPRE increases with the length of the update interval faster than bigger t_{Avg} (20 or 30 minutes) because it generates a higher churn rate. Moreover, we find that using both PL data set and King data set, while the length of the update interval is smaller than or equal to 500 seconds, the NPRE of Phoenix varies little under all three different t_{Avg} settings. In other words, for handling the scenario while the mean session time is as small as 10 minutes, we just need to measure the reference hosts and update NC once in every 500 seconds, which means the sampling rate is $64\ samples \div 500\ seconds = 0.128\ sample/sec$. It is a light measurement traffic comparing with basic Vivaldi, which sets the sampling rate as $1\ sample/sec$ [10].

F. Robustness against Measurement Anomalies

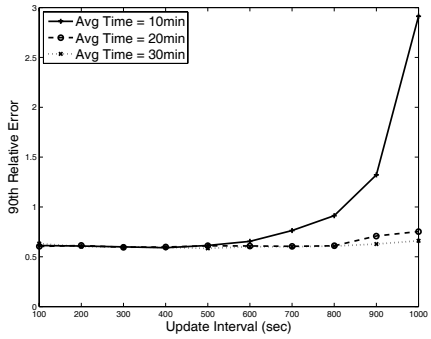
In our study in Section IV-B - Section IV-E, we assume that all the measurement results from the data sets are accurate. However, in the real world, measurement may contain various kinds of errors [29]. A robust NC system should be able to give an accurate prediction under a small amount of measurement anomalies. The robustness against measurement anomalies of an NC system are evaluated by the experiment proposed in [29]. For a certain data set, we randomly pick p percent links of the data set and increase the RTTs of these links to L times of the original values. Then we run every NC system on the modified data set. For the calculation of the RE after the simulation, we compare the predicted distances with the actual network distances (i.e. the original value without injected errors). We vary the p value and see the resulting value of median RE. We set the L value as 3 according to [29].

We do the experiment on PL data set and King data set, and have drawn similar conclusions. We vary the p value from zero to 20. Fig. 6 shows the median relative errors with varying percentage of inaccurately measured links. From these figures, we learn that in IDES, Vivaldi and Phoenix(Simple) system, the median RE increases when the amount of inaccurately measured links increases. In contrast, the results in Phoenix are quite different, while the median RE will also increase along with p, the extent of increase is not that much. If the percentage of the measurement anomalies is less than 10%, the median RE value of Phoenix just increased by less than 15%. Thus Phoenix is very robust to small amount of measurement anomalies. The difference between Phoenix and Phoenix(Simple) demonstrates that the weight is able to eliminate the impact of measurement anomalies greatly. Moreover, the difference between Phoenix and Vivaldi indicates link-based weight assignment is more suitable for handling measurement anomalies than host-based weight assignment, since it can detect the abnormal measurement results easily and reduce the impact of this particular measurement. In contrast, the host-based weight assignment in Vivaldi is not able to capture the measurement anomalies of a certain link.

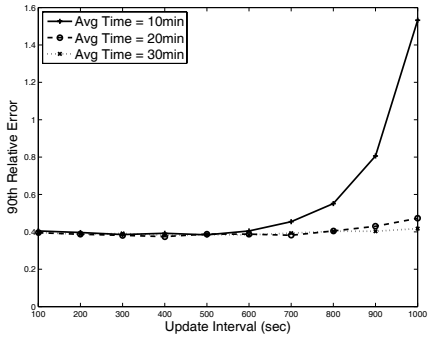
G. Application: Scalable End-to-end Network Monitoring

In this subsection, we will evaluate the suitability of using Phoenix as distance prediction module in scalable end-to-end network monitoring scenario. Most network monitoring systems are designed for capturing the dynamic states of the end-to-end network paths [48]. By performing measurement periodically, the near real-time parameter values of the end-to-end links being monitored can be obtained. For example, S^3 (Scalable Sensing Service) [47] is a service which measures all-pairs network parameters like distance of hundreds of PlanetLab hosts periodically. It collects and publishes a snapshot of latest measured results every four hours. These information can be used for resource placement and location [47], server selection in cloud services [44], overlay construction [51], and path selection [47]. However, such end-to-end measurement based approach cannot scale well with the increase of the host number in the system. Deploying an NC system to obtain the predicted all-pairs distances is a scalable solution. In [36], some NC systems include GNP and Vivaldi are implemented and evaluated through the infrastructure of S^3 service.

To evaluate the accuracy of an NC system in the end-to-end network monitoring scenario, we need the dynamic data set as the input. In the previous NC related literatures except [4], only aggregate data sets which combine measurements taken at different times are used. In these data sets, the final RTT between two hosts is obtained by taking the median [10], [45] or minimum value of measured RTTs [49], [53] over

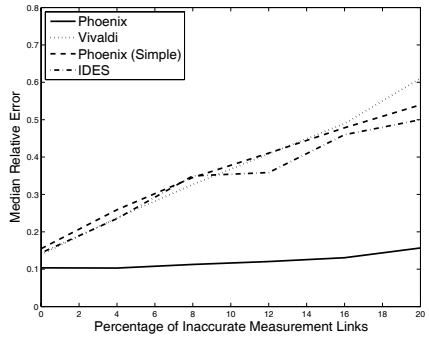


(a) PL data set

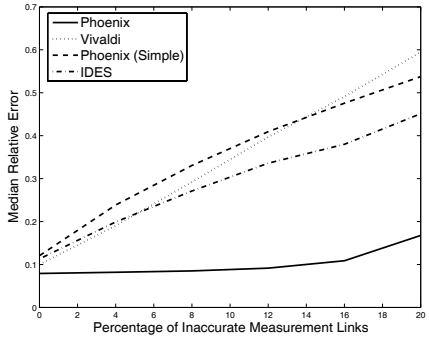


(b) King data set

Fig. 5: Churn study of Phoenix.



(a) PL data set



(b) King data set

Fig. 6: The impact on measurement anomalies.

long periods of time (days or even weeks). Obviously, if we aggregate all the different RTT values of a certain link into one value, the variation cannot be reflected at all. Different from aggregate data sets, dynamic data sets include a sequence of all-pair distances snapshots, taken at consecutive time intervals. In this subsection, we use the "king200-allpairs-1h" data set in [27] to evaluate the efficiency of Phoenix in network monitoring. This data set includes 200 hosts and all pairs of these hosts are measured at least once within every hour using the King method. The duration of the data collection is 44 hours. Compared with performing all-pairs measurement directly, which requires each host to measure the distances between itself and every other host (199 other hosts in total), using NC can reduce this number to 32. Therefore, up to $(199 - 32) \div 199 \times 100\% = 83.92\%$ measurement overhead can be saved.

We compare Phoenix with Vivaldi and IDES in our evaluation. Fig. 7 shows the NPRE of all the three NC systems during the 44 hours data collection period. We can find that Phoenix achieves much lower relative error than Vivaldi and IDES consistently during the whole data collection period. The average NPRE value of Phoenix, Vivaldi and IDES is 0.56, 0.87, 1.36, respectively. Therefore Phoenix can achieve scalable yet accurate network monitoring. Precisely, Phoenix reduces the average NPRE of Vivaldi by 35.63%. Meanwhile, Phoenix reduces the average NPRE of IDES by 58.82%. Therefore, we conclude that Phoenix achieves significantly better prediction accuracy than Vivaldi and IDES.

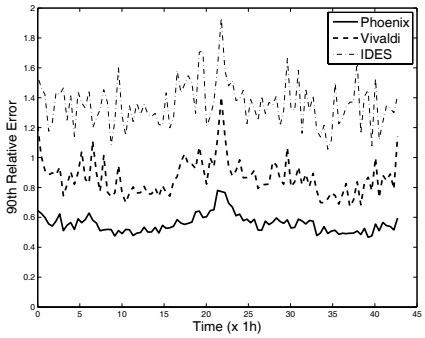
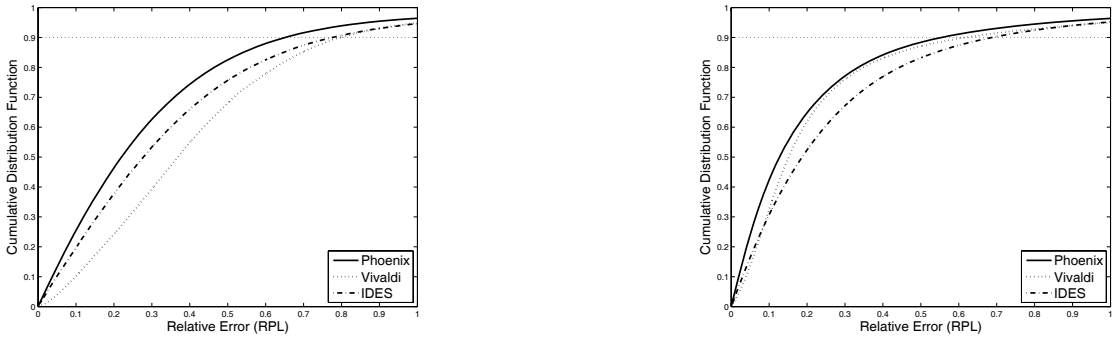


Fig. 7: 90th percentile relative error for the "K200-allpairs-1h" data set.

V. Analysis of TIV Characterization

To date, some metrics have been proposed to calculate the severity of TIV for each link, such as Relative Path Loss (RPL) [16], [41]. By using these metrics, we can study the severity of TIV of every link based on either measured distances or predicted distances. However, there is still no existing study about the degree of approximation between measured distances and predicted distances in terms of the statistic properties of the severity of TIV. This reflects how well an NC system can characterize the TIV property of the Internet. Without a general quantitative metric, we cannot distinguish different NC systems from the perspective of characterizing the TIV property on the Internet. In this section, two quantitative metrics are proposed. Using these two metrics, we can see whether Phoenix can achieve better



(a) PL data set

(b) King data set

Fig. 8: CDF of relative error (RPL).

TIV characterization ability compared with other existing approaches.

There are two steps in designing our metrics. First, we choose a typical metric which is used for evaluating the severity of TIV of each end-to-end link. We apply this metric based on both measured distances and predicted distances. Therefore each link will have two values, which can reflect the severity of TIV of the measured distances and the predicted distances. As the second step, we calculate the difference of the resulted two values. For an NC system which can characterize the TIV property better, this difference will be smaller.

RPL [16], [41] is a typical metric to evaluate the severity of TIV of a certain host. RPL of the link between host H_i and host H_j is defined as $RPL(i,j) = min_{1\leq k\leq N}(\frac{D(i,k)+D(k,j)}{D(i,j)})$.

With the value of RPL, the severity of TIV of a certain link can be characterized. If the RPL value of a certain link is smaller than 1, there exists at least one other host H_k, the distance from H_i via H_k to H_j is shorter than the direct path between node H_i and node H_j.

For the evaluation of the distance prediction accuracy, we use relative error (RE) of the distance between each host pair to measure how well a predicted distance matches the corresponding measured distance. Similarly, to evaluate the difference between the RPL based on measured distances and corresponding RPL^E based on the predicted distances[2], we propose to use the *relative error of the RPL (RE_{RPL})* for each link. We define the new metric RE_{RPL} as follows:

$$RE_{RPL}(i,j) = \frac{\mid RPL^E(i,j) - RPL(i,j) \mid}{min(RPL^E(i,j), RPL(i,j))} \qquad (15)$$

Small RE_{RPL} value indicates better characterization of the severity of TIV of the Internet. For any two hosts H_i and H_j, if the RPL value based on the measured distances equals to the RPL value based on the predicted distances, the RE_{RPL} value will be zero.

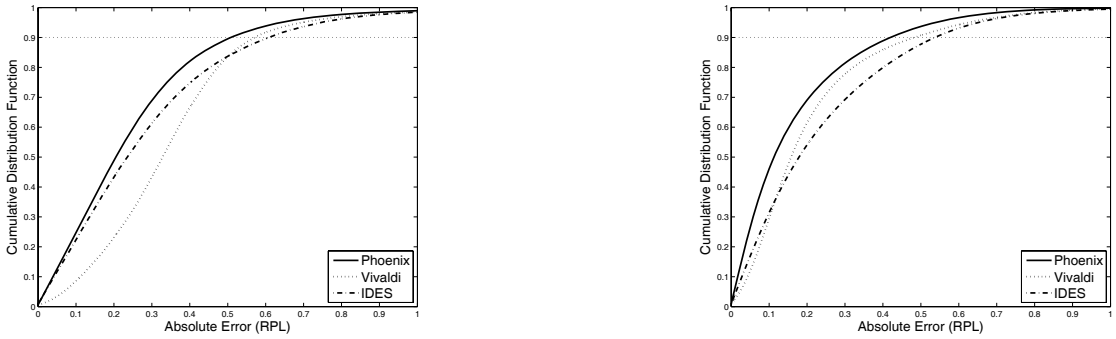
Besides relative error, absolute error is also used for the distance prediction accuracy evaluation [2]. Similarly, we propose to evaluate the *absolute error of the RPL (AE_{RPL})* of each link. This new metric AE_{RPL} is defined as $AE_{RPL}(i,j) = \mid RPL^E(i,j) - RPL(i,j) \mid$.

Fig. 8 shows the CDF of RE_{RPL} of all N^2 end-to-end links by using three NC implementations, i.e., Phoenix, IDES and dynamic neighbor Vivaldi, using two different data sets. For the PL data set, the 90th percentile RE_{RPL} of IDES, Vivaldi and Phoenix are 0.77, 0.80, and 0.65, respectively. For the King data set, the 90th percentile RE_{RPL} of IDES, Vivaldi and Phoenix are 0.69, 0.62, and 0.55, respectively. Meanwhile, in Fig. 9, the CDF of AE_{RPL} of these three NC implementations are shown. For the PL data set, the 90th percentile AE_{RPL} of IDES, Vivaldi and Phoenix are 0.61, 0.57, and 0.51, respectively. For the King data set, the 90th Percentile AE_{RPL} of IDES, Vivaldi and Phoenix are 0.54, 0.49, and 0.42, respectively. In short, using either RE_{RPL} or AE_{RPL} as a metric, the 90th percentile value of Phoenix is much smaller than IDES and Vivaldi using both of these data sets. Thus Phoenix can characterize the TIV property much better than other existing NC systems. Comparing with IDES, we can see after introducing weight, the performance improvement is not only in terms of prediction accuracy metrics like relative error (which we have studied in Section IV) but also in terms of TIV characterization. That indicates although there are no constraints of TIV in IDES, the error propagation which is caused by treating different reference hosts equally in NC calculation, plays a bad role in characterizing TIV property. In other words, while using the same basic model, Phoenix explores the advantage of the matrix factorization model better than IDES. For the comparison between IDES and Vivaldi, the results vary among different data sets. Therefore TIV aware neighbor selection [42] can help Vivaldi to characterize the TIV property, performing even better than IDES in some data sets. In a word, both of the basic model and the detailed NC calculation algorithm are important for characterizing the wide existence of TIV on the Internet.

VI. Conclusion and Future Work

In this paper, we investigate matrix factorization based NC systems in depth for accurate and scalable end-to-end Internet distance prediction. We design and implement a fully decentralized matrix factorization based NC system, so-called Phoenix. Phoenix employs a weight-based NC calculation policy to reduce the impact of the inaccurate reference NCs.

[2]Due to the triangle inequality, for Euclidean distance based NC systems such as Vivaldi, the RPL^E of any link is always larger than or equal to 1.



(a) PL data set

(b) King data set

Fig. 9: CDF of absolute error (RPL).

Extensive simulation results using real Internet traces show that Phoenix achieves much higher prediction accuracy than state-of-the-art NC systems. Moreover, we demonstrate that Phoenix has short convergence time, avoids the drift of the NCs successfully and performs well in churn scenario while keeping a low sampling rate. Phoenix is robust against small amount of measurement anomalies. As an important application of NC systems, we evaluate Phoenix in a scalable network monitoring scenario and demonstrate the usefulness of Phoenix. In short, Phoenix is an accurate, decentralized and robust solution to scalable Internet distance predictions.

We also provide detailed discussion on characterizing the TIV property in NC systems which has never been studied numerically before. Two quantitative metric called RE_{RPL} and AE_{RPL} are designed and applied for the evaluation. According to our experimental results, Phoenix can characterize the severity of TIV much better than other existing NC systems. This shows that Phoenix improves the prediction accuracy by better characterizing TIV.

As the next step, we are interested in designing security mechanisms for Phoenix system to prevent from the potential malicious attacks. We will further improve Phoenix for use in large-scale Internet applications, enabling it become a robust, accurate and widely used NC system.

REFERENCES

[1] I. Abraham and D. Malkhi, "Compact routing on Euclidian metrics," in *Proc. 2004 ACM PODC*.

[2] S. Agarwal and J. Lorch, "Matchmaking for online games and other latency-sensitive P2P systems," in *Proc. 2009 ACM SIGCOMM*.

[3] B. Morini, M. Porcelli, and R. H. Chan, "A reduced Newton method for constrained linear least-squares problems," *J. Computational Applied Mathematics*, vol. 233, no. 9, pp. 2200–2212, 2010.

[4] Y. Chen, P. Sun, X. Fu, *et al*., "Improving prediction accuracy of matrix factorization based network coordinate systems," in *Proc. 2010 IEEE ICCCN*.

[5] Y. Chen, Y. Xiong, X. Shi, *et al*., "Pharos: accurate and decentralised network coordinate system," *IET Commun.*, vol. 3, no. 4, pp. 539–548, 2009.

[6] Y. Chen, G. Zhao, *et al*., "Handling node churn in decentralized network coordinate system," *IET Commun.*, vol. 3, no. 10, pp. 1578–1586, 2009.

[7] Z. Chen, Y. Chen, C. Ding, *et al*., "Pomelo: accurate and decentralized shortest-path distance prediction in social graphs," in *Proc. 2011 ACM SIGCOMM (Poster)*.

[8] M. Costa, M. Castro, A. Rowstron, *et al*., "PIC: practical Internet coordinates for distance estimation," in *Proc. 2004 IEEE ICDCS*.

[9] J. Cowling, D. Ports, *et al*., "Census: location-aware membership management for large-scale distributed systems," in *Proc. 2009 USENIX ATC*.

[10] F. Dabek, R. Cox, and F. Kaashoek, "Vivaldi: a decentralized network coordinate system," in *Proc. 2004 ACM SIGCOMM*.

[11] B. Donnet, B. Gueye, and M. Kaafar, "A survey on network coordinates systems, design, and security," *IEEE Commun. Surveys Tuts.*, Dec. 2010.

[12] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," in *Proc. 2006 ACM SIGCOMM*.

[13] B. Gueye and G. Leduc, "Resolving the noxious effect of churn on Internet coordinate systems," in *Proc. 2009 IWSOS*.

[14] M. A. Kaafar, B. Gueye, *et al*., "Towards a two-tier Internet coordinate system to mitigate the impact of triangle inequality violations," in *Proc. 2008 IFIP-TC6 Netw.*

[15] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*, chapter 23, p. 161. Prentice-Hall, 1974.

[16] J. Ledlie, P. Gardner, and M. Seltzer, "Network coordinates in the wild," in *Proc. 2007 NSDI*.

[17] J. Ledlie, P. Pietzuch, and M. Seltzer, "Stable and accurate network coordinates," in *Proc. 2006 IEEE ICDCS*.

[18] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[19] S. Lee, Z. Zhang, *et al*., "On suitability of Euclidean embedding of Internet hosts," in *Proc. 2006 ACM SIGMetrics/Performance*.

[20] S. Lee, Z. Zhang, *et al*., "Fundamental effects of clustering on the Euclidean embedding of Internet hosts," in *Proc. 2007 IFIP Netw.*

[21] Y. Liao, M. Kaafar, *et al*., "Detecting triangle inequality violations in Internet coordinate systems by supervised learning," in *Proc. 2009 IFIP-TC6 Netw.*

[22] Y. Liao, P. Geurts, *et al*., "Network distance prediction based on decentralized matrix factorization," in *Proc. 2010 IFIP-TC6 Netw.*

[23] H. Lim, J. C. Hou, *et al*., "Constructing Internet coordinate system based on delay measurement," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 513–525, 2005.

[24] E. K. Lua, T. Griffin, M. Pias, *et al*., "On the accuracy of embeddings for Internet coordinate systems," in *Proc. 2005 ACM IMC*.

[25] E. K. Lua, X. Zhou, *et al*., "Scalable multicasting with network-aware geometric overlay," *Comput. Commun.*, vol. 31, no. 3, pp. 464–488, 2008.

[26] C. Lumezanu, R. Baden, D. Levin, N. Spring, *et al*., "Symbiotic relationships in Internet routing overlays," in *Proc. 2009 NSDI*.

[27] C. Lumezanu, R. Baden, N. Spring, *et al*., "Triangle inequality variations in the Internet," in *Proc. 2009 ACM IMC*.

[28] C. Lumezanu and N. Spring, "Measurement manipulation and space selection in network coordinates," in *Proc. 2008 IEEE ICDCS*.

[29] Y. Mao, L. Saul, and J. M. Smith, "IDES: an Internet distance estimation service for large network," *IEEE J. Sel. Areas Commun.*, 2006.

[30] T. S. E. Ng and H. Zhang, "A network positioning system for the Internet," in *Proc. 2004 USENIX ATC*.

[31] T. S. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proc. 2002 IEEE INFOCOM*.

[32] M. Pias, J. Crowcroft, S. Wilbur, *et al*., "Lighthouses for scalable distributed location," in *Proc. 2003 IPTPS*.

[33] P. Pietzuch and J. Ledlie, "Network-aware operator placement for stream-processing systems," in *Proc. 2006 IEEE ICDE*.

[34] W. H. Press *et al*., *Numerical Recipes*, 3rd edition, chapter 14, p. 745. Cambridge University Press, 2007.

[35] S. Rhea, D. Geels, T. Roscoe, *et al*., "Handling churn in a DHT," in *Proc. 2004 USENIX ATC*.

[36] P. Sharma, Z. Xu, S. Banerjee, and S. Lee, "Estimating network proximity and latency," *ACM SIGCOMM CCR*, vol. 36, no. 3, pp. 39–50, 2006.
[37] M. Sherr, M. Blaze, and B. T. Loo, "Veracity: practical secure network coordinates via vote-based agreements," in *Proc. 2009 USENIX ATC*.
[38] M. Steiner and E. W. Biersack, "Where is my peer? Evaluation of the Vivaldi network coordinate system in azureus," in *Proc. 2009 International IFIP-TC6 Netw.*
[39] M. Szymaniak, D. Presotto, G. Pierre, and M. Steen, "Practical large-scale latency estimation," *Comput. Netw.*, vol. 52, no. 7, pp. 1343–1364, 2008.
[40] L. Tang and M. Crovella, "Geometric exploration of the landmark selection problem," in *Proc. 2004 PAM*.
[41] L. Tang and M. Crovella, "Virtual landmarks for the Internet," in *Proc. 2003 ACM IMC*.
[42] G. Wang, B. Zhang, and T. S. E. Ng, "Towards network triangle inequality violation aware distributed systems," in *Proc. 2007 ACM IMC*.
[43] G. Wang and T. S. E. Ng, "Distributed algorithms for stable and secure network coordinates," in *Proc. 2008 ACM IMC*.
[44] P. Wendell, J. W. Jiang, *et al*., "DONAR: decentralized server selection for cloud services," in *Proc. 2010 ACM SIGCOMM*.
[45] B. Wong, A. Slivkins, *et al*., "Meridian: a lightweight network location service without virtual coordinates," in *Proc. 2005 ACM SIGCOMM*.
[46] D. Wu, P. Dhungel, X. Hei, *et al*., "Understanding peer exchange in Bittorrent systems," in *Proc. 2010 IEEE P2P*.
[47] P. Yalagandula, P. Sharma, *et al*., "S3: a scalable sensing service for monitoring large networked systems," in *Proc. ACM SIGCOMM INM*, 2006.
[48] P. Yalagandula, S. J. Lee, *et al*., "Correlations in end-to-end network metrics: impact on large scale network monitoring," in *Proc. 2008 IEEE Global Internet*.
[49] B. Zhang, T. S. E. Ng, *et al*., "Measurement-based analysis, modeling, and synthesis of the Internet delay space," in *Proc. 2006 ACM IMC*.
[50] R. Zhang, Y. C. Hu, *et al*., "A hierarchical approach to Internet distance prediction," in *Proc. 2006 IEEE ICDCS*.
[51] R. Zhang, C. Tang, Y. C. Hu, S. Fahmy, and X. Lin, "Impact of the inaccuracy of distance prediction algorithms on Internet applications: an analytical and comparative study," in *Proc. 2006 IEEE INFOCOM*.
[52] X. Zhao, A. Sala, C. Wilson, *et al*., "Orion: shortest path estimation for large social graphs," in *Proc. 2010 WOSN*.
[53] H. Zheng, E. K. Lua, M. Pias, *et al*., "Internet routing policies and round-trip-times," in *Proc. 2005 PAM*.
[54] Y. Zhu, Y. Chen, Z. Zhang, *et al*., "Taming the triangle inequality violations with network coordinate system on real Internet," in *Proc. 2010 ACM, Workshop Re-Architecturing Internet*.

Yang Chen is a postdoc research fellow at University of Goettingen, Germany. He received his B.S. and Ph.D. degrees from Department of Electronic Engineering, Tsinghua University in 2004 and 2009, respectively. He visited Stanford University as visiting student in 2007. He served as a guest editor of Elsevier Computer Networks Special Issue on Measurement-based Optimization of P2P Networking and Applications, a guest editor of IET Communications Special Issue on Peer-to-Peer Systems and Online Social Networking, TPC co-chair of P2PNet'09, P2PNet'10, and HotPOST'11, Session chair/TPC member for several conferences such as IEEE ICDCS, IEEE ICCCN, and IEEE GLOBECOM.

Xiao Wang is a Ph.D. student at Department of Computer Science and Engineering, University of Washington. She received her B.Eng. in Department of Electronic Engineering from Tsinghua University. Her research interest lies broadly in systems and networking, specifically in network measurement, privacy and mobile computing.

Cong Shi is a PhD student at College of Computing, Georgia Institute of Technology. He received his B.S. and M.S. degrees (with honors) in computer science from Shanghai Jiaotong University. His research interests cover P2P networks, delay tolerant networks and online social networks.

Eng Keong Lua is a faculty member of College of Engineering, Carnegie Mellon University, Pittsburgh, USA and Systems Scientist, Carnegie Mellon CyLab USA and Japan. He obtained his Ph.D. degree in Computer Science from the University of Cambridge. His research areas include Peer-to-Peer networks, and network security with social networking.

Xiaoming Fu received his Ph.D. in Computer Science from Tsinghua University, China in 2000. He is a professor at the University of Goettingen. He has also held visiting positions at ETSI, University of Cambridge, Columbia University, Tsinghua University, and UCLA.

Beixing Deng received his Ph.D. in Faculty of Computational Mathematics and Cybernetics from Lomonosov Moscow State University. He is an associate professor with Electronic Engineering Department, Tsinghua University.

Xing Li received his Ph.D. in Electrical Engineering from Drexel University. He is a Professor with the Electronic Engineering Department, Tsinghua University and the deputy director of China Education and Research Network (CERNET) Center.