Taming the Triangle Inequality Violations with Network Coordinate System on Real Internet*

Yibo Zhu[1], Yang Chen[2], Zengbin Zhang[3], Xiaoming Fu[2], Dan Li[4], Beixing Deng[1] and Xing Li[1]
[1] Department of Electronic Engineering, Tsinghua University, Beijing, China
[2] Institute of Computer Science, University of Goettingen, Goettingen, Germany
[3] Department of Computer Science, UC Santa Barbara, USA
[4] Department of Computer Science, Tsinghua University, Beijing, China
E-mail: zyb07@mails.tsinghua.edu.cn, yang.chen@cs.uni-goettingen.de

ABSTRACT

Network Coordinate (NC) systems are efficient in scalable Internet latency estimation. While most of the focus has been put on how to distort Triangle Inequality Violation (TIV) in metric spaces to relieve the inaccuracy caused by it, TIV is a persistently and widely existing phenomenon on the Internet and thus should be *embraced* by future NC systems rather than being eliminated. Besides high accuracy, such an NC system can also provide the benefit of reducing the data transmission time by use of proper relay routes. With that in mind, we design an NC system with a hierarchical architecture, which is motivated by the natural idea of partitioning the three TIV links into different autonomous NC systems, in order to make as many as TIVs inherently embeddable in metric space. We implement and deploy our work, named *Toread*, on real Internet. Evaluation results show that *Toread*'s metric space can well characterize more than 60% TIVs, thus *Toread* is highly accurate (0.54 in *Toread* versus 1.06 in *Pyxida* at 90th percentile Relative Error) and effective in searching detour paths (succeeds in 58.2% cases).

1. INTRODUCTION

Network Coordinate (NC) systems is efficient and lightweight in the scalable prediction of Internet latencies. Most NC systems are based on the Euclidean distance model which obeys Triangle Inequality. As a result, these systems are incapable to represent Triangle Inequality Violations (TIVs) in their metric spaces, therefore the performance is seriously impacted. In literature, methods have been proposed to avoid coordinates update from serious TIV edges, e.g. detecting and then abandoning serious TIV edges [16]. These methods relieved the severity of the problem but didn't solve it fundamentally.

*The work is supported by the National Science Foundation of China (No. 60850003).

ACM ReArch 2010, November 30, 2010, Philadelphia, USA.

TIV is a phenomenon that is widely existing among end-to-end latencies on real Internet, thus should be considered as a characteristic of Internet [13]. They should not be effaced from modeling. Furthermore, TIV also provides a good opportunity to improve the overlay routing by using the faster-than-direct relay routes (*detours*). Since such TIV-caused detour routes tend to have high embedding errors in a traditional NC system, *PeerWise* [12] proposed a method to search detours among the high-embedding-error hosts and links. However, despite the extra overhead introduced, the NC system used here has lost its original purpose, i.e., accurately estimating the latencies on Internet.

For the above reasons, we expect future NC systems to respect the natural existence of TIVs and retain the feature of TIVs instead of simply eliminating them. By properly embedding TIVs into metric space, NC system can 1) achieve **better accuracy**; 2) provide TIV-related service such as **detour searching**. Moreover, TIVs on Internet are highly dynamic and behave differently from aggregate datasets [13]. It's essential to design a practical NC system on Internet that addresses TIV embedding problem.

In this paper, we design, implement and deploy a practical NC system called *Toread* [15]. Our motivation is to break a triangle's three links into two inherently uncorrelated parts, each of which is estimated by one overlay. In this way, we can embed TIVs into metric space. Our study shows that by introducing an extra layer, on which hosts are properly assigned into clusters, more than 60% TIVs become embeddable. We call this extra layer *cluster overlay*.

We deploy the Toread service on PlanetLab. As a representative demonstration, we apply Toread to assist the widely used online game system. Our results show that Toread outperforms traditional NC systems in two folds. On one hand, Toread can achieve higher accuracy than existing deployed NC systems. According to our experimental results, Toread produces 90th percentile Relative Error (NPRE) [17] about half less than Pyxida (0.54 versus 1.06). On the other hand, Toread is effective in searching detours. Nearly 60% of the links' detours, if exist, can be found by Toread with five extra measurements.

The rest of the paper is organized as follows. In Section 2, we describe the design of Toread. The application of Toread in the context of an emulated online game system is discussed in Section 3. Section 4 presents experimental performance evaluation results. In Section 5, we review the background and related work of Toread. Finally we conclude this paper in Section 6.

2. DESIGN

2.1 Basic Definition

The main idea of Toread is to partition most TIVs into two isolated parties and the latencies are estimated within two different metric spaces, one is in global level shared by all hosts, while the other in cluster level shared by hosts in the cluster. In this way, the latency between two hosts from the same cluster is estimated by their cluster level NC. When they are from different clusters, their coordinates in global level overlay will be used. We denote the cluster that host A belongs to cluster C_A; peer B belongs to cluster C_B. Host A is embedded as $x_{A,g}$ in global overlay and $x_{A,c}$ in cluster overlay. The estimated latency $d^E(A,B)$ is calculated as follows:

$$d^E(A,B) = \begin{cases} |x_{A,c} - x_{B,c}| & \text{if } C_A = C_B \\ |x_{A,g} - x_{B,g}| & \text{if } C_A \neq C_B \end{cases} \quad (1)$$

Compared with previous NC systems such as Pyxida, Toread has following unique features. Firstly, the cluster overlay serves to break TIVs into uncorrelated parts, thus TIVs are inherently allowed in metric space. Secondly, links in one cluster can be relatively estimated by two overlays. Toread leverages both estimations in TIV discovery rather than discarding one of them.

2.2 Case Study: Why Cluster Overlay helps

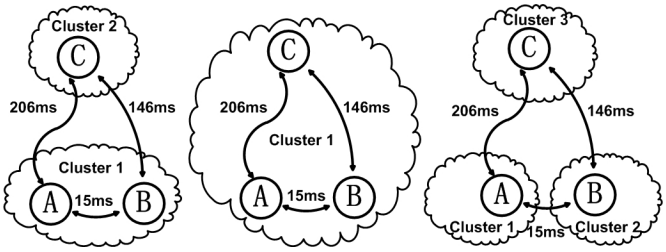


Figure 1: Three TIV conditions

In Toread, according to the metric spaces by which the links in TIVs are relatively estimated, TIVs can be classified into three categories: 1) two hosts belong to a cluster while the third belongs to another cluster; 2) all hosts belong to the same cluster; 3) each host belongs to a separate cluster. Figure 1 shows a TIV case we measured on PlanetLab and the three possible conditions. Let us focus only on the first condition for now: A and B are in a same cluster while C is in a different cluster. The prediction of AC and BC are made by global overlay metric space while AB is predicted by another overlay of NC, namely the cluster metric space. The cluster overlay is not influenced by host C so the TIV is embeddable in metric space.

However, among the three cases above, only the first condition can be embedded by Toread's architecture; we call them *embeddable TIVs*. Fortunately, we find out that more than 60% of TIVs on Internet have a similiar topological pattern. By properly assigning clusters, more than 60% TIVs turn out to be embeddable. The detailed results will be further discussed in section 2.3.

By embedding TIVs into metric space in this way, Toread can better model latency space on Internet. We run Toread on PlanetLab and calculated the Relative Error (RE) between predicted RTTs and real RTTs. Figure 2 shows the histogram of the RE on such links. When estimated in Toread's metric space, the portion of low-RE links is much larger than that in single-layer space, while the high-error ones (RE>1) decrease 1/3.

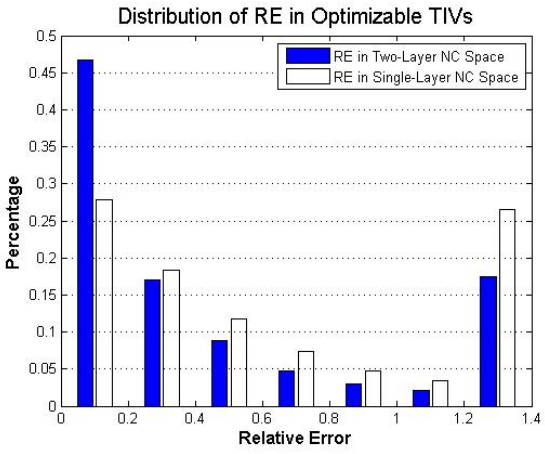


Figure 2: Two Layer NC vs. Single Layer NC on Estimating Links in Embeddable TIVs

Moreover, the cluster overlay provides Toread the opportunities to detect the embeddable TIVs by comparing two different estimations. It will be explained in section 2.4.

2.3 Cluster Assignment: For More Embeddable TIVs

2.3.1 Major Condition of TIVs

We study end-to-end latency we gathered during April 1-30, 2010 on nearly 400 PlanetLab hosts. According to the result, we observe that more than 60% of TIVs occur when two hosts are geographically near each other while the third host is far away from both of them. Due to the natural near-linearity between geographical distances and link latencies, this feature is well reflected on their latency topology.

Intuitively, this can be explained by the fact that Border Gateway Protocol (BGP) is more likely to miss the best route on distant links. Once the longest link in a triangle is routed with large deviation, it may exceed the sum of delays on the other two links. For example, in Figure 1, A denotes *sanfrancisco.planetlab.pch.net*, while B is *planetlab8.millennium.berkeley.edu* and C is *csplanetlab4.kaist.ac.kr*. A and B both locate in America, while C locates in Asia. We can infer from this TIV that the route between A and C is far from the best. We call these TIVs *major TIVs*.

2.3.2 Anchor Selection

The feature of major TIVs suggests that if links within a cluster have small latencies while high delay links are out of the cluster, the links in major TIVs should be estimated by two metric spaces. Thus the major TIVs turn out to be embeddable TIVs.

In Toread, some always-on hosts such as DNS servers, called *Anchors*, help hosts decide which cluster to join. Every cluster has at least one Anchor. Hosts join the clusters whose Anchors have the least latencies to them. Figure 3

is a snapshot of hosts' global layer, which is a 2 dimensions Euclidean metric space, in 3-cluster Toread on PlanetLab. Every dot means a host which has the same global layer 2D coordinates. The dot is colored in accordance with its cluster. Clearly, with the help of Anchors, hosts are partitioned into three aggregations.

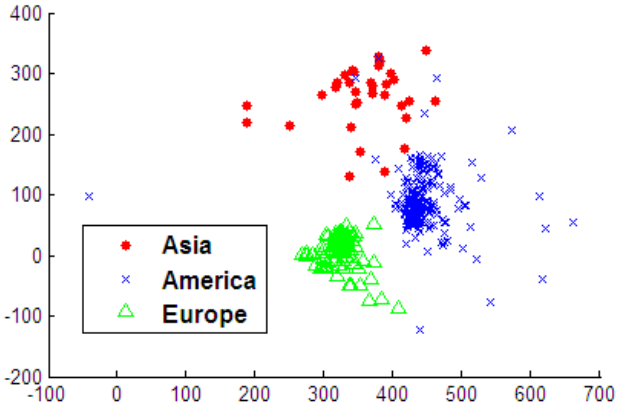


Figure 3: Hosts' NC Scatter Gram

Anchor Number	One Anchor	Three Anchors
Embeddable TIV	58.6%	64.4%
Avg. RTT in a Cluster	69.3ms	65.4ms
<100ms in a Cluster	83.5%	86.8%

Table 1: Single Anchor vs. Multi-Anchors

In order to eliminate the measurement outliers to Anchors that happen accidentally, we propose that each cluster in Toread has several Anchors. Peers will choose clusters by the median of the RTTs to Anchors. In Table 1, we compare multi-anchor Toread and single-anchor Toread.

In the case of three-anchors, the ratio of embeddable TIVs reaches 64.4%. Besides, there are more less-than-100ms links which can be less impacted by TIVs because they are estimated by cluster overlay. Less-than-100ms delay is usually the basic requirements for Internet applications such as online gaming. In sum, the increase of Anchors benefits Toread.

2.3.3 Number of Clusters

The portion of embeddable TIVs is relevant to the number of clusters. By increasing the number of clusters, some of the second kind of TIVs described in subsection 2.2 will turn out to be embeddable. At the same time, some embeddable TIVs possibly become the third kind of TIVs thus no longer embeddable.

For comparison, we relatively ran single-anchor Toread of different numbers of clusters on PlanetLab. The clusters in 3-cluster Toread are America, Europe and Asia. The 4-cluster Toread replaces America with west America and east America. The 5-cluster Toread further replaces Europe with west Europe and east Europe. The portions of 3 kinds of TIVs are included in Figure 4.

With the increase of cluster number, the proportion of 2nd TIV condition decreases rapidly while the 3rd TIV condition

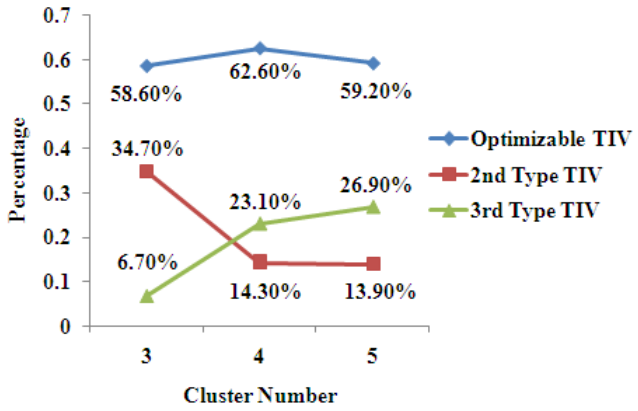


Figure 4: TIV Proportions vs. Cluster Number

increases. When the cluster number is 4, the proportion of embeddable TIV is largest.

2.4 Searching Detours

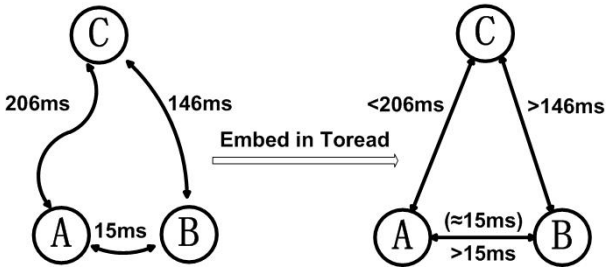


Figure 5: TIV Embedded in Toread

By better modeling TIVs, Toread can also provide applications faster-than-direct detour searching service. For a given link, Toread is focusing on finding TIVs in which the target link is the longest. By selecting the other two-hop route, Toread can provide a accelerating detour. Also, a link under large deviation tend to have multiple detours. Toread is able to select the shortest in this case.

When embedding a TIV host triple into a metric space that obeys Triangle Inequality, all of the links in the TIV are impacted. The longest link's latency will be under estimated, while the short links will be over-estimated. Figure 5 still takes the previous three hosts A,B and C as an example. When host A searches detours for link AC, Toread will look for relay candidates that are either in the same cluster as A or as C.

We say A finds B as one of the candidates. The link AB will be better estimated by cluster layer. However, we find that it is not enough for accurately estimating the latency between A and C relaying B because link BC is also impacted by the TIV. Toread modifies its estimation of BC with the ratio of AB's cluster estimation to AB's global estimation (that ratio implies the embed error of B in global layer). Denote estimated latency between A and C with relay B as $d^E(A, C; B)$, estimated latency of link AB in cluster overlay as $d^E(A, B)_c$ and in global overlay as $d^E(A, B)_g$, Toread predicts the latency as:

$$d^E(A,C;B) = d^E(A,B)_c + \frac{d^E(A,B)_c}{d^E(A,B)_g} \times d^E(B,C)_g \quad (2)$$

Toread makes predictions for candidate detours and then ranks them. Then, measurements are conducted to verify the 1 to 5 predicted shortest paths plus the direct path. Finally Toread gives suggestion that whether to use a detour. The results will be discussed in Section 5.

3. IMPLEMENTATION

3.1 General Framework

Toread is a novel cross-platform NC system implementation. We code Toread in about 2000 lines of Python. Toread consists of the following modules:

1) *Ping Module*: Because some hosts don't respond to ICMP ping, an independent ping module in Toread is developed. This module can also be driven by Toread neighbors to measure target hosts in order to help neighbors verify the predicted detours.

2) *NC Calculation Module*: By NC Module, hosts running Toread can update their own NC with recent measurements and maintain its neighbors' NCs.

3) *RTT Estimation Module*: Toread calculates estimated RTT with coordinates and provides applications an API to acquire these results.

4) *Relay Indication Module*: Toread quickly measures up to five predicted fastest detours. Then Toread provides applications the best route to certain hosts.

Toread's work flow consists of the following steps (we take host A as an instance):

1) *Cluster Choice*: Host A firstly pings prior configured Anchors. A finds out that it has smallest RTTs to cluster C_A's Anchors thus it joins cluster C_A.

2) *Bootstrap*: Host A gets its first neighbors in both global layer and cluster layer from hosts in global's and C_A's bootstrap list.

3) *Maintenance of NCs*: Periodically, host A updates its global and cluster NC with algorithm Vivaldi [7]. It also responds to its neighbors in both global overlay and C_A.

Toread runs as a daemon and responds to applications' requests for RTT estimation and detours.

3.2 Implementation Details

We implement four clusters, each with three Anchors selected. The numbers are chosen according to the analysis in Section 2.3. The four clusters are west American cluster, east American cluster, European cluster and Asian cluster. The anchors were randomly selected from hosts locating in the above four regions.

Toread's global layer is a 4 Dimension with 1 Height [7] metric space. Nevertheless, the height is removed from cluster metric because we find out that it negatively impacts the accuracy on short links. An intuitive explanation is that a large height, which is possibly due to the impact of long links, will cause large errors on short links because the height is directly added into the estimation.

We configured Toread's total update period to be the same as Pyxida for fair comparison. Pyxida uses a 4 Dimension with 1 Height metric space by default. In this case, Toread's theoretical traffic for NC communications is slightly less than Pyxida (global's 4D+1H plus cluster's 4D vs. 4D+1H times two).

3.3 Toread Application: Online Game System

Online gaming is one of most popular Internet applications. The user experience is essential in online game systems. In a game, the delay between actions and effects, or *lag*, is the game players' major concern. For most of online games, the latencies between players' machines contributes the most to lag [1]. We apply Toread to assist an emulated online game system. Figure 6 shows the architecture.

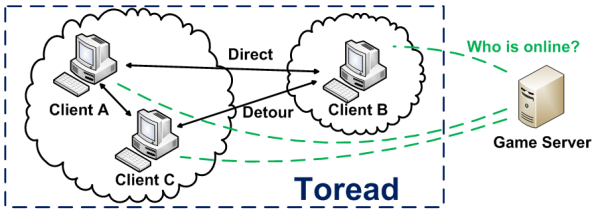


Figure 6: System Architecture

Toread improves the user experience by two means:

1) *Matchmaking*: Game players usually prefer less lag. In *matchmaking* [1], game players selects low delay opponents according to Toread's RTT estimations instead of brute-force measurements or random.

2) *Finding Detours*: However, game players not only care about lag, but also prefer playing with with opponents in the same level. So players sometimes have to play with slow links. Toread can search faster-than-direct detours for slow links, thus help game software determine whether to use a detour and which detour to used.

In section 4, we compare Toread's and Pyxida's performance in matchmaking. However, Pyxida doesn't have the function of seeking for detour. So we compare Toread' performance with the perfect results.

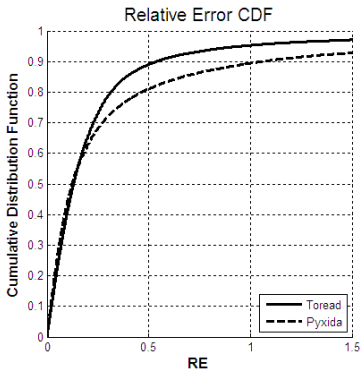
4. PERFORMANCE EVALUATION

From Mar 15, 2010 to Apr 15, 2010, we performed comprehensive experiments to verified Toread's performance of modeling TIVs in terms of latency estimation accuracy and detour searching. We firstly compare Toread's RE [17] and Closest Neighbor Absolute Error (CNAE) [17] with Pyxida's. Then emulated online game system demonstrates how the detour searching service along with higher estimation accuracy of Toread benefits a practical Internet application.

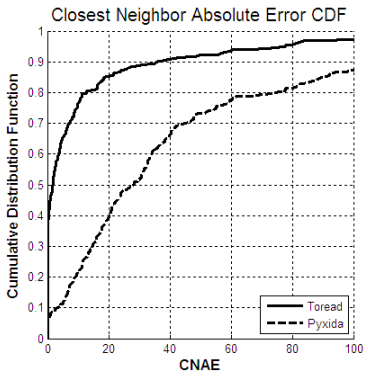
4.1 Experiment Setup

We deployed Toread on 335 hosts in PlanetLab. Every experiment lasts for 24 hours, and then we collect hosts' coordinates. Simultaneously, all hosts ping each other in a random order to retrieve real RTT matrix. We used Tcpdump to capture sent and received packets on Toread's ports during the entire experiment in order to evaluate Toread's overhead. For fair comparison, we run Pyxida and collect coordinates at the exact same time. The results are discussed in subsection 4.2.

In the online game system experiment, all the 335 hosts are emulated as players' machines. Every host runs Toread and is aware that the other 334 hosts are online and waiting for joining a game. Each host chooses the closest neighbor

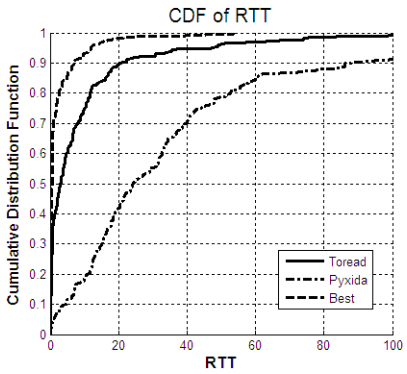


(a) RE CDF

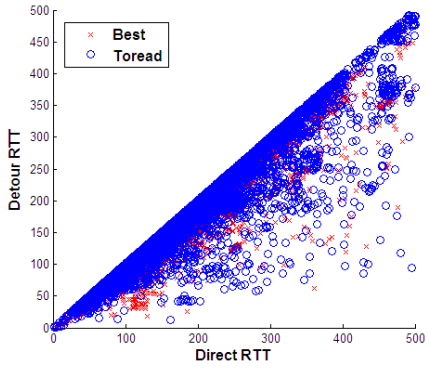


(b) CNAE CDF

Figure 7: Toread vs. Pyxida



(a) Selected RTT CDF



(b) Detour vs. Direct

Figure 8: Performance in Online Game System

according to Toread's or Pyxida's prediction. For all links between hosts, Toread tries to find detours. We studied the improvement brought by detours in subsection 4.3.

4.2 Estimation Accuracy and Overhead

In NC system performance evaluation, RE results indicate the general accuracy of NC systems; CNAE results indicate the NC systems' performance on closest neighbor selection. A host with zero CNAE means the host can correctly select the closest neighbor. All the REs and CNAEs are included in the following Cumulative Distribution Function (CDF) graph.

As Figure 7b shows, Toread is generally more accurate than Pyxida. Toread produces 90% percentile RE 49% less than Pyxida (0.54 versus 1.06). In CNAE CDF, Toread's advantage on short links prediction is even more impressively shown. In Pyxida, only 8% hosts can correctly choose the closest neighbor while in Toread, there are five times, i.e. 40%.

Our explanation is that, according to our observation, the large REs in Pyxida are produced by short links. Toread better estimates the short links in TIVs (usually these links are not cross clusters). The improvement of these links results in a significant advantage on estimating short links and thus results in an overall better RE results.

As Table 2 shows, Toread costs every peer less than 1KB/s Internet traffic. On average, Toread costs no more than Pyxida. In sum, Toread achieves higher performance without increasing the overhead.

System	Min(B/S)	Max(B/S)	Average(B/S)
Toread	290.73	587.46	423.26
Pyxida	562.44	1777.78	892.05

Table 2: Toread vs. Pyxida : Overhead

4.3 Emulated Online Game System

Figure 8a shows the performance comparison of the online game system equipped with Toread and Pyxida respectively. The ideal performance acquired by brute force measurement is also included.

We can see that Toread highly improves the system's performance, and is close to the ideal performance. In terms of finding less-than-50ms links, which are required for most applications, Toread's about 95% exceeds Pyxida's 80%. This result fits the CNAE results.

Figure 8b shows the comparison between detours found by Toread and best detours. In this case, Toread makes 5 measurements to verify predicted paths. This process usually costs less than 3 seconds, which is acceptable for a newly

established game. According to our data, 48.6% links can be improved by detours. Of these links, 58.2% are found by Toread. Considering the proportion of embeddable TIVs is about 64%, which is mentioned in section 2.3, this result shows Toread can model embeddable TIVs well. The improvement of latencies is about 15ms on average.

5. BACKGROUND AND RELATED WORK

NC systems embed hosts into a coordinate space with only $O(N)$ measurements. The predicted RTT between two nodes is defined as the distance between their coordinates. Many algorithms such as GNP [14], Vivaldi [7] and Phoenix [4] were proposed. Simulations showed that NC systems are effective in predicting latencies among hosts. Pyxida [10] and Htrae [1] are probably so far the only two existing practical implementations of decentralized NC systems. Unfortunately, TIVs are pernicious for them. None of the above algorithms or systems can inherently reveal the existence of TIVs.

In recent years, many NC system architectures have been proposed to improve the overall accuracy [3, 5, 6, 8, 17]. [5, 6] is our early hierarchical design with evaluations, but little understanding was given on why the accuracy can be improved in that way. Moreover, since those designs are all based on simulations with aggregate datasets, while unfortunately TIVs are often either concealed or false informed in these datasets, as [13] has mentioned, their real performance is actually unknown, and could potentially be impacted by the time-varying feature of Internet latencies.

[8] proposed a two-tier structure to address TIV issues. Our work differs from [8] in the following aspects. Firstly, [8] is trying to mitigate what TIVs impose on NC systems. However, Toread is designed under a totally different principle, i.e. **retaining** and **leveraging** the feature of TIVs. Furthermore, the clustering design of [8] is oracle-based, which means the system should know all the latencies among the hosts. It is impractical for a real system. Toread provides a whole practical solution along with optimizations towards its unique goal, i.e. inherently embedding as many as TIVs in metric space.

Internet's traffic flow can be optimized by offering detours existing in TIVs. PeerWise [12] is an effective system that searches detours referring to the hosts' embedded errors of Pyxida. The reason PeerWise utilized Pyxida is not that Pyxida is accurate, but, on the contrary, that Pyxida's accuracy is seriously impacted by TIVs. That means an Internet application may need to maintain another NC system besides Pyxida used in PeerWise for higher latency estimation accuracy. Toread is designed to be a win-win solution, which keeps good latency estimation accuracy as well as achieves accurate and efficient detour discovery.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose Toread, which is a decentralized NC system with a novel architecture design which addresses TIV modeling on real Internet. Our work demonstrates how the architecture of Toread divides the *major TIVs* and embed them into metric space while preserving their features. By taking this advantages in better modeling the Internet, Toread significantly outperforms the most widely deployed practical NC system Pyxida in terms of prediction accuracy. Besides that, we also demonstrate that Toread can provide detour searching with negligible measurement overhead.

Toread can benefit many future Internet applications, e.g. IPv6 based multihoming [9]. Moreover, Toread exposes the opportunity of better routing efficiency by searching detours in a scalable way. It can be a good enhancement to current routing systems and thus improve the performance of future Internet. We are going to design such protocol and system and evaluate it on CERNET2 [2] .

7. REFERENCES

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

[2] CERNET2, 2010. `http://www.cernet2.edu.cn/index_en.htm`

[3] Y. Chen, P. Sun, X. Fu, et al. Improving Prediction Accuracy of Matrix Factorization Based Network Coordinate Systems. In Proc. of IEEE ICCCN, 2010.

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

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

[6] Y. Chen, Y. Xiong, X. Shi, et al. Pharos: Accurate and Decentralised Network Coordinate System. IET Communications, 2009, 3(4):539-548.

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

[8] M. A. Kaafar, B. Gueye, F. Cantin, et al. Towards a Two-tier Internet coordinate system to mitigate the impact of triangle inequality violations. In Proc. of IFIP-TC6 Networking, 2008.

[9] C. Launois, S. Uhlig, O. Bonaventure. Scalable Route Selection for IPv6 Multihomed Sites. In Proc. of IFIP-TC6 Networking, 2005.

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

[11] J. Ledlie, P. Pietzuch, and M. Seltzer. Stable and Accurate Network Coordinates. In Proc. of ICDCS 2006.

[12] C. Lumezanu, R. Baden, D. Levin, et al. Symbiotic relationships in Internet routing overlays. In Proc. of NSDI, 2009.

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

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

[15] Toread Project, 2010. `http://code.google.com/p/toread/`.

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

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