Phoenix: Towards an Accurate, Practical and Decentralized Network Coordinate System

Yang Chen[1], Xiao Wang[1], Xiaoxiao Song[1], Eng Keong Lua[2], Cong Shi[3], Xiaohan Zhao[1], Beixing Deng[1], and Xing Li[1]

[1] Tsinghua National Laboratory for Information Science and Technology
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[2] College of Engineering, Carnegie Mellon University, Pittsburgh, PA 15213
[3] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332
chenyang04@mails.tsinghua.edu.cn

Abstract. Network coordinate (NC) system allows efficient Internet distance prediction with scalable measurements. Most of the NC systems are based on embedding hosts into a low dimensional Euclidean space. Unfortunately, the accuracy of predicted distances is largely hurt by the persistent occurrence of Triangle Inequality Violation (TIV) in measured Internet distances. IDES is a dot product based NC system which can tolerate the constraints of TIVs. However, it cannot guarantee the predicted distance non-negative and its prediction accuracy is close to the Euclidean distance based NC systems. In this paper, we propose Phoenix, an accurate, practical and decentralized NC system. It adopts a weighted model adjustment to achieve better prediction accuracy while it ensures the predicted distances to be positive and usable. Our extensive Internet trace based simulation shows that Phoenix can achieve higher prediction accuracy than other representative NC systems. Furthermore, Phoenix has fast convergence and robustness over measurement anomalies.

Keywords: P2P, Network Coordinate System, Triangle Inequality Violation, Dot Product, Weighted Model.

1 Introduction

Network Coordinate (NC) system is an efficient and scalable mechanism to predict distance (Round Trip Time) between any two Internet hosts without explicit measurements. In most of the NC systems, each host is assigned a set of numbers called coordinates to represent its position in the Euclidean space, and the distance between any two hosts can be predicted by their coordinates using Euclidean distance. NC system reduces the active probing overhead significantly, which is especially beneficial to large-scale distributed applications. To date, NC systems are widely used in different Internet applications such as application layer multicast [1], locality-aware server selection [1], distributed query optimization [2], file-sharing via BitTorrent [3], network modeling [4], compact routing [5], and application layer anycast [6].

L. Fratta et al. (Eds.): NETWORKING 2009, LNCS 5550, pp. 313–325, 2009.
© IFIP International Federation for Information Processing 2009

Unfortunately, the Euclidean distance based NC systems have a common unremediable drawback, i.e., the predicted distances among each three hosts must satisfy the triangle inequality. Lots of existing studies report the existence of Triangle Inequality Violations (TIV) in the Internet delay structure [8,7,10,22,25,32]. As a result, these distances cannot be predicted accurately by using Euclidean distance based NC even if we increase the dimension of the space.

A dot product based NC system named IDES is proposed in [9] . The key idea of this system is that a large distance matrix can be approximately factorized into two smaller matrices by methods like Singular Value Decomposition (SVD) or Non-negative Matrix Factorization (NMF) [11]. This results in a compressed version of the Internet distance matrix. In contrast to the Euclidean distance based NC systems, the distances predicted by IDES do not have to satisfy the triangle inequality. However, IDES is still not an ultimate solution. First, unlike any other existing NC system, IDES will give negative predicted distance. This will cause the malfunction of the system because the distance (Round Trip Time) can not be negative. In addition, the prediction accuracy of IDES is close to typical Euclidean distance based NC system such as GNP [18] according to the experiments in [1].

In this paper, we propose an accurate, practical and decentralized NC system named Phoenix. Phoenix is also based on dot product, but remedies IDES's flaws. Phoenix can achieve much higher prediction accuracy than other typical representative NC systems such as IDES and Vivaldi [19]. The key contributions of this paper are twofold. (1) We propose a weight calculation algorithm to distinguish referred NCs with high errors and low errors. With the error propagation eliminated, Phoenix demonstrates the advantage of dot product based NC systems. Our extensive Internet trace based simulation results show that Phoenix can achieve much higher prediction accuracy than state-of-the-art methods. Our simulation results also demonstrate Phoenix's fast convergence and robustness over measurement anomalies. (2) Compared with IDES, Phoenix not only performs better in prediction accuracy but also guarantees the predicted distance non-negative. The results show that the implementation of Phoenix is an accurate solution to build a practical NC system.

The rest of this paper is organized as follows. In Section 2, we review the related work. The design of our accurate, practical and decentralized NC system - Phoenix is proposed in section 3. We evaluate the performance of Phoenix and compare it with two representative NC systems with extensive simulation results in Section 4. We conclude the whole paper in Section 5.

2 Related Work

2.1 Euclidean Distance Based Network Coordinates and Triangle Inequality Violation (TIV)

Suppose there are N Internet hosts. Let S be the set of these N hosts. Let D be the $N \times N$ distance matrix among the hosts in S. Thus $D(i, j)$ represents the measured Round Trip Time (RTT) between host i and host j.

Basically, NC is an embedding of these N hosts into m-dimensional Euclidean space R^m. Defining x_i as the NC of host i, we have $x_i = (r_1^i, r_2^i, ..., r_m^i), r_k^i \in R, 1 \leqslant k \leqslant m$. Then x_i and x_j can be used to predict the RTT between host i and host j. We use $D^E(i,j)$ to represent this predicted RTT. The definition of $D^E(i,j)$ is as follows.

$$D^E(i,j) = \|x_i - x_j\| = \sqrt{\sum_{1 \leq k \leq m} (r_k^i - r_k^j)^2} \tag{1}$$

Several NC systems have been proposed in the literature, and they can be categorized into two classes [8], namely centralized NC systems and decentralized NC systems. Centralized NC systems such as GNP [18], Virtual Landmarks [14] require a fixed set of dedicated landmarks to orient the NC calculation of the whole system, which will be a bottleneck of the system. Therefore, decentralized NC systems such as Vivaldi [19], NPS [26], PIC [31] were proposed to make NC system work well on large-scale applications. In this paper, we compare our system with Vivaldi since it's the representive Euclidean distance based NC system due to its clean and decentralized implementation. It is deployed in many well-known Internet systems, such as Bamboo DHT [28], Stream-Based Overlay Network (SBON) [2] and Azureus BitTorrent [3].

The prediction accuracy of an NC system is often denoted by the relative error (RE) of predicted distance over the real RTT measured on Internet. Relative Error (RE) of the distance between host i and host j is defined as [10, 12, 13, 14, 15, 16, 17]

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

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

Suppose there are three hosts, A, B and C. Let's consider the triangle ABC. Suppose AB is the longest edge of the triangle. If $D(A,B) > D(A,C) + D(C,B)$, then ABC is called a TIV, due to the violation of the triangle inequality. As mentioned in [25], any three hosts with TIV cannot be embedded into Euclidean space within some level of accuracy, for the distances among them in Euclidean space must obey triangle inequality. However, the TIV is natural and persistent in Internet [25]. Therefore the existence of TIV causes a serious problem for every Euclidean distance based NC systems [7, 8]. In other words, it hurts the accuracy of Euclidean distance based NC a lot.

2.2 Dot Product Based NC and IDES

Suppose there are N Internet hosts. Let D be the $N \times N$ distance matrix among these N hosts. $D(i,j)$ represents the measured RTT between host i and host j. This $N \times N$ matrix can be factorized into two smaller matrices. $D \approx XY^T$ where X and Y are $N \times d$ matrices ($d << N$).

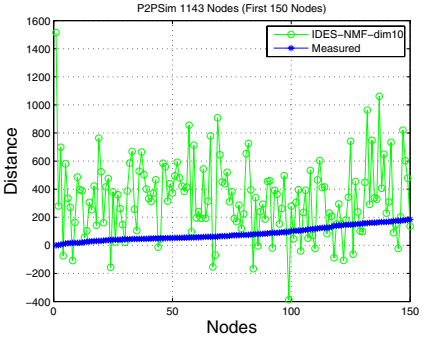


Fig. 1. Maximum RRL using IDES-NMF

In dot product based NC system, for host i, $\boldsymbol{X_i}$ is the *outgoing vector* and $\boldsymbol{Y_i}$ is the *incoming vector*. The predicted distance from host i to host j is simply the dot product between the outgoing vector of host i and the incoming vector of host j as follows

$$D^E(i,j) = \boldsymbol{X_i} \cdot \boldsymbol{Y_j} = \sum_{k=1}^{d} X(i,k) \cdot Y(j,k) \tag{3}$$

IDES [9] is the representative NC system based on dot product. There are several serious problems with IDES. First, IDES's predicted distance can be negative, which is very harmful to many practical applications. In a simple example, we apply maximum RRL test [10] using the P2PSim latency data set [9] for the IDES(NMF) methods in 10-dimensional space (with 15 landmarks). We use IDES simulator which is available at the author's homepage [29]. In this test, we select the site with maximum RRL value as the target site. Fig.1 demonstrates the results. The x-axis enumerates sites while the y-axis corresponds to the RTT distance of each site from the selected site. The signature plot marked with +'s indicates RTTs in the original distance matrix, and the sites on the x-axis have been sorted to ensure that this plot is in ascending order of RTTs. The signature plot marked with o's is the predicted distances given by IDES. It is obvious that IDES results in negativity in distances. This is not realistic or relevant since the distance stands for RTT on Internet. Despite the small percentage of negative distances, their impact on the whole application will be very severe. For example, a typical NC application in overlay multicast is the construction of Minimum Spanning Tree (MST) using NC predicted distance [1]. If we use Dijkstra algorithm to construct the MST, even one negative distance can fail the algorithm.

Furthermore, estimation error will be propagated in IDES's distance prediction. IDES uses the least square error for NC calculation. For a certain host, it uses the NCs of reference hosts and its distances to reference hosts to calculate its own NC. In the NC calculation, this host gives equal confidence to each referred

NC in both basic IDES and decentralized IDES. However, some NCs are very inaccurate due to different factors, such as network congestion or error propagation. Once these inaccurate NCs are referred, their errors will be propagated to the new hosts. Thus, it is not surprising that IDES doesn't show significant improvement on the prediction accuracy over Euclidean distance based NC [1].

In IDES, we can add non-negative constraint in the NC calculation of ordinary hosts to avoid negative predicted distance. However, the prediction accuracy still cannot be improved [9]. In next section, we will propose our solution, an accurate, practical and decentralized NC system called Phoenix. Phoenix will not only guarantee the predicted distance non-negative but also improve the prediction accuracy a lot.

3 Design of Phoenix

3.1 System Architecture

In this section, we propose the design of Phoenix, a practical dot product based NC system. Our design focuses on the most important aspects for NC systems, i.e., accuracy, decentralization and practicability. Basically, NC is used for Internet distance prediction; so the prediction accuracy is principally important. Moreover, NC is widely used in distributed applications, there may be thousands of hosts in the swarm; therefore, NC system should be decentralized. Last but not the least, NC system should be practical. In other words, it should never give negative predicted distances.

Phoenix maps each host to two d-dimensional row vectors - an incoming vector and an outgoing vector. The predicted distance from host i to host j is simply the dot product between the outgoing vector of host i and the incoming vector of host j. In contrast to IDES, all the elements in these two vectors are non-negative, which guarantees Phoenix never gives negative predicted distance.

Unlike GNP [18] or other centralized NC systems, Phoenix system has no fixed network infrastructure or distinguished hosts to serve the whole system. Any host with calculated NC can serve as a *reference host* to orient the new host to participate in. Thus Phoenix is efficient for large scale applications since the communication and computation overhead will be distributed evenly to all hosts in the system. After a new host joins the system, it can pick any host with calculated NC as one of its reference hosts. Let N be the set of hosts whose NCs have been calculated. When a new host H_{new} joins the system, it can select any m reference hosts randomly from the set N with size $m < |N|$ and starts its NC update procedure. In every round, H_{new} measures its RTTs to these m hosts as well as retrieves the incoming and outgoing vectors of these m hosts. Then its NC can be calculated and updated periodically. We will propose the detailed NC calculation algorithm in section 3.2.

For the first m hosts of the Phoenix system, the NC calculation is a bit different. If $|N| \leq m$, the new host H_{new} will be considered as one of the *early hosts*. These early hosts will probe each other to obtain the $|N| \times |N|$ distance matrix. The system will use NMF algorithm [11] to get the incoming vectors and the outgoing vectors of these early hosts.

3.2 Weighted Non-negative Least Squares NC Calculation

In Phoenix, we use a weighted non-negative least squares module to calculate the NC. The intuition behind this weighted NC calculation is as follows. The more accurate the referred NC (vector) is, the higher confidence (weight) should be given to this NC. In contrast, some referred NCs with abnormal high error will not be considered for NC calculation. Similar intuition is also used in decentralized Euclidean distance based NC systems such as [21].

For a host H_{new}, it has m reference hosts namely $R_1, R_2, ..., R_m$. The outgoing vectors of these m reference hosts are $X_1, X_2, ..., X_m$ and the incoming vectors of these m reference hosts are $Y_1, Y_2, ..., Y_m$. We define w_{X_i} as the weight of vector X_i and w_{Y_i} as the weight of vector Y_i. In our design, for all X_i and Y_i, we have $0 \leqslant w_{X_i} \leqslant 1$ and $0 \leqslant w_{Y_i} \leqslant 1$.

Suppose D_i^{out} is the distance from the host H_{new} to reference host R_i, D_i^{in} is the distance from reference host R_i to the host H_{new}. The solution of the X_{new} and Y_{new} with the weighted non-negative least squares error is as follows.

$$\boldsymbol{X_{new}} = arg \min_{U \in R^{+d}} \sum_{i=1}^{m} w_{Y_i} (D_i^{out} - \boldsymbol{U} \cdot \boldsymbol{Y_i})^2 \tag{4}$$

$$\boldsymbol{Y_{new}} = arg \min_{U \in R^{+d}} \sum_{i=1}^{m} w_{X_i} (D_i^{in} - \boldsymbol{X_i} \cdot \boldsymbol{U})^2 \tag{5}$$

We define an $m \times 1$ matrix $D_W^{out} = [\sqrt{w_{Y_1}} D_1^{out} \ \sqrt{w_{Y_2}} D_2^{out} \ ... \ \sqrt{w_{Y_m}} D_m^{out}]^T$ and an $m \times d$ matrix Y_W with $\sqrt{w_{Y_i}} Y_i$ as its row vectors. Therefore the Eq.(4) can be solved as follows.

$$\boldsymbol{X_{new}} = arg \min \|Y_W \boldsymbol{X_{new}^T} - D_W^{out}\|, \ s.t. \boldsymbol{X_{new}} \geqslant 0. \tag{6}$$

Likewise,we define an $m \times 1$ matrix $D_W^{in} = [\sqrt{w_{X_1}} D_1^{in} \ \sqrt{w_{X_2}} D_2^{in} \ ... \ \sqrt{w_{X_m}} D_m^{in}]^T$ and an $m \times d$ matrix X_W with $\sqrt{w_{X_i}} X_i$ as its row vectors. Therefore the Eq.(5) can be solved as follows.

$$\boldsymbol{Y_{new}} = arg \min \|X_W \boldsymbol{Y_{new}^T} - D_W^{in}\|, \ s.t. \boldsymbol{Y_{new}} \geqslant 0. \tag{7}$$

Eq. (6)-(7) are non-negative least squares constraints problems, they can be solved by the algorithm proposed in [20]. In our simulator [33] written in Matlab 6.0, the function called *lsqnonneg* is used to solve the nonnegative least-squares constraints problem.

In Phoenix, we calculate the weight as follows. If H_{new} is a newly joined host, we set all $w_{X_j} = w_{Y_j} = 1$ and calculate its initial NC. From this round, it applies Eq.(8)-(11) to determine the weights and updates its NC periodically.

For each reference host j of H_{new}, we define the error of each vector as follows.

$$\begin{aligned} E_{X_j} &= \|X_j \cdot Y_{H_{new}} - D(j, H_{new})\| \\ E_{Y_j} &= \|X_{H_{new}} \cdot Y_j - D(H_{new}, j)\| \end{aligned} \tag{8}$$

Thus the median values of both the error of $E_{X_i}(1 \leq i \leq m)$ and $E_{Y_i}(1 \leq i \leq m)$ are

$$\begin{aligned} M_X &= median_i(E_{X_i}) \\ M_Y &= median_i(E_{Y_i}) \end{aligned} \tag{9}$$

Then we can get the w_{X_j} and w_{Y_j} as follows.

$$w_{X_j} = \begin{cases} 1, & \text{if } E_{X_j} < M_X; \\ M_X/E_{X_j}, & \text{if } M_X < E_{X_j} < C * M_X; \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

$$w_{Y_j} = \begin{cases} 1, & \text{if } E_{Y_j} < M_Y; \\ M_Y/E_{Y_j}, & \text{if } M_Y < E_{Y_j} < C * M_Y; \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

C is set as 5 in our Phoenix implementation.

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(·), X(·), Y(·))
  if round = 1 then
    w_{X_j} = w_{Y_j} = 1
    [X_new, Y_new] = NC_Calculation(d(·), X(·), Y(·), w_X(·), w_Y(·))
  end if
  [w_X(·), w_Y(·)] = Weight_Calculation(D(·), X(·), Y(·), X_new, Y_new)
  [X_new, Y_new] = NC_Calculation(D(·), X(·), Y(·), w_X(·), w_Y(·))
  Wait(Update_Interval);
  round = round + 1
end while
```

Algorithm 1 shows the procedure of a new host joining the Phoenix NC system (Suppose this host is not one of the early hosts). This new host first contacts the Rendezvous Point (RP) of the Phoenix system like all other P2P schemes. After obtaining a list of reference host candidates, it contacts them and select m hosts out of them as its reference hosts. Then it starts its NC calculation and updates its NC periodically using the weighted NC calculation model. In every round, the new host measures its RTTs to its reference hosts as well as retrieves their NCs before the weighted NC calculation. Specifically, in its first round, it will calculate its initial NC using non-negative least squares (without introducing weight), then the weighted model can be applied from then on.

Table 1. TIV of the Data Sets

DataSet	*Hosts*	*triples in TIVs*	*pairs in TIVs*
AMP	110	4.29%	48.07%
PlanetLab	169	25.71%	86.53%
King	1740	12.32%	85.52%
P2PSim	1143	17.10%	97.33%
Meridian	2500	23.50%	96.55%

4 Performance Evaluation

4.1 Setup of the Experiment

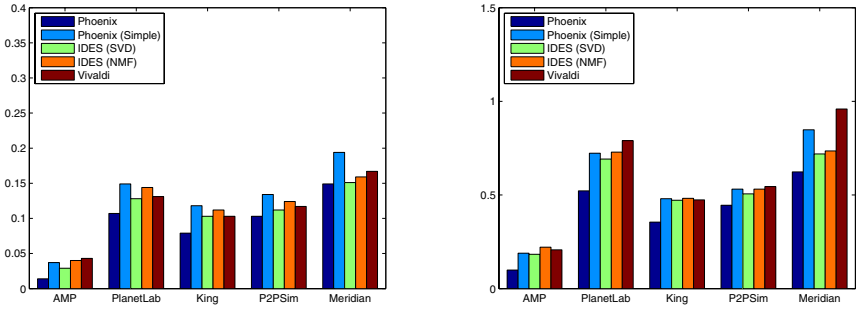
In our experiment, we compare Phoenix with IDES and Vivaldi. All of these three systems use 10-dimensional coordinates. In Phoenix, each host has m reference hosts. Likewise, there are m randomly selected landmarks in IDES. In Vivaldi, c_c and c_e are set to 0.25 as an empirical value and each host has m neighbors. In our experiments, we set m as 32. 10 runs are performed on each data set and the average results are reported.

We use five typical data sets from real Internet measurement to study the prediction accuracy of different NC systems. The first data set is the AMP data set [9], which includes the RTTs among 110 Internet hosts. The hosts are mainly at NSF supported HPC sites, with about 10% outside the US. The second data set is the PlanetLab data set [9], which includes the RTTs among 169 PlanetLab [23] hosts all over the world. The third data set is King data set which includes the RTTs among 1740 Internet DNS servers [19]. The fourth data set is P2PSim data set, which includes the RTTs among 1143 Internet DNS servers. These DNS servers were obtained from an Internet scale Gnutella network trace [9]. The fifth data set is Meridian data set which is from the Cornell Meridian project [27]. It measures the pairwise RTTs between 2500 hosts.

The effect of TIV is studied on these five data sets with two metrics proposed in [30]. The triple of hosts violating the triangle inequality is called a bad triangle. The first metric is *triples in TIVs*, which is defined as the percentage of triples that form bad triangles. The other metric is *pairs in TIVs*, which is defined as the percentage of pairs of hosts that are long sides in bad triangles (i.e., pairs that have an alternate shorter path). Table.1 shows the results. It is obvious that TIVs are quite different among these data sets. Thus we can evaluate our Phoenix NC system in different Internet delay structures.

4.2 Evaluation Results on Prediction Accuracy

In this subsection, we compare the REs of IDES, Vivaldi and Phoenix. More precisely, we evaluate both IDES(SVD) and IDES(NMF). Besides the complete



(a) Median Relative Error (b) 90th Percentile Relative Error

Fig. 2. Prediction Accuracy Comparison

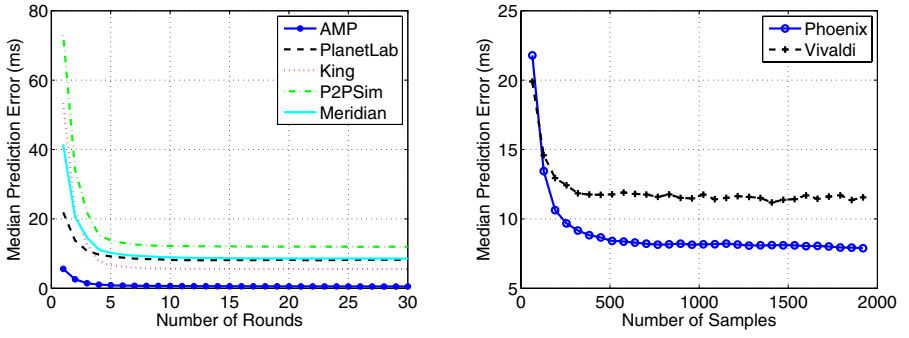
Phoenix, we also show the results of Phoenix(Simple). In the NC calculation procedure, Phoenix(Simple) doesn't use weighted model. It just simply sets all the weight to be 1. The comparison between Phoenix and Phoenix(Simple) demonstrates the improvement of the weighted model. Thus we have five different implementations of NC systems in this comparison.

Fig.2 shows the comparison between these five NC implementations under five different data sets. As in [18, 9], more attention is paid to the 90th Percentile Relative Error (NPRE) since it can guarantee 90% of the hosts have lower RE values than it. In all these five data sets, Phoenix performs the best. The performance of Phoenix(Simple) is close to IDES(SVD) and IDES(NMF) because weighted model hasn't been applied on it. But it's still an improvement over IDES since it will never give negative predicted distance. Compared with Vivaldi, the representative Euclidean distance based NC, Phoenix can reduce the NPRE by between 18.34% (P2PSim data set) and 52.17% (AMP data set). Our simulation results demonstrate that Phoenix can achieve high prediction accuracy in a decentralized and practical way.

4.3 Convergence Behavior of Phoenix

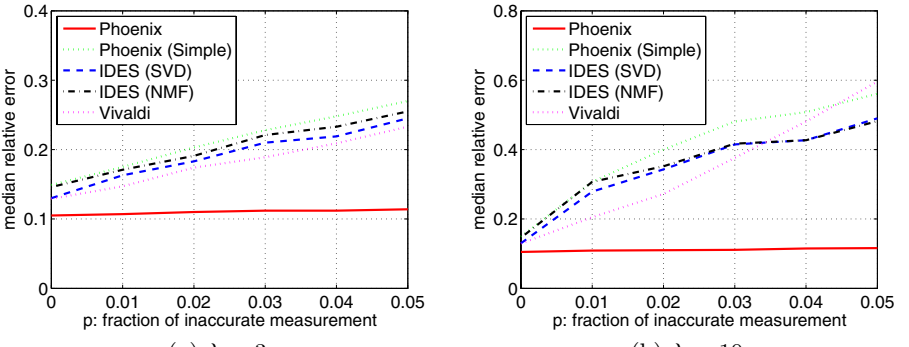
In this subsection, we study Phoenix in terms of the number of rounds (samples) required for convergence under a flash-crowd scenario, i.e., all hosts join simultaneously. We define *median prediction error* as $median_{i,j}(\|D^E(i,j) - D(i,j)\|)$. As in [19], we plots the median prediction error as a function of NC update rounds used per host in Fig.4(a). We can see in all the five data sets, the convergence of Phoenix is fast according to our simulation results. Basically, Phoenix will converge in less than 10 rounds.

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 neighbors and each neighbor has incoming



(a) Median Prediction Error (b) Phoenix versus Vivaldi

Fig. 3. Convergence Behavior of Phoenix



(a) $\lambda = 3$ (b) $\lambda = 10$

Fig. 4. The Impact on Measurement Anomalies

vector and outgoing vector, thus each update round needs 64 samples. We have done the comparison on all the five data sets and have drawn similar conclusions. Due to space limitation, we only show the results on PlanetLab data set. In Fig.4(b), we can see only in first round (first 64 samples), the median prediction error in Phoenix is 9% larger than that of Vivaldi. Thereafter, Phoenix converges very fast. The median prediction error in Phoenix needs less than 200 samples to converge to 12ms whereas in Vivaldi it needs about 300 samples. Also, we can find that the final median prediction error of Phoenix is about 31% smaller than Vivaldi. Thus the convergence of Phoenix is very fast and effective.

4.4 Robustness over Measurement Anomalies

In previous study, 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 [9]. A robust NC system will still give accurate prediction under a small amount of measurement anomalies. The robustness of these systems are evaluated by the experiment proposed in [9]. For a certain data set, we randomly pick p percent links of the data set and increase the RTTs of these links to λ times of the original values. Then we run NC system on the modified data set. When calculating the RE after the simulation, we compare the predicted distances to the actual network distances (i.e. the original value without injected errors). We vary the p value and see the evolution of median RE. As in [9], λ value is set as 3 and 10.

We have done the experiment on all the five data sets and have drawn similar conclusions. Due to space limitation, we only show the results on PlanetLab data set in Fig. 4. In IDES, Vivaldi and Phoenix(Simple) system, the amount of median RE increases when the amount of inaccurately measured links increases. The higher the degree of measurement error λ is, the faster the median RE increases along with p. In contrast, the results in Phoenix are quite different, while median RE only increases slightly along with p. Thus Phoenix is very robust to small amount of measurements anomalies. The difference between Phoenix and Phoenix(Simple) demonstrates that the weighted model can eliminate the impact of measurement anomalies greatly.

5 Conclusion and Future Work

In this paper we proposed the design and implementation of a decentralized dot product based NC system, Phoenix. Phoenix employs a weighted NC calculation model to reduce the effect of error propagation and it's a practical system which never gives negative predicted distance. Extensive simulation results with real Internet traces show that Phoenix achieves much higher prediction accuracy than state-of-the-art NC systems in different typical Internet data sets. We have also demonstrated that the convergence of Phoenix is fast and it performs robustly over small amount of measurement anomalies. Compared with IDES, Phoenix not only has better prediction accuracy but also can guarantee all the predicted distances non-negative. In short, Phoenix is an accurate, practical and decentralized solution to scalable Internet distance prediction.

Our future work of Phoenix is wide-area deployment. We will improve Phoenix in large scale Internet experiments and we believe Phoenix will be a robust, accurate and widely-used NC system.

Acknowledgment

This work is supported by the National Science Foundation of China (No.60473087, No.60703052, No.60850003) and National Basic Research Program of China (No.2007CB310806). Thanks to Mr. Zengbin Zhang from Tsinghua University for his comments and suggestions.

References

1. Zhang, R.M., Tang, C.Q., Hu, Y.C., et al.: Impact of the Inaccuracy of Distance Prediction Algorithms on Internet Applications: an Analytical and Comparative Study. In: Proc. of IEEE INFOCOM (2006)
2. Pietzuch, P., Ledlie, J.: Network-aware operator placement for stream-processing systems. In: Proc. of ICDE (2006)
3. Azureus bittorrent, `http://azureus.sourceforge.net/`
4. Zhang, B., Ng, T.S.E., Nandi, A., et al.: Measurement-Based Analysis, Modeling, and Synthesis of the Internet Delay Space. In: Proc. of ACM IMC (2006)
5. Abraham, I., Malkhi, D.: Compact routing on euclidian metrics. In: Proc. of ACM PODC (2004)
6. Wang, G., Chen, Y., Shi, L., Lua, E.K., Deng, B.X., Li, X.: Proxima: Towards Lightweight and Flexible Anycast Service. In: Proc. of IEEE INFOCOM Student Workshop (2009)
7. Lee, S., Zhang, Z., Sahu, S., et al.: On Suitability of Euclidean Embedding of Internet Hosts. In: Proc. of ACM SIGMetrics/Performance (2006)
8. Wang, G., Zhang, B., Ng, T.S.E.: Towards Network Triangle Inequality Violation Aware Distributed Systems. In: Proc. of ACM IMC (2007)
9. Mao, Y., Saul, L., Smith, J.M.: IDES: An Internet Distance Estimation Service for Large Network. IEEE Journal on Selected Areas in Communications, JSAC (2006)
10. Lua, E.K., Griffin, T., Pias, M., et al.: On the Accuracy of Embeddings for Internet Coordinate Systems. In: Proc. of ACM IMC (2005)
11. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–791 (1999)
12. Pias, M., Crowcroft, J., Wilbur, S., et al.: Lighthouses for Scalable Distributed Location. In: Proc. of IPTPS (2003)
13. Zhang, R., Hu, Y.C., Lin, X., et al.: A Hierarchical Approach to Internet Distance Prediction. In: Proc. of IEEE ICDCS (2006)
14. Tang, L., Crovella, M.: Virtual Landmarks for the Internet. In: Proc. of ACM IMC (2003)
15. Tang, L., Crovella, M.: Geometric Exploration of the Landmark Selection Problem. In: Barakat, C., Pratt, I. (eds.) PAM 2004. LNCS, vol. 3015, pp. 63–72. Springer, Heidelberg (2004)
16. Chen, Y., Xiong, Y., Shi, X., et al.: Pharos: Accurate and Decentralised Network Coordinate System. IET Communications (to appear)
17. Chen, Y., Zhao, G., Li, A., et al.: Handling Node Churn in Decentralised Network Coordinate System. IET Communications (to appear)
18. Ng, T.S.E., Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches. In: Proc. of IEEE INFOCOM (2002)
19. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: A Decentralized Network Coordinate System. In: Proc. of ACM SIGCOMM (2004)
20. Lawson, C.L., Hanson, R.J. (eds.): Solving Least Squares Problems, ch. 23, p. 161. Prentice-Hall, Englewood Cliffs (1974)
21. Lehman, L., Lerman, S.: A Decentralized Network Coordinate System for Robust Internet Distance Prediction. In: Proc. of ITNG (2006)
22. Ledlie, J., Gardner, P., Seltzer, M.: Network Coordinates in the Wild. In: Proc. of NSDI (2007)
23. PlanetLab, `http://www.planet-lab.org/` (accessed, November 2008)

24. Shavitt, Y., Tankel, T.: Big-Bang Simulation for Embedding Network Distances in Euclidean Space. In: Proc. of INFOCOM (2003)
25. Zheng, H., Lua, E.K., Pias, M., et al.: Internet Routing Policies and Round-Trip-Times. In: Proc. of the Passive Active Measurement Workshop (2005)
26. Ng, T.S.E., Zhang, H.: A Network Positioning System for the Internet. In: Proc. of USENIX Annual Technical Conference (2004)
27. Wong, B., Slivkins, A., Sirer, E.G.: Meridian: A Lightweight Network Location Service without Virtual Coordinates. In: Proc. of ACM SIGCOMM (2005)
28. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling Churn in a DHT. In: Proc. of the USENIX Annual Technical Conference (2004)
29. IDES Simulator, `http://www.research.att.com/~maoy/ides-0.1.tar.gz`
30. Lumezanu, C., Levin, D., Spring, N.: PeerWise Discovery and Negotiation of Faster Paths. In: Proc. of HotNets (2007)
31. Costa, M., Castro, M., Rowstron, A., et al.: PIC: Practical Internet Coordinates for Distance Estimation. In: Proc. of IEEE ICDCS (2004)
32. Lua, E.K., Griffin, T.: Embeddable Overlay Networks. In: Proc. of IEEE ISCC (2007)
33. Phoenix Simulator, `http://www.net-glyph.org/~chenyang/Phoenix-sim.zip`