NCShield: Protecting Decentralized, Matrix Factorization-Based Network Coordinate Systems

Yang Chen, *Senior Member, IEEE,* Shining Wu, Jun Li, *Senior Member, IEEE,* and Xiaoming Fu, *Senior Member, IEEE*

***Abstract*—Network Coordinate (NC) systems provide a scalable means for Internet distance prediction and are useful for various Internet-based services, such as cloud or web-based services. Decentralized, matrix factorization-based NC (MFNC) systems have received particular attention recently. They can serve large-scale distributed services (as opposed to centralized NC systems) and do not need to satisfy the triangle inequality (as opposed to Euclidean-based NC systems). However, because of their decentralized nature, MFNC systems are vulnerable to various malicious attacks.**

In this paper, we provide the first study on attacks toward MFNC systems, and propose a trust and reputation-based approach called *NCShield* to counter such attacks. It is fully decentralized and can easily be customized. Different from previous approaches, NCShield is able to distinguish between legitimate distance variations and malicious distance alterations. Using four representative data sets from the Internet, we show that NCShield can defend against not only the typical disorder, repulsion and isolation attacks, but also more advanced attacks such as frog-boiling attacks. For example, when selecting node pairs with a shorter distance than a predefined threshold in an online game scenario, even if 30% nodes are malicious, NCShield can reduce the false positive rate from 45.5% to 3.7%.

***Index Terms*—Service computing, Internet topology, network coordinate systems, security**

I. Introduction

Nowadays the Internet plays a key role in people's daily-life. Numerous online services are provided through the Internet. To serve end users all over the world, online service providers always want to build an efficient networking infrastructure by carefully considering the underlying Internet topology. The end-to-end distance information (a.k.a. Round Trip Time or RTT) can help determine the proximity among Internet nodes. In a network of N nodes, there are $N(N-1)/2$ end-to-end paths. For large-scale distributed systems, obtaining the distance information of all end-to-end node pairs is extremely difficult due to the high measurement overhead. To provide a scalable Internet distance prediction, Network Coordinate (NC) systems have been widely used. Using NC systems, we only need $O(N)$ measurements to scalably predict the distances of all $N(N-1)/2$ end-to-end paths, thus significantly reducing the measurement overheads. NC systems can be used in various distributed applications, such as cloud or web-based services [15], [20], [26], [29], [35], [44], [51], overlay networks [40], [48], anonymous communications [39], network monitoring [9], [37], and online social networks [10], [49]. NC systems can play an important role particularly in the emerging field of services computing. Existing examples include selecting servers for cloud services [15], [26], [44], composing cloud services [20], positioning web services [51], locating cloud resources [35], and aggregating resources across multiple data centers [29].

NC systems follow two basic models. Most traditional NC systems (such as GNP [33], PIC [13], NPS [34], and Vivaldi [14]) are Euclidean-based NC (ENC) systems. All nodes are embedded in an Euclidean space R^d, and every node is assigned a d-dimensional coordinate ($d \ll N$). The distance between any two nodes can be predicted by typical Euclidean distance calculation. However, ENC systems require the estimated distances among every three Internet nodes to satisfy the triangle inequality, a condition that often does not hold true on today's Internet [21], [25], [30], [32], [50]. The other model, which has recently received much attention, is the matrix factorization-based NC (MFNC) systems, including IDES [32], Phoenix [9], and DMFSGD [27]. MFNC systems do not have the triangle inequality constraint anymore, and thus can achieve a better distance prediction accuracy than ENC systems. Such improvement in distance prediction accuracy will be especially beneficial for cloud-based services, as today's public cloud providers always have multiple data centers around the world. Choosing the "closest" data center based on accurately predicted distances will minimize the data delivery latency, leading to a better experience for users.

An NC system can be either centralized or decentralized. Centralized NC systems, such as GNP [33] and IDES [32], rely on a small set of landmark nodes, which could easily become the scalability bottleneck. Nowadays a large number of online services and systems need to serve millions of users simultaneously (e.g., cloud or web-based services, CDNs, multi-player gaming), decentralized NC systems are the only feasible option for scalable distance prediction in such systems. We therefore focus on decentralized NC systems (such as Phoenix [9] and DMFSGD [27]) in this paper.

Decentralization, however, makes a decentralized NC system vulnerable to certain security attacks. While every node in the system can advertise to other nodes arbitrary information at its own discretion, malicious nodes in the system can falsify

Yang Chen is with the School of Computer Science, Fudan University, China, and the Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China.

Shining Wu and Xiaoming Fu are with the Institute of Computer Science, University of Göttingen, Germany.

Jun Li is with the Department of Computer and Information Science, University of Oregon, USA.

E-mail: chenyang@fudan.edu.cn, {shining.wu, fu}@cs.uni-goettingen.de, lijun@cs.uoregon.edu

coordinates or delay the responses to RTT probing packets in order to disrupt an NC system.

While several approaches [17], [36], [38], [46] have been proposed to secure ENC systems, unfortunately, little has been done towards MFNC systems which yields to higher accuracy than ENC systems when there are no attacks for both types of systems. It is thus interesting to study the impact of attacks to MFNC systems. We tackle this deficiency in this paper. In particular, we make the following contributions:

1) We formalize potential malicious attacks toward decentralized MFNC systems. So far, protecting MFNC systems from malicious attacks has not been considered, and no attack model for such systems exists. Also, through extensive evaluation using four representative data sets collected from the Internet, we show how these attacks can disrupt existing MFNC systems such as Phoenix [9] and DMFSGD [27].
2) We propose a trust and reputation-based approach, called **NCShield**, to defend decentralized MFNC systems. Different from the approaches for ENC systems, our solution is able to distinguish between ordinary distance variation and malicious distance alteration. When choosing nodes to calculate coordinates, instead of relying on additional infrastructures, such as distributed hash tables (DHTs) (as in [6]) or a centralized reputation computation agent (RCA) (as in [36]), NCShield uses secure gossip [4] to ensure lightweight and unbiased node sampling. Based on their scalable measurements, nodes can vote in a distributed way to identify malicious nodes.
3) NCShield is fully decentralized and can be easily integrated into existing MFNC systems. It is also easy to add new, customized modules to NCShield in order to tackle more advanced attacks, such as frog-boiling attacks.
4) NCShield achieves a high distance prediction accuracy. In our experiments using classic aggregate data sets, a dynamic data set, and an online game scenario, NCShield consistently shows a high accuracy in the distance prediction. For example, in the online game scenario when selecting node pairs with a shorter distance than a pre-defined threshold, even if 30% nodes in the system are malicious, NCShield can reduce the false positive rate from 45.5% to 3.7%.

Part of our manuscript appeared in [45]. Compared to [45], in this article we advance our study substantially in following several important ways:

1) Rather than inspecting the DMF [28] coordinate system that we studied earlier, in this article we instead inspect a new, more advanced network coordinate system called DMFSGD [27]. We not only study whether NCShield is effective in this new system (Section II), but also evaluate its performance with this new system (Section VI-B).
2) We further handle the new emerging frog-boiling attack [7] that has been found more harmful to NC systems. We describe and model this attack (Section III), introduce an anti-frog-boiling module into NCShield (Section IV-C), and evaluate NCShield's performance against this newly introduced attack (Section V-C, VII).
3) Furthermore, we upgrade the algorithm of NCShield in multiple ways. For example, it divides the original secure coordinate calculation process into the independent grading and voting steps (Section IV-B); besides conventional attacks, it now can cope with the new emerging frog-boiling attacks (Section IV-C).

Overall, compared to what we have published earlier, the NCShield system we presented in this article is more up-to-date in terms of network coordinate systems to protect, more robust and flexible in terms of attacks it can handle, and more comprehensive and informative in terms of the evaluation results.

The rest of the paper is organized as follows. We first describe the background of our work in Section II, including how decentralized MFNC systems function and how malicious nodes conduct attacks in decentralized ENC systems. We list prospective attacks towards MFNC systems in Section III, including both conventional attacks and advanced attacks. In Section IV we present our defense approach, NCShield. Then in Section V we describe the evaluation methodology and metrics. In Section VI, we present the communication overhead analysis and results of NCShield with aggregate data sets. In Section VII, we evaluate NCShield with a dynamic data set against all five attacks we modeled. In Section VIII, we emulate NCShield in an online game scenario with Phoenix to show its feasibility and effectiveness. We summarize the related work in Section IX and conclude this paper in Section X.

II. Background

In this section, we describe how a decentralized MFNC system works, how its accuracy is evaluated, and two representative MFNC systems—Phoenix and DMFSGD.

We assume that we have a network with N nodes. We can use an $N \times N$ matrix D to represent the Internet distance between each two of them, i.e., $D(i, j)$ represents the measured distance between node i and node j ($1 \leq i \leq N, 1 \leq j \leq N$). The key idea of MFNC systems is that a large $N \times N$ matrix D can be approximated by the product of two smaller $N \times d$ matrices X and Y with methods such as Singular Value Decomposition (SVD) [23] and Non-negative Matrix Factorization (NMF) [24]. The intuition behind this model is the low rank nature of Internet distance matrices [41].

In MFNC systems, for each host i, it will be assigned a d-dimensional outgoing vector X_i and a d-dimensional incoming vector Y_i ($d \ll N$). Therefore, the predicted distance between node i and node j is determined by the dot product of node i's outgoing vector and node j's incoming vector, as in Eq. 1, where $D^E(i, j)$ is the predicted distance from node i to node j, $\vec{X}_i$ and $\vec{Y}_j$ are respectively i's outgoing vector and j's incoming vector:

$$D^E(i, j) = \vec{X}_i \cdot \vec{Y}_j \tag{1}$$

Compared with ENC systems, MFNC systems do not have the restriction of the triangle inequality for predicted distances. Therefore, MFNC systems are able to achieve a much better

prediction accuracy [9], [27], as it does not need to satisfy the triangle inequality principle.

Two representative distributed MFNC systems are Phoenix and DMFSGD. We describe them below.

In the Phoenix NC system, assuming a node H has m neighbors as its reference nodes ($d < m << N$), its new coordinates will minimize both the error of the predicted distances from H to reference nodes and the error of the predicted distances from reference nodes to H, as shown in the two equations below:

$$\vec{X}_{new} = arg \min_{\vec{x} \in \Re_+^d} \sum_{i=1}^{m} w_{Y_i} \|\vec{x} \cdot \vec{Y}_i - D_i^{out}\|^2 \quad (2)$$

$$\vec{Y}_{new} = arg \min_{\vec{y} \in \Re_+^d} \sum_{i=1}^{m} w_{X_i} \|\vec{X}_i \cdot \vec{y} - D_i^{in}\|^2 \quad (3)$$

$\vec{X}_{new}$ and $\vec{Y}_{new}$ are the calculated d-dimensional outgoing/incoming vectors of node H. $\vec{X}_i$ and $\vec{Y}_i (1 \leq i \leq m, i \in \mathbb{N})$ are the outgoing vectors and incoming vectors of H's i-th reference node, respectively. D_i^{out} is the measured distance from node H to its reference node i, and D_i^{in} is the measured distance from reference node i to node H. The weights w_{X_i} and w_{Y_i} ($i = 1...m$) are both within the range of [0,1], and are calculated by a weight-based algorithm that improves the overall prediction accuracy by alleviating error propagation.

The DMFSGD NC system adopts a stochastic gradient descent (SGD) algorithm. This algorithm tries to update the coordinates gradually along the directions to minimize a regularized loss function. The update rules are:

$$\vec{X}_{new} = (1 - \eta\lambda)\vec{X}_{old} + \eta \sum_{i=1}^{m} (D_i^{out} - \vec{X}_{old}\vec{Y}_i^T)\vec{Y}_i \quad (4)$$

$$\vec{Y}_{new} = (1 - \eta\lambda)\vec{Y}_{old} + \eta \sum_{i=1}^{m} (D_i^{in} - \vec{X}_i\vec{Y}_{old}^T)\vec{X}_i \quad (5)$$

where η, called learning rate or step size, is a positive number which controls the speed of the updates; and λ, the regularization factor, is a fixed positive number which restricts the coordinates from overfitting and drifting gradually.

Finally, an NC system is considered accurate if the predicted distance between two nodes based on their coordinates is roughly equal to the measured distance.

For a pair of nodes i and j, denoting their predicted distance $D^E(i,j)$ and their measured distance $D(i,j)$, the accuracy of $D^E(i,j)$ can be evaluated using its *Relative Error (RE)* [8], [9], [11], [14], [25], [32]–[34] as shown in Eq. 6 below. Clearly, a RE has a non-negative value. If predicted distance equals to measured distance, the RE will be zero.

$$RE = \frac{|D^E(i,j) - D(i,j)|}{\min(D^E(i,j), D(i,j))} \quad (6)$$

For evaluating an NC system, the main metric that has been widely used is *ninetieth percentile relative error (NPRE)* [8], [9], [11], [32]–[34]. It guarantees that 90% of the links have a RE value lower than the NPRE value. A smaller NPRE value indicates a higher overall prediction accuracy [17], [36], [38], [46].

III. Prospective Attacks in Decentralized MFNC Systems

We list prospective malicious attacks towards decentralized MFNC systems in this section. We investigate five different types of attacks, i.e., disorder attack, repulsion attack, isolation attack, frog-boiling attack I, and frog-boiling attack II. The first three are conventional attacks classified based on their objectives, and the two frog-boiling attacks are emerging advanced attacks with more complicated attacking methods and more threatening effects. These attacks have been identified in the context of traditional Euclidean-based NC (ENC) systems and shown to be dangerous to ENC systems [7], [18]. We now introduce these attacks in MFNC systems.

Disorder attack: In this type of attack, malicious nodes try to cause a disorder for the entire NC system. A malicious node can inject fake information into the system by announcing its outgoing vector and incoming vector with randomly generated values. Also, it can add a random delay for the round-trip time (RTT) probing packets. As a result, the overall prediction accuracy will be decreased and the NC system will become harder to converge.

Repulsion attack: The objective of this type of attack is to convince legitimate nodes that they are far away from a victim node. In topology-aware distributed applications, they will naturally avoid to select the victim to communicate with. Recall a node's distance to a vicim node is the dot product of the node's outgoing vector and the victim's incoming vector. In this attack, malicious nodes mislead the victim node to derive its incoming vector $\vec{Y}$ to become a vector $\vec{Y}_{target}$ such that other legitimate nodes would believe that their distances to the victim would be far. A straightforward way is to assign large values to the elements of $\vec{Y}_{target}$. However, this can easily be detected by checking against extreme values. In our attack model, the following equation will be used:

$$\vec{Y}_{target} = \alpha * \vec{R}_1 * \vec{Y}_{max} + \beta * \vec{Y}_{max} \quad (7)$$

In Eq. 7, $\vec{R}_1$ is a randomly generated d-dimensional vector with every element randomly chosen between (0,1]. $\vec{Y}_{max}$ is a multiplier. Every element of $\vec{Y}_{max}$ is configured as the maximum value of the elements of $\vec{Y}$ before the attack is launched. We ensure $\alpha + \beta = 1$ so that the element values of $\vec{Y}_{target}$ will not be too large, and it cannot be discovered by extreme value-based detection. To mislead the victim's outgoing vector towards $\vec{Y}_{target}$, a malicious node will falsify its coordinates and delay RTT probes. In our attack model, we set $\vec{X}_{mal} = \gamma * \vec{R}_2 * \vec{X}_{max}$, and accordingly the RTT will be delayed as $d_{delay} = \vec{X}_{mal} \cdot \vec{Y}_{target}$. Similar to $\vec{R}_1$, $\vec{R}_2$ is also a random-generated d-dimensional vector, and γ can be randomly chosen between (0, 1].

Isolation attack: The objective of this type of attack is to convince a victim node that it is located in a particular area of the network and is isolated from legitimate nodes, thus more easily connecting with malicious nodes. The outgoing vector of the victim becomes compromised as $\vec{X}_{target} = \vec{C}$ ($\vec{C}$ is a d-dimensional vector), so that the victim believes it is faraway from regular nodes but near malicious nodes (recall the distance from the victim to a given node is the dot

produce of the victim's outgoing vector and the given node's incoming vector), and tries to connect to the latter when it needs neighbors to connect to.

Frog-boiling attack I: As introduced in [7], in this type of attack, malicious nodes gradually alter their own coordinates with only a small, hardly noticeable change at every round of updating coordinates. Such attack is very hard to detect. However, after a number of rounds, the change will become large enough for the attackers to mislead legitimate nodes. Specifically, at each round for updating coordinates, a malicious node can falsify both of its coordinates with a fixed small amount ($\vec{\delta}$) using the following form: $\vec{X}_{mal} = \vec{X}_{ori} + \vec{\delta} * t$ and $\vec{Y}_{mal} = \vec{Y}_{ori} + \vec{\delta} * t$, while $\vec{X}_{ori}$ and $\vec{Y}_{ori}$ are its legitimate coordinates before the attack begins, and t is the current number of rounds. This type of frog-boiling attack requires a pair of pre-selected target coordinates ($\vec{X}_{target}$ and $\vec{Y}_{target}$). The attacker gradually alters the coordinates to approach this pair of coordinates. We can see that between two continuous rounds, a node's coordinates just change slightly. However, once t becomes a large value, $\vec{X}_{mal}$ and $\vec{Y}_{mal}$ will be significantly different from $\vec{X}_{ori}$ and $\vec{Y}_{ori}$.

Frog-boiling attack II: We also introduce another type of frog-boiling attack for further investigation of NCShield. Different from frog-boiling attack I, this type of frog-boiling attack requires not only a pair of pre-selected target coordinates ($\vec{X}_{target}$ and $\vec{Y}_{target}$), but also a pre-defined RTT delay d_{target}. It then gradually alters the coordinates and RTTs to approach these pre-set values over totally T rounds. Within each round, it falsifies the coordinates as: $\vec{X}_{mal} = \vec{X}_{ori} + (\vec{X}_{target} - \vec{X}_{ori})/T * t$ and $\vec{Y}_{mal} = \vec{Y}_{ori} + (\vec{Y}_{target} - \vec{Y}_{ori})/T * t$. The RTT is delayed as $d_{delay} = d_{ori} + (d_{target} - d_{ori})/T * t$.

IV. Design of NCShield

We design a *score-and-vote* based approach called **NCShield** to protect decentralized MFNC systems from being attacked. We let the nodes in an MFNC system help each other check whether some reference nodes (neighbors) provide untrustworthy coordinates and RTTs. For each node, besides choosing a list of reference nodes (NList), it randomly picks a list of verification nodes (VList) to determine whether its neighbors are malicious or not. The VList nodes can vote together to provide an aggregate opinion.

There are four challenges in our system design: (1) We need a robust node sampling protocol to ensure all nodes in NList and VList are selected randomly, even in a large dynamic system with a number of malicious nodes. (2) We need to propose a fair voting policy to discover malicious nodes, and we need to seamlessly integrate the voting results into the NC calculation. (3) Besides handling typical disorder, repulsion, and isolation attacks, our system should be able to defense the new emerging frog-boiling attacks. (4) The system should be lightweight and scalable. As NC systems aim to play as a building block in large-scale distributed systems, we require our system to introduce a moderate overhead.

As shown in Fig. 1, for a new node to join the NCShield system, it needs four steps to obtain its coordinates. In the first step, it needs to discover existing nodes in the system without any bias (Section IV-A). Then in the second step, it measures its distances to nodes in NList and retrieves their coordinates. For the third step, on one hand, its VList nodes will do independent grading and voting to defend against conventional attacks such as disorder attack, repulsion attack, or isolation attack (Section IV-B). On the other hand, it will use an anti-frog-boiling module (Section IV-C) to defend against frog-boiling attacks. In the last step, the node determines the reliable nodes in its NList, and uses their information to calculate its own coordinates.

We describe the key components of NCShield in this section. In Section IV-A, we discuss the unbiased peer discovery. In Section IV-B, we introduce the voting-based malicious node detection. In Section IV-C, we describe the anti-frog-boiling module. Finally, in Section IV-D, we demonstrate how we integrate NCShield into MFNC systems.

A. Peer Discovery

Peer discovery is a fundamental building block in NCShield. In an MFNC system with NCShield, every node needs a list of neighbors (NList) for the NC calculation, and a list of verification nodes (VList) to judge the trustworthiness of the neighbors. We believe that attackers would have a strong incentive to be chosen as either a neighbor or a verification node, in order to mislead the NC calculation and verification. Malicious nodes might collude together to increase their chance to be included in a legitimate node's NList or VList.

In NCShield, we employ the Brahms protocol [4] for peer discovery. Brahms is a gossip-based protocol to achieve an unbiased sampling of peers in distributed systems, and it can overcome Byzantine attacks by a linear portion of the system. When a node H_{new} joins the system, in addition to maintaining its NList and VList, it also launches a sampler required by Brahms, and implements a balance algorithm to control the contribution of pushes and pulls in gossip sessions, thus achieving an unbiased sampling of both NList and VList members. More specifically, it will operate in the following steps:

Contacting the rendezvous point (RP): Node H_{new} registers itself to the RP, and obtains a list of existing nodes randomly selected by RP.

Contacting candidates: Node H_{new} sends probe messages to nodes on the candidate list. When node H_{new} receives a confirmation message from a node, it will add this node to its NList or VList. H_{new} repeats this operation until both lists reach the pre-set scale. Note that due to node churns or insufficient candidates, NList and VList might be short of enough nodes.

Secure gossip process: In order to discover more nodes in the system, H_{new} can start a Brahms process. Brahms is a gossip protocol, which allows online nodes to exchange their knowledge. The stream of new nodes discovered through the above exchanges is balanced and sampled with min-wise independent permutations [5], so that node H_{new} can update its sample list and maintain the unbiased property of its sample list. Finally a new unbiased sample of candidates can be obtained.

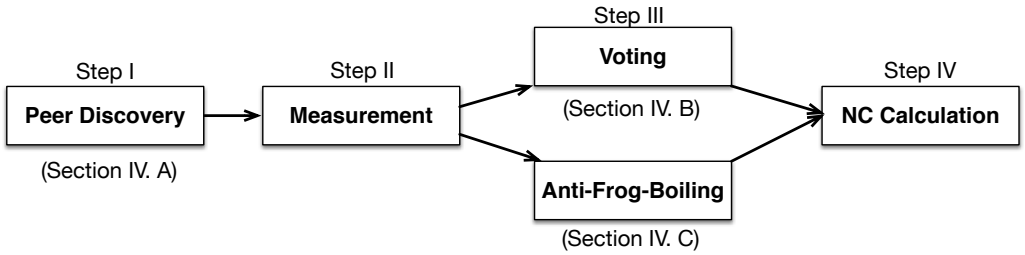


Fig. 1: Basic Structure of NCShield

With the help of Brahms, malicious nodes will not be able to intensively advertise themselves to legitimate nodes and increase the percentage of malicious nodes in every legitimate node's NList and VList.

B. Voting-Based Malicious Node Detection

As we have mentioned in Section II, every node refers to the reference nodes in its NList to calculate and update its own coordinates periodically. In this subsection, we demonstrate how to leverage VList nodes to find out malicious nodes in NList, and eliminate their impact in the NC calculation.

Once a node H has obtained its NList and VList, it starts to calculate its incoming vector and outgoing vector. Let m and u represent the number of neighbors assigned to node H and the number of verification nodes, respectively. For every node H_n in the NList, node H asks it for the latest coordinates, and conducts measurement to obtain the RTT between itself and H_n. To ensure whether H_n is trustworthy, the node H uses a two-step verification procedure as follows.

Independent grading: In this step, every VList member of H retrieves coordinates of node H_n and conducts the RTT measurement. For a node H_v in VList, based on the suspicious outgoing and incoming vectors of node H_n and the RTTs between itself and node H_n. H_v calculates two scores: $s^{in}_{H_vH_n}$ and $s^{out}_{H_vH_n}$, as in Eq. 8 and Eq. 9.

$$s^{in}_{H_vH_n} = \frac{|D^E(H_v, H_n) - D(H_v, H_n)|}{\min(D^E(H_v, H_n), D(H_v, H_n))} \tag{8}$$

$$s^{out}_{H_vH_n} = \frac{|D^E(H_n, H_v) - D(H_n, H_v)|}{\min(D^E(H_n, H_v), D(H_n, H_v))} \tag{9}$$

A smaller score value indicates a higher level of trust. To determine whether a score value is acceptable, we introduce a pre-defined score threshold (ST). If the score value is smaller than ST, we find it as "trustworthy". Otherwise, we find it as "malicious".

H collects the reports from all nodes in its VList, then it can have an aggregate opinion for the incoming and outgoing vectors of H_n as follows.

$$a^{in}_{H_n} = \sum_{H_v \in VList} (s^{in}_{H_vH_n} < ST) \tag{10}$$

$$a^{out}_{H_n} = \sum_{H_v \in VList} (s^{out}_{H_vH_n} < ST) \tag{11}$$

Voting: Node H integrates the returned information and calculates v_{H_n} (Eq. 12) to decide whether adopting or ignoring node H_n's coordinates according to a pre-defined vote threshold (VT).

$$v_{H_n} = (a^{in}_{H_n} \geq VT)\&\&(a^{out}_{H_n} \geq VT) \tag{12}$$

Finally, after collecting the voting information of all neighbors, node H calculates the total number of reliable neighbors r (Eq. 13) and decides whether to start the NC update process, according to a pre-defined number of reliable neighbor threshold (RT).

$$r = \sum_{H_n \in NList} v_{H_n} \tag{13}$$

Besides the process described above, several details are noteworthy:

1) Before calculating v_{H_n} for neighbor node H_n, a comparison should be made in case that H_n sends different coordinates of itself to node H and the VList members of H. If we can safely conclude that H_n is providing inconsistent information to different nodes, we can directly mark it as a malicious node.
2) To obtain input for the independent grading, every node H_v in VList needs to ask H_n for its coordinates and measures the RTT between itself and H_n. This procedure should have no difference from what node H does. Otherwise, a malicious node H_n can act legitimately if it is aware that the request is from a node in VList, including sending reliable coordinates to H_v and not delaying the RTT measurement requests.
3) Since each node maintains two vectors and at least one of them needs to be verified, various voting strategies can be applied. In this paper, we apply a relatively stringent strategy such that a positive vote is made only if both $a^{in}_{H_n}$ and $a^{out}_{H_n}$ are larger than or equal to the threshold (VT).
4) Malicious nodes may also appear in VList. Brahms guarantees an unbiased percentage of malicious nodes in both VList and NList, (*i.e.*, the same ratio of malicious nodes as that in the whole network). In our simulation, these nodes will vote randomly.
5) A specific module is integrated into the secure NC calculation to prevent frog-boiling attacks (or other more advanced attacks). We present the details of this module in Section IV-C.

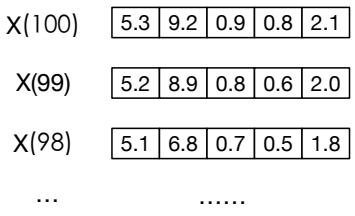


Fig. 2: Detection of Frog-Boiling Attacks

C. Handling More Complicated Attacks: an Anti-Frog-Boiling Module

For more complicated attacks such as frog-boiling attacks, voting-based solutions cannot be accurate and effective enough. To discover nodes in the NList conducting frog-boiling attacks, we introduce a specified anti-frog-boiling module. For a node H_v, this module will check against every node in its NList, and find out the nodes performing frog-boiling attacks.

Since frog-boiling attackers manipulates their own coordinates in the same direction round after round gradually, we use this feature for the detection. A pair of $\delta(\cdot)$ vectors is introduced. For a node H_n, within round t, $\delta(\cdot)$ can be calculated as in Eq. 14 and 15.

$$\vec{\delta}^X(t) = sign(\vec{X}_{H_n}(t) - \vec{X}_{H_n}(t-1)) \quad (14)$$

$$\vec{\delta}^Y(t) = sign(\vec{Y}_{H_n}(t) - \vec{Y}_{H_n}(t-1)) \quad (15)$$

Here "sign" means the signum function [1]. For example, in Fig. 2, we can see that $\vec{\delta}^X(100)$ and $\vec{\delta}^X(99)$ are identical, since every element in $\vec{X}$ grows gradually from time to time. However, if we change the third element in $\vec{X}(99)$ from 0.8 to 1.1, then we can see that $\vec{\delta}^X(100)$ and $\vec{\delta}^X(99)$ are not equal. In other words, not all the elements in vector X move in the same direction round after round.

Based on these $\delta(\cdot)$ vectors, we calculate a "frog factor" (f) for this neighbor:

$$f = (\vec{\delta}^X(t) == \vec{\delta}^X(t-1))\&\&(\vec{\delta}^Y(t) == \vec{\delta}^Y(t-1)) \quad (16)$$

The value of f indicates whether a frog-boiling attack is going on. If f equals to 0, we apply the aforementioned grading and voting steps. In contrast, if f equals to 1, NCShield will find that H_n is performing a frog-boiling attack, and will ignore its information in updating H's coordinates.

The anti-frog-boiling module is highly extensible. We can upgrade this module to prevent from more complicated malicious attacks in the future.

D. Integrating NCShield with MFNC Systems

NCShield is compatible with different MFNC systems such as Phoenix and DMFSGD. In this subsection, we use a piece of pseudocode shown in Fig. 3 to demonstrate the overall workflow of NCShield. In the first three lines, we define ST, VT and RT. For a new host, it will contact the RP to obtain a list of existing hosts (line 4). Based on this list, it will connect to NList and VList members (lines 5-7). Afterwards, the host will conduct measurements and update its coordinates periodically (lines 8-24). As in any existing NC system without security considerations, the host communicates with nodes in its NList to get the RTT information, and these latest coordinates. Particularly, line 10 denotes the peer discovery in NCShield, line 11 shows how the new host delivers necessary information to VList members. In lines 12-17, we demonstrate how VList members work, including frog-boiling detection (line 14), independent grading (line 15), and voting (line 16). In lines 18-22, we can see how the new host finally get feedbacks from VList members, and updates its coordinates accordingly.

[1]For any $x > 0$, $sign(x) = 1$; for any $x = 0$, $sign(x) = 0$; for any $x < 0$, $sign(x) = -1$.

V. Evaluation Methodology and Metrics

To show the usefulness of NCShield in various respects, we conduct a series of simulation to evaluate NCShield in both Phoenix and DMFSGD systems. In this section, we describe the evaluation methodology, metrics, data sets, scenarios, and system parameters.

A. Communication Overhead Analysis

We aim to have a lightweight NC system. Therefore, the communication overhead is a critical issue. We analysis the communication overhead of NCShield, and compare it with the DHT-based Veracity system in Section VI-A. We have found that NCShield is more cost-effective, and it generates less additional traffic.

B. Evaluation with Aggregate Data Sets

Three representative aggregate data sets are used in this evaluation in Section VI-B. The first data set is the AMP data set [32], which includes the RTTs among 110 Internet hosts. The hosts are mainly at NSF supported HPC sites, with about 10% outside the US. The AMP data set has been used in [32], [48]. The second data set is the PlanetLab data set [52], which includes the RTTs among 335 PlanetLab hosts all over the world, collected during March-April, 2010. The PlanetLab data set has been used in [14], [32], [48], [52]. The third data set is the King data set which includes the RTTs among 1740 Internet DNS servers [14]. The King data set has been used in [14], [42], [43], [48]. These data sets can present three different Internet delay spaces [47].

Since an aggregate data set only provides **one** "snapshot" of a pairwise RTT matrix, frog-boiling attacks and the anti-frog-boiling module do not function in this evaluation. Therefore we evaluate NCShield against disorder, repulsion and isolation attacks in Section VI-B. As described in Section III, in disorder attacks, malicious nodes would send false coordinates, including both outgoing vectors and incoming vectors. We use E_{max} to represent the value of the largest element of these vectors before launching the attack. For malicious nodes, each element of these two vectors is randomly generated within the range of [0, E_{max}]. In repulsion attacks, each element

define ST SCORE_THRESHOLD
define VT VOTE_THRESHOLD
define RT RELIABLE_NEIGHBORS_THRESHOLD
$Get\_Initial\_Host\_Candidates(RP)$
$Generate\_NList\_and\_VList()$
$Connect\_to\_NList\_Members()$
$Connect\_to\_VList\_Members()$
while $forever$ **do**
  $Get(d(\cdot), \boldsymbol{X}, \boldsymbol{Y})$ {Probing RTT and requesting incoming and outgoing coordinates from NList}
  $Check\_and\_Renew\_Candidates\_with\_Brahms()$ {Secure gossip}
  $Deliver\_to\_VList(\boldsymbol{X}, \boldsymbol{Y}, neighbor\_addrs)$ {Providing information for VList members}
  for all $Members \in VList$ **do**
    $d_{v\_to\_n}(\cdot) = Measure\_to\_Neighbor(neighbor\_addrs)$ {Probing RTT from verification nodes to neighbor nodes}
    $\delta(\cdot) = Cal\_\delta(current(\boldsymbol{X}, \boldsymbol{Y}), history(\boldsymbol{X}, \boldsymbol{Y}))$ {Anti-frog-boiling}
    $s(\cdot) = Cal\_Score(d_{v\_to\_n}(\cdot), \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{X_v}, \boldsymbol{Y_v})$ {Grading}
    $v(\cdot) = Cal\_Vote(s(\cdot), ST, \delta(\cdot))$ {Voting}
  end for
  $Deliver\_to\_Host(v(\cdot))$ {Node H collects votes}
  $r = Parse\_Vote(v(\cdot), VT)$ {Node H makes decisions}
  if $r \geq RT$ **then**
    MFNC_Update_Coordinate() {NC calculation}
  end if
  Wait(NC_UPDATE_INTERVAL)
end while

Fig. 3: Pseudocode of MFNC Systems with NCShield

in the incoming and outgoing vectors of malicious nodes are randomized within the range of [0, $\frac{1}{2}E_{max}$]. Thus, the RTT probes are delayed according to the randomly generated target vectors. In isolation attacks, the coordinates of malicious nodes are randomized the same way as in repulsion attack. However, the attack targets are outgoing vectors of victims, which are aimed to be set to a vector of pre-defined maximum values. The RTT probes are also delayed correspondingly.

Futhermore, we assume that the attackers are "smart" enough, and they can be injected into the NC system successfully. We foresee that they will act as legitimate nodes in the very beginning, and launch malicious attacks later. Simulations for each scenario are repeated 5 times and the average results are obtained.

C. Evaluation with a Dynamic Data Set

The Internet distances are time-varying. However, existing security schemes for ENC systems do not consider this variation at all. To the best of our knowledge, whether existing NC security schemes are robust to the Internet distance variation remains an unknown problem. We believe that a practical security scheme should be able to distinguish between ordinary distance variation and maliciously generated distance provided by the attackers.

To evaluate the robustness of NCShield in dynamic environments, we introduce the "K200-allpairs-1h" dynamic data set [30]. This data set contains 200 nodes and the data collection lasts 44 hours using King method. We have obtained 99 continuous snapshots of all pairwise RTTs. NCShield is evaluated not only against disorder, repulsion and isolation attacks, but also against the two types of frog-boiling attack. In frog-boiling attack I, malicious nodes alter their coordinates by increasing 0.1 in each dimension, within each round. The RTT probing packets are also delayed by the value which is evenly assigned to the 98 rounds, generated from uniform randomness in [100..1000] ms. Thus the alteration is small enough to make detections difficult. In frog-boiling attack II, malicious nodes not only change their coordinates gradually, but also delay RTT probing packets in a smooth way.

We run on the first one of 99 snapshot matrices to achieve an acceptable convergence of coordinates. Then from the second snapshot matrix, the malicious nodes start their attacks. The results are presented in Section VII.

D. Evaluation in an Online Game Scenario

Besides using typical RE metric for evaluating the prediction accuracy, we also evaluate NCShield in a practical scenario, i.e., a popular online game scenario, in Section VIII. As introduced in [12], [31], identifying all end-to-end links with shorter latencies than a pre-defined threshold is critical for various real-time interaction games, and NC systems are a scalable solution to get a prediction of RTTs of all links. According to the requirement of first-person shooter games, we set this threshold as 100ms [12].

A link is defined as a good (resp. bad) link when its measured RTT is below (resp. above) the pre-defined threshold. A true positive (TP) indicates that a good link is correctly predicted as "good", while a false positive (FP) shows that a bad link is wrongly predicted as "good". Likewise, a true negative (TN) tells that a bad link is correctly predicted

TABLE II: NPRE of Repulsion Attacks

NC	Data	Defense	Percentage of Malicious Nodes			
			0%	10%	30%	50%
Phoenix	AMP	OFF	0.284	0.505	1.059	2.706
		ON	0.285	0.289	0.307	0.394
	PL	OFF	0.444	0.701	1.410	2.259
		ON	0.492	0.558	0.674	0.752
	King	OFF	0.450	1.170	1.558	4.029
		ON	0.456	0.548	0.590	0.619
DMFSGD	AMP	OFF	0.183	0.556	1.766	1.992
		ON	0.183	0.194	0.293	0.250
	PL	OFF	0.559	0.866	2.992	3.796
		ON	0.559	0.634	1.111	1.009
	King	OFF	0.782	1.525	2.491	2.583
		ON	0.782	0.776	0.965	0.854

TABLE III: NPRE of Isolation Attacks

NC	Data	Defense	Percentage of Malicious Nodes			
			0%	10%	30%	50%
Phoenix	AMP	OFF	0.285	0.558	1.157	3.412
		ON	0.285	0.301	0.305	0.373
	PL	OFF	0.445	0.689	1.328	2.567
		ON	0.469	0.529	0.653	0.676
	King	OFF	0.444	0.988	1.582	5.001
		ON	0.463	0.531	0.557	0.586
DMFSGD	AMP	OFF	0.145	0.416	0.582	1.348
		ON	0.145	0.160	0.308	0.319
	PL	OFF	0.611	0.685	1.616	1.731
		ON	0.611	0.643	1.061	1.169
	King	OFF	0.781	1.196	3.644	3.225
		ON	0.781	0.801	0.911	0.998

as "bad", while a false negative (FN) points that a good link is wrongly predicted as "bad". False positive rate (FPR) and false negative rate (FNR), the metrics we adopt in this evaluation, are defined by $FPR = FP/(FP + TN)$ and $FNR = FN/(TP + FN)$, respectively.

We conduct this simulation with AMP, PL335 and King data sets against disorder attacks, and with the dynamic data set against frog-boiling attacks. Half of the nodes, i.e. 55 nodes, 167 nodes, 870 nodes and 100 nodes in each data set, respectively, are chosen to be participants of online game, and these nodes are legitimate nodes. We vary the percentage of malicious nodes from 10% to 50%, with an interval of 10%.

E. System Parameters

In both Phoenix and DMFSGD, each node is assigned 32 [9], [27] neighbors and 7 [38] VList members, which are selected randomly. The coordinate dimension is set to 10 [9], [27]. Using default values in [9], [27], we set update rounds of Phoenix and DMFSGD to 30 and 50, respectively. For Phoenix, the constant C is set to 10 [9], ST (SCORE_THRESHOLD defined in Algorithm § IV-D) is 1.0, VT (VOTE_THRESHOLD) is 6, and RT (RELIABLE_NEIGHBORS_THRESHOLD) is 10. While in DMFSGD, the regulation coefficient λ is set to 1, learning rate η is 1e-2 [27], ST is 0.4, VT is 4, and NT is 10. Later we will see that the parameters of NCShield are not sensitive to the delay space. The malicious group size is set from 10% to 50%, with an interval of 10%.

VI. Results with Aggregate Data Sets

In this section, we evaluate NCShield with aggregate data sets. We first analyze the communication overhead of NCShield using a gossip algorithm, and compare it with Veracity using DHT. We then present the results of our simulation study in Phoenix and DMFSGD using three aggregate data sets.

A. Communication Overhead Analysis

As NC systems are aiming at reducing traffic overhead, it is necessary to guarantee that a defense approach does not introduce viable overhead. According to the pseudocode in Section IV-D, table I summarizes the coordinate verification steps of Veracity using a DHT and NCShield using a gossip protocol. The communication overhead of each step is also shown, counted in number of messages. In the table, m represents the number of neighbors, u stands for the number of VList members of each node, and N is the number of nodes in the NC system. According to the table, NCShield can significantly save the communication costs from Veracity. According to the parameter configuration above, $m = 32$ and $u = 7$, with a node scale of 1024, the communication overhead in one round of coordinate verification process of all nodes are 2674688 messages and 997376 messages in Veracity and NCShield, respectively. Compared with DHT based solution, our gossip-based approach can save 62.7% traffic for coordinate verification operations.

B. Results with Aggregate Data Sets

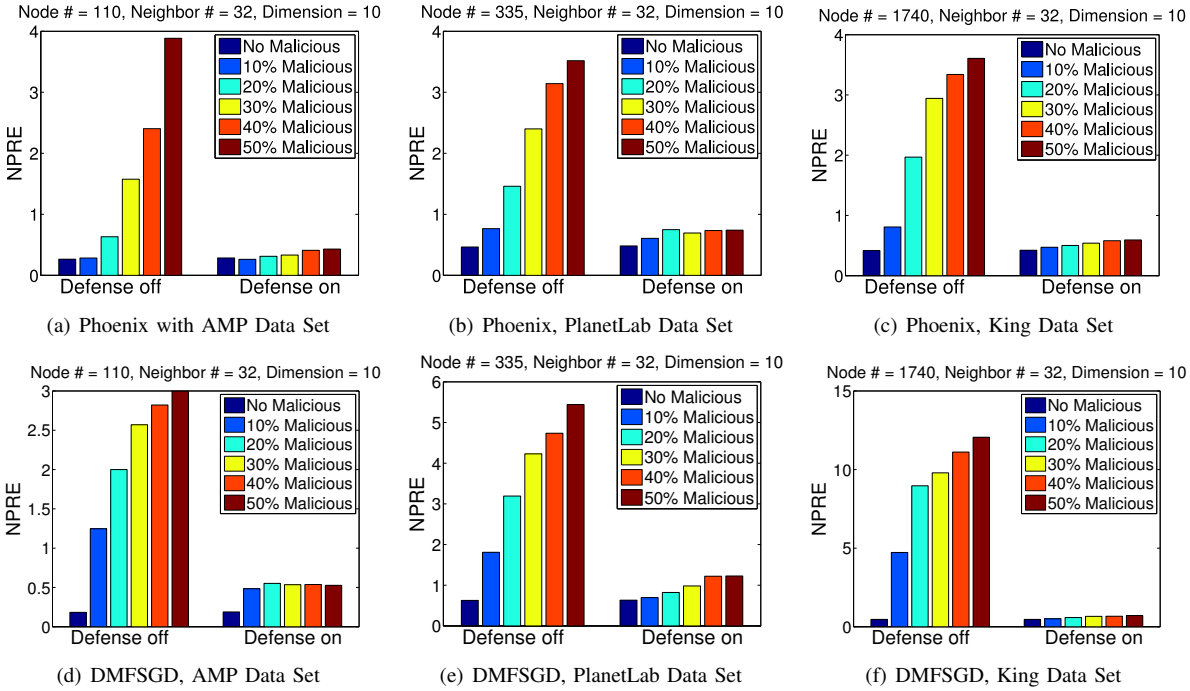
In this subsection, we use the parameter settings of Phoenix, and DMFSGD defined in § V-E. In this subsection, "defense on" means that we have applied NCShield to secure an MFNC system, and "defense off" means that we have not applied NCShield.

We first launch disorder attacks, and Fig. 4 shows the results. Particularly, we examine both systems by both enabling and disabling NCShield. The NPRE results are calculated with all the participant nodes *EXCEPT* malicious nodes. The figure indicates that NCShield can largely reduce the negative effect of disorder attacks. By using NCShield, when we increase the percentage of the malicious nodes, the NPRE will not increase remarkablely. Differently, if we do not apply NCShield, we can see a rapid increase of NPRE. For example, using the AMP data set to simulate Phoenix (Fig. 4(a)), we can see that the NPRE is 0.223 when there is no attacker in the system. If we do not apply NCShield, when 30% malicious nodes exist, the NPRE increases to 0.998 which is a significant jump. In contrast, when we apply NCShield in Phoenix, the NPRE drops to 0.315. Therefore, we can see that NCShield can prevent Phoenix from disorder attacks. Moreover, in Fig. 4(b)-4(f), we can see that for both Phoenix and DMFSGD systems, NCShield performs very well using different data sets.

The results of launching repulsion attacks are shown in Table II. The NPRE results are calculated with the outgoing vectors of all nodes *EXCEPT* malicious nodes and the incoming vectors of victim nodes. From the table we can see that NCShield can significantly protect legitimate nodes from repulsion attacks. As we can identify and terminate repulsion

TABLE I: Comparison of Communication Overhead between Veracity and NCShield

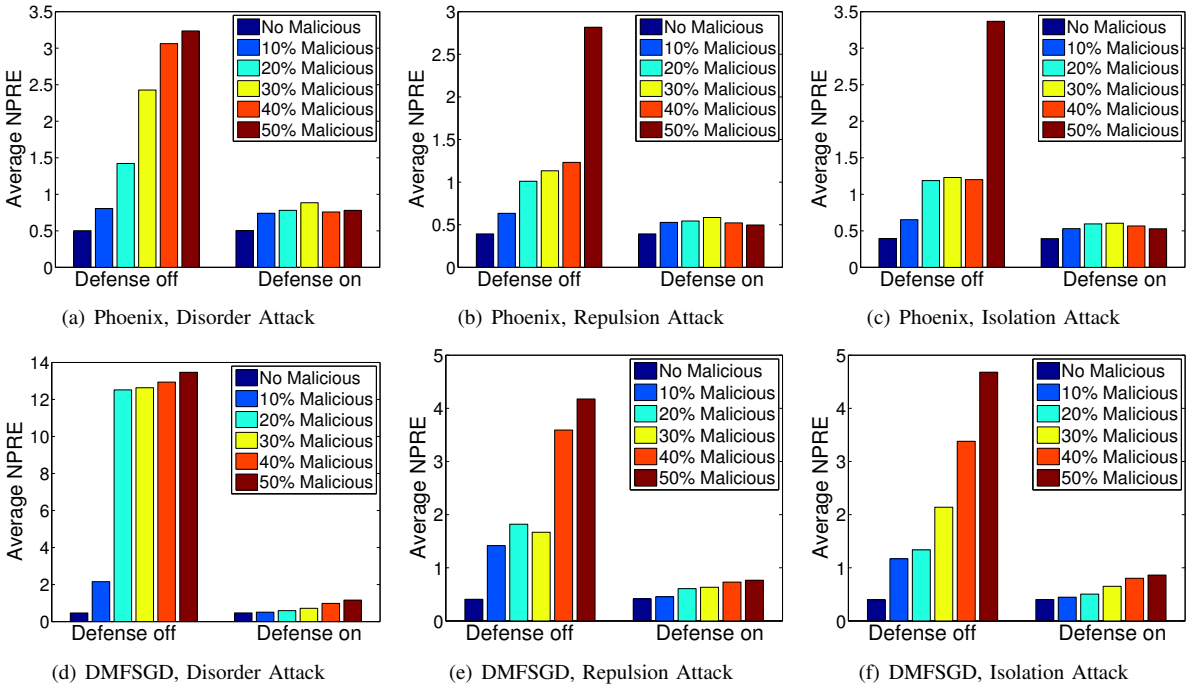
Veracity using DHT	Overhead	NCShield using Gossip	Overhead
Pub. contact VSets	$Nu(log_2 N)$	H contacts NList	$Nm * 2$
VSets ping pub.	$Nu * 2$	H contacts VList	Nu
Invest. contacts pub.	$Nm * 2$	VList contact NList	$Nmu * 2$
Invest. contacts VSets	$Nmu(log_2 N)$	VList ping NList	$Nmu * 2$
VSets return results	Nmu	VList return results	Nu



(a) Phoenix with AMP Data Set (b) Phoenix, PlanetLab Data Set (c) Phoenix, King Data Set

(d) DMFSGD, AMP Data Set (e) DMFSGD, PlanetLab Data Set (f) DMFSGD, King Data Set

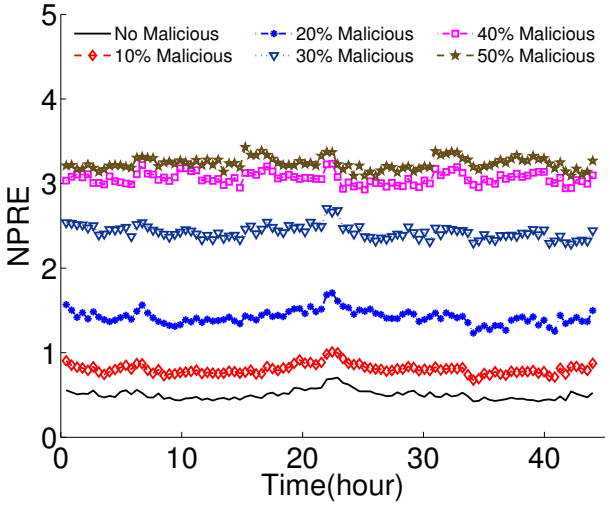
Fig. 4: NCShield against Disorder Attacks with Aggregate Data Sets



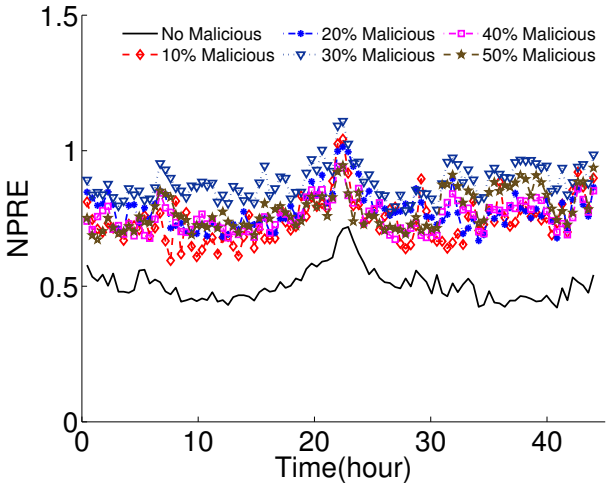
(a) Phoenix, Disorder Attack (b) Phoenix, Repulsion Attack (c) Phoenix, Isolation Attack

(d) DMFSGD, Disorder Attack (e) DMFSGD, Repulsion Attack (f) DMFSGD, Isolation Attack

Fig. 6: Average NPRE Results with the Dynamic Data Set



(a) Phoenix w/o Defense



(b) Phoenix with Defense

Fig. 5: NCShield against Disorder Attacks with the Dynamic Data Set ("K200-allpairs-1h")

attacks, other nodes will not believe that the victims are far away from them.

Similarly, the results of isolation attacks are shown in Table III. The NPRE results are calculated with the outgoing vectors of victim nodes and the incoming vectors of all nodes *EXCEPT* malicious nodes. Our results have also shown that NCShield can remedy isolation attacks well, and accordingly the victims will no longer be pushed to somewhere near the malicious nodes in the delay space.

VII. Results with the Dynamic Data Set

In this section, we examine NCShield with a dynamic data set. We use the same parameter settings of Phoenix and DMFSGD defined in Section V-E. For instance, we present the evaluation against the conventional disorder, repulsion and isolation attacks as those in Section VI-B.

Fig. 5 shows the NPRE variations in Phoenix under disorder attack with the "K200-allpairs-1h" data set. In Fig. 5(a), without defense, the RE increases significantly when malicious nodes increase. As Internet distances varying from time to time, the performance degrades when the system is under disorder attack. While in Fig. 5(b), with NCShield, the NPRE values are much smaller than the corresponding ones in Fig. 5(a). Therefore, NCShield achieves a very good performance in the dynamic data set as well. The increase of RE is obviously mitigated, which indicates NCShield works well in this scenario.

Fig. 6 shows the average NPRE results both in Phoenix and DMFSGD systems under all first three attacks. From the figure we can see that NCShield can remedy the three attacks significantly for both systems. For example, in Phoenix with 30% malicious nodes, NCShield can help decrease the average NPRE from 1.133 to 0.585 under repulsion attack. In DMFSGD, when 10% malicious node conducting disorder attack, NCShield decreases the average NPRE from 1.661 to 0.904.

Furthermore, we present the evaluation against frog-boiling attack I and II, which aim at disrupting the NC systems in the way of gradually falsifying the coordinates and delaying RTT probing packets.

Figs. 7(a), 7(b), 8(a) and 8(b) show the simulation results of NCShield against frog-boiling attack I. The NPRE values are calculated with the outgoing and incoming vectors of all nodes *EXCEPT* malicious nodes. Figs. 7(c), 7(d), 8(c) and 8(d) show the simulation results of NCShield against frog-boiling attack II. The NPRE values are calculated in the same way. The comparison of each two subfigures shows that NCShield can defend the systems against frog-boiling attacks significantly.

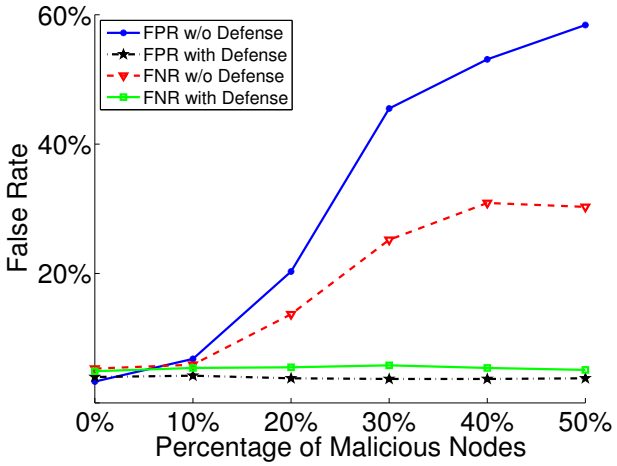
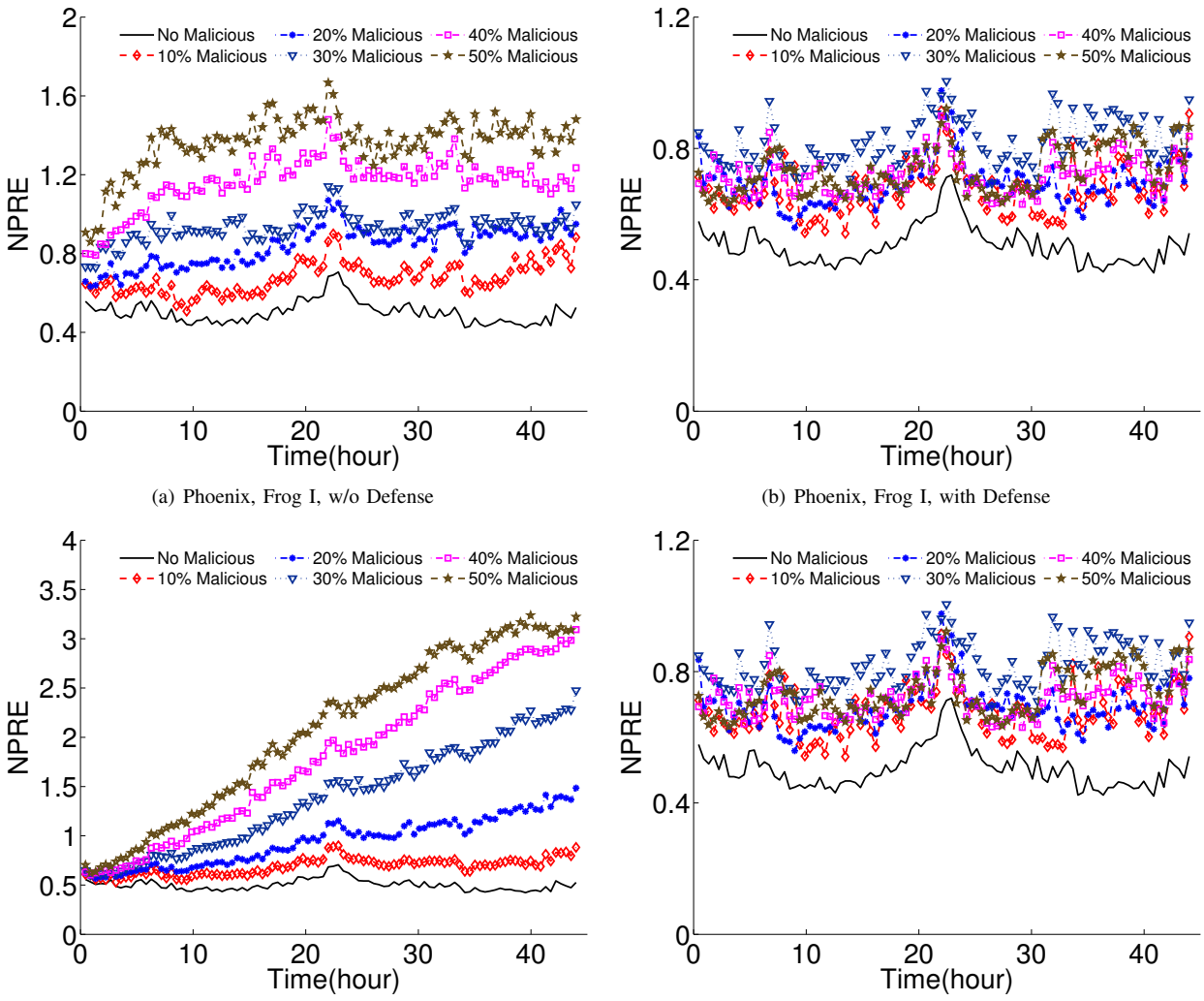


Fig. 9: Online Game Scenario, Disorder Attacks, Phoenix System (PL335 Data Set)

VIII. Online Game Scenario Evaluation

We use an online game scenario [2], [31] as a representative service to evaluate the usefulness of NCShield. To simulate Phoenix, we adopt the same parameter settings as previous sections. Here we only show the results of PL335 data set in Fig. 9, while the results of the other two aggregate data sets are similar. For the Phoenix NC system, when 30% malicious nodes are conducting the disorder attack, NCShield



(a) Phoenix, Frog I, w/o Defense

(b) Phoenix, Frog I, with Defense

(c) Phoenix, Frog II, w/o Defense

(d) Phoenix, Frog II, with Defense

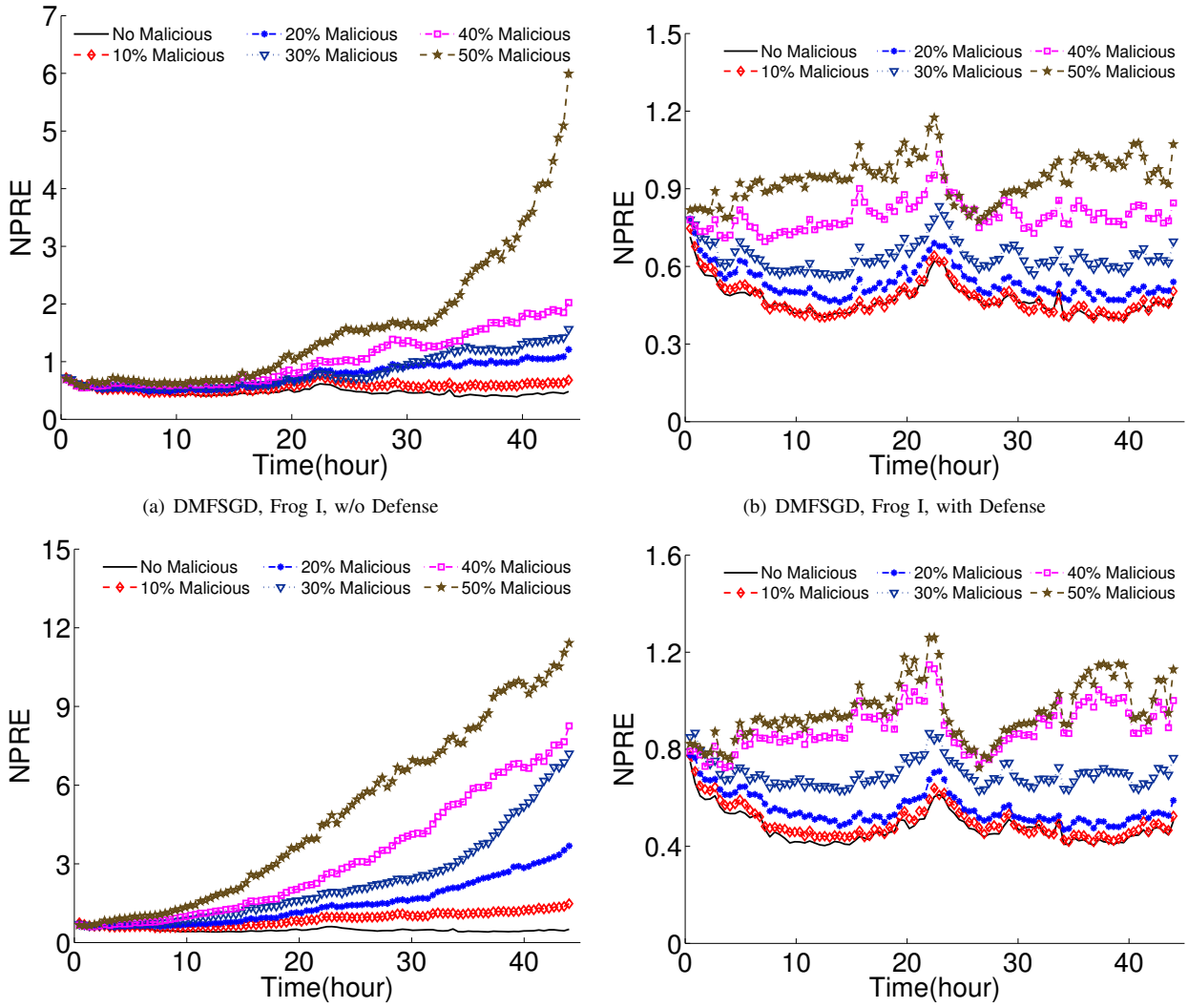
Fig. 7: NCShield against Frog-boiling Attacks in Phoenix

can reduce the FPR and FNR from 45.5% and 25.2% to 3.7% and 5.8%, respectively. Most of the negative impacts on link selection introduced by attackers are eliminated. This shows that in an online game scenario using Phoenix for link selection, NCShield is practical to prevent performance degradation caused by disorder attacks. We have also found similar results in simulating DMFSGD.

IX. Related Work

Researchers have proposed several approaches to defend decentralized ENC systems. All existing approaches are based on a common idea, i.e., using additional scalable measurement to judge whether a reference node is trustworthy. These approaches can be broadly classified into two categories: node behavior based approaches [17], [46], and trust and reputation based approaches [36], [38].

Kaafar *et al.* [17] propose a node behavior based approach for ENC system defense. Their intuition is that the dynamics of a node in a normal system can be modeled by a linear state space, and accordingly can be tracked by a Kalman filter. In their solution, a set of dedicated nodes are chosen as trusted surveyors to observe the dynamics of nodes and maintain the parameters of Kalman filter. Malicious behaviors of a node can be identified by its neighboring surveyors using the Kalman filter. The bottleneck of this approach is the large number of dedicated surveyor nodes, which produce a significant overhead when the scale of the system becomes large (e.g., 800-1000 surveyors are suggested for an NCS serving 10,000 nodes). To overcome this weakness, Zage *et al.* [46] propose a fully distributed approach without relying on a set of dedicated surveyors. It detects malicious nodes by observing inconsistent behaviors with their neighbors (temporal outlier) or the space of metrics (spatial outlier). This approach can get rid of a large number of dedicated surveyors. However, as the detection of temporal outliers depends on nodes' history information [38], it does not perform well with frequent node churns. Furthermore, the Kalman filer and outlier detection methods cannot defend the frog boiling attacks, as stated in [7], since the gradual coordinate alteration shows little impact within each update round.



(a) DMFSGD, Frog I, w/o Defense

(b) DMFSGD, Frog I, with Defense

(c) DMFSGD, Frog II, w/o Defense

(d) DMFSGD, Frog II, with Defense

Fig. 8: NCShield against Frog-boiling Attacks in DMFSGD

Both RVivaldi [36] and Veracity [38] use a trust and reputation system to secure ENC systems. RVivaldi employs two types of entities: a centralized Reputation Computation Agent (RCA) and surveyors, where surveyors monitor nodes and RCA performs centralized computation for every node's trust and reputation score. As a result, the centralized RCA becomes a single point of failure since it is responsible for computing the reputation scores for all nodes in the system. Veracity [38] does not need a centralized RCA. Instead, it employs two sets of nodes, VSet (voting node set) and RSet (reference node set), to help verify the process of updating node coordinates. It deploys a Distributed Hash Table (DHT) to help VSet and RSet construction as well as neighbor selection. However, as shown in § VI-A, this approach requires a significant amount of communication overhead in order to maintain its overlay routing structure. In addition, it requires additional security methods such as [6] to protect this additional infrastructure, which further adds extra overhead.

In addition, all of the four defense approaches above are only evaluated using *aggregate* data sets, in which the RTT between any two hosts in the data set is a *single* value, based on either the median [14] or the minimum of measured RTTs [47], [50] over a period of time (days or even weeks). However, the Internet distances are changing from time to time [30]. These approaches do not consider the distance variation in their investigation, and their performance on the real Internet remains unknown.

X. Conclusions

As decentralized MFNC systems can scale to millions of Internet users and are more accurate than Euclidean-based NC systems by getting rid of the limit of the triangle inequality, they have become a useful choice for helping large-scale Internet services, such as cloud-based services. Therefore, securing MFNC systems is critical for various Internet services and applications. As we have shown in this paper with the disorder, repulsion, isolation and two types of frog-boiling attacks towards Phoenix and DMFSGD systems, the security threats of decentralized MFNC systems can be very severe,

and even a small number of malicious nodes can deteriorate the accuracy of the entire MFNC system significantly.

We proposed a score and vote based approach, called NCShield, to secure MFNC systems. Besides preventing MFNC systems from classic attacks, NCShield can handle the new emerging frog-boiling attacks. Through extensive simulations using both aggregate data sets and a dynamic data set, NCShield is able to effectively defend MFNC systems. We have also investigated how to apply NCShield in Internet applications based on MFNC systems. We introduced an online game scenario, and studied the widely used link selection operation. Our results have shown that NCShield can significantly reduce the negative effects introduced by malicious attacks. To the best of our knowledge, our work is the first work to address the frog-boiling attacks in NC systems, and we are the first to consider the Internet distance variation in securing NC systems.

In the near future, we plan to deploy NCShield on the commodity Internet. We believe NCShield can be an integrated and useful part of a series of running Internet-based services, especially for cloud or web-based services. As an example, we aim to integrate NCShield into CloudGPS [15], which is a server selection scheme in cloud computing environments. CloudGPS uses Phoenix as the distance prediction module, and accordingly NCShield can play an important role in securing CloudGPS.

ACKNOWLEDGMENT

The authors are grateful to Xiaohan Zhao from the University of California, Santa Barbara for her helpful suggestions. We also thank Dr. Yongjun Liao from the University of Liege and Dr. Wei Du from the University of Göttingen for providing the simulator of DMFSGD and its instructions. This work has been partially sponsored by the State Lower Saxony's project "Simulation Science Center" as well as EU FP7 IRSES MobileCloud and FP7 ITN CleanSky projects.

REFERENCES

[1] I. Abraham and D. Malkhi. Compact Routing on Euclidian Metrics. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 141–149. ACM, 2004.

[2] S. Agarwal and J. R. Lorch. Matchmaking for online games and other latency-sensitive P2P systems. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 315–326. ACM, 2009.

[3] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.

[4] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: Byzantine resilient random membership sampling. *Computer Networks*, 53(13):2340–2359, 2009.

[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[6] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. *ACM SIGOPS Operating Systems Review*, 36(SI):299–314, 2002.

[7] E. Chan-Tin, V. Heorhiadi, N. Hopper, and Y. Kim. The frog-boiling attack: Limitations of secure network coordinate systems. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):27, 2011.

[8] Y. Chen, P. Sun, X. Fu, and T. Xu. Improving prediction accuracy of matrix factorization based network coordinate systems. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pages 1–8. IEEE, 2010.

[9] Y. Chen, X. Wang, C. Shi, E. K. Lua, X. Fu, B. Deng, and X. Li. Phoenix: A Weight-Based Network Coordinate System using Matrix Factorization. *Network and Service Management, IEEE Transactions on*, 8(4):334–347, 2011.

[10] Z. Chen, Y. Chen, C. Ding, B. Deng, and X. Li. Pomelo: accurate and decentralized shortest-path distance estimation in social graphs. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 406–407, 2011.

[11] Z. Chen, Y. Chen, Y. Zhu, C. Ding, B. Deng, and X. Li. Tarantula: Towards an Accurate Network Coordinate System by Handling Major Portion of TIVs. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE, 2011.

[12] M. Claypool and K. Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, 2006.

[13] M. Costa, M. Castro, R. Rowstron, and P. Key. PIC: Practical Internet coordinates for distance estimation. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 178–187. IEEE, 2004.

[14] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 15–26. ACM, 2004.

[15] C. Ding, Y. Chen, T. Xu, and X. Fu. Cloudgps: a scalable and isp-friendly server selection scheme in cloud computing environments. In *Quality of Service (IWQoS), 2012 IEEE 20th International Workshop on*, pages 1–9. IEEE, 2012.

[16] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *Computers, IEEE Transactions on*, 52(2):139–149, 2003.

[17] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti, and W. Dabbous. Securing internet coordinate embedding systems. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 61–72. ACM, 2007.

[18] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous. Real attacks on virtual networks: Vivaldi out of tune. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, pages 139–146. ACM, 2006.

[19] A. Klein, F. Ishikawa, and S. Honiden. Towards network-aware service composition in the cloud. In *Proceedings of the 21st international conference on World Wide Web*, pages 959–968. ACM, 2012.

[20] A. Klein, F. Ishikawa, and S. Honiden. SanGA: A Self-Adaptive Network-Aware Approach to Service Composition. *Services Computing, IEEE Transactions on*, 7(3):452–464, 2014.

[21] J. Ledlie, P. Gardner, and M. Seltzer. Network Coordinates in the Wild. In *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, pages 299–311, 2007.

[22] J. Ledlie, P. Pietzuch, and M. Seltzer. Stable and accurate network coordinates. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 74–74. IEEE, 2006.

[23] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[24] D. D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562. 2001.

[25] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha. On suitability of euclidean embedding of internet hosts. In *ACM SIGMETRICS Performance Evaluation Review*, volume 34, pages 157–168. ACM, 2006.

[26] X. Li, J. Wu, and S. Lu. QoS-Aware Service Selection in Geographically Distributed Clouds. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pages 1–5, 2013.

[27] Y. Liao, W. Du, P. Geurts, and G. Leduc. DMFSGD: A decentralized matrix factorization algorithm for network distance prediction. *IEEE/ACM Transactions on Networking*, 21(5):1511–1524, 2013.

[28] Y. Liao, P. Geurts, and G. Leduc. Network distance prediction based on decentralized matrix factorization. In *NETWORKING 2010*, pages 15–26. Springer, 2010.

[29] X. Lu, H. Wang, J. Wang, J. Xu, and D. Li. Internet-based Virtual Computing Environment: Beyond the data center as a computer. *Future Generation Comp. Syst.*, 29(1):309–322, 2013.

[30] C. Lumezanu, R. Baden, N. Spring, and B. Bhattacharjee. Triangle Inequality Variations in the Internet. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 177–183. ACM, 2009.

[31] J. Manweiler, S. Agarwal, M. Zhang, R. Roy Choudhury, and P. Bahl. Switchboard: a matchmaking system for multiplayer mobile games. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 71–84. ACM, 2011.

[32] Y. Mao, L. K. Saul, and J. M. Smith. Ides: An internet distance estimation service for large networks. *Selected Areas in Communications, IEEE Journal on*, 24(12):2273–2284, 2006.

[33] T. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 170–179. IEEE, 2002.

[34] T. E. Ng and H. Zhang. A Network Positioning System for the Internet. In *USENIX Annual Technical Conference, General Track*, pages 141–154, 2004.

[35] T. Ries, V. Fusenig, C. Vilbois, and T. Engel. Verification of Data Location in Cloud Networking. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 439–444, Dec 2011.

[36] D. Saucez, B. Donnet, and O. Bonaventure. A Reputation-Based Approach for Securing Vivaldi Embedding System. In *Dependable and Adaptable Networks and Services*, pages 78–85. Springer, 2007.

[37] P. Sharma, Z. Xu, S. Banerjee, and S.-J. Lee. Estimating network proximity and latency. *ACM SIGCOMM Computer Communication Review*, 36(3):39–50, 2006.

[38] M. Sherr, M. Blaze, and B. T. Loo. Veracity: practical secure network coordinates via vote-based agreements. In *Proc. of USENIX ATC*, pages 1–14, 2009.

[39] M. Sherr, A. Mao, W. R. Marczak, W. Zhou, B. T. Loo, and M. Blaze. A3: An Extensible Platform for Application-Aware Anonymity. In *Proc. 17th Network and Distributed System Security Symposium (NDSS)*, pages 1–20, 2010.

[40] M. Steiner and E. W. Biersack. Where is my peer? evaluation of the vivaldi network coordinate system in azureus. In *NETWORKING 2009*, pages 145–156. Springer, 2009.

[41] L. Tang and M. Crovella. Virtual Landmarks for the Internet. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, pages 143–152, New York, NY, USA, 2003. ACM.

[42] G. Wang and T. Ng. Distributed algorithms for stable and secure network coordinates. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 131–144. ACM, 2008.

[43] G. Wang, B. Zhang, and T. Ng. Towards network triangle inequality violation aware distributed systems. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 175–188. ACM, 2007.

[44] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford. DONAR: Decentralized Server Selection for Cloud Services. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 231–242. ACM, 2010.

[45] S. Wu, Y. Chen, X. Fu, and J. Li. Ncshield: securing decentralized, matrix factorization-based network coordinate systems. In *Quality of Service (IWQoS), 2012 IEEE 20th International Workshop on*, pages 1–9. IEEE, 2012.

[46] D. J. Zage and C. Nita-Rotaru. On the Accuracy of Decentralized Virtual Coordinate Systems in Adversarial Networks. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 214–224. ACM, 2007.

[47] B. Zhang, T. Ng, A. Nandi, R. Riedi, P. Druschel, and G. Wang. Measurement based analysis, modeling, and synthesis of the internet delay space. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 85–98. ACM, 2006.

[48] 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 *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12. IEEE, 2006.

[49] X. Zhao, A. Chang, A. D. Sarma, H. Zheng, and B. Y. Zhao. On the Embeddability of Random Walk Distances. *Proc. VLDB Endow.*, 6(14):1690–1701, 2013.

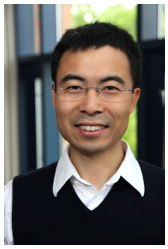
[50] H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin. Internet routing policies and round-trip-times. In *Passive and Active Network Measurement*, pages 236–250. Springer, 2005.

[51] J. Zhu, Y. Kang, Z. Zheng, and M. Lyu. Wsp: A network coordinate based web service positioning framework for response time prediction. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 90–97, 2012.

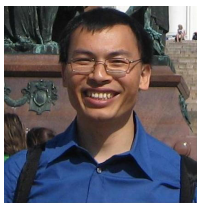
[52] Y. Zhu, Y. Chen, Z. Zhang, X. Fu, D. Li, B. Deng, and X. Li. Taming the triangle inequality violations with network coordinate system on real internet. In *Proceedings of the Re-Architecting the Internet Workshop (ReARCH'10)*, pages 7:1–7:6. ACM, 2010.

Yang Chen is a pre-tenure associate professor in the School of Computer Science at Fudan University. He received his B.S. and Ph.D. degrees from the Department of Electronic Engineering, Tsinghua University, in 2004 and 2009, respectively. From September 2009 to April 2011, he has been a Research Associate and the Deputy Head of Computer Networks Group, Institute of Computer Science, University of Goettingen, Germany. From April 2011 to September 2014, he worked as a Postdoctoral Associate at the Department of Computer Science, Duke University, USA. His research interests include Internet architecture, online/mobile social networking, and cloud computing.

Shining Wu received his B.S. degree from the Department of Electronic Engineering, Tsinghua University, in 2010. In September 2013, he received his M.S. degree in Computer Science from the University of Göttingen, Germany. He has published papers in several international conferences including IWQoS and ICC.



Jun Li is an associate professor in the Department of Computer and Information Science at the University of Oregon, USA, and directs the Network & Security Research Laboratory there. He received his Ph.D. from UCLA in 2002 (with honors), M.E. from Chinese Academy of Sciences in 1995 (with a Presidential Scholarship), and B.S. from Peking University in 1992, all in computer science. Specializing in computer networks, distributed systems, and their security, Dr. Jun Li is currently researching Internet monitoring and forensics, Internet architecture, social networking, cloud computing, and various network security topics. He has also served on several US National Science Foundation research panels and on more than 60 international technical program committees. He is currently an editor of Computer Networks and chair of several workshops and symposiums. Dr. Li is a 2007 recipient of the NSF CAREER award.



Xiaoming Fu received his bachelor and master degrees from Northeastern University, China, and his Ph.D. degree from Tsinghua University, Beijing, China. He was a research staff at the Technical University Berlin until joining the University of Göttingen, Germany in 2002, where he has been a professor in computer science and heading the Computer Networks Group since 2007. His research interests include network architectures, protocols, and applications. He is currently an editorial board member of IEEE Communications Magazine, IEEE Transactions on Network and Service Management, Elsevier Computer Networks, and Computer Communications, and has published over 150 papers in journals and international conference proceedings. He is the coordinator of EU FP7 GreenICN, CleanSky and MobileCloud projects, and the recipient of ACM ICN 2014 and IEEE LANMAN 2013 Best Paper Awards and the 2005 University of Göttingen Foundation Award for Exceptional Publications by Young Scholars.