KoNKS: Konsensus-style Network Koordinate System

Eric Chan-Tin
Oklahoma State University
218 MSCS, Stillwater, OK 74074
chantin@cs.okstate.edu

Nicholas Hopper
University of Minnesota
200 Union Street SE, 4-192 Keller Hall,
Minneapolis, MN 55455
hopper@cs.umn.edu

ABSTRACT

A network coordinate system assigns coordinates to each node in a network in such a way that the network latency between any two nodes can be accurately estimated by the distance between their coordinates. Although several network coordinate systems have been shown to be accurate and efficient, nearly all of the systems in the literature are insecure, in the sense that an attacker with knowledge of the scheme can cause arbitrary peers to produce inaccurate distance estimates. This includes several recently proposed "secure" network coordinate schemes.

We describe a new decentralized network coordinate system, KoNKS, and argue that it meets a well-specified security goal under a realistic threat model. We demonstrate that KoNKS is as accurate as current network coordinate systems, does not require any trusted entities, and is resistant against all known attacks, in addition to arguing for its security against future attacks within our threat model.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Security

Keywords

Security, network coordinate systems, network latency

1. INTRODUCTION

A network coordinate system [7,13,14] assigns virtual coordinates (network positions) to every node in the network. These coordinates are assigned so that the coordinate distance between two nodes reflects the real network distance between those two nodes. This allows any peer in the sytem to accurately estimate the network distance between any pair of nodes, without having the pair of nodes contact each other. Network coordinate systems' ability to predict the network latency between arbitrary pairs of nodes can be used in many applications: finding the closest node to download content from in a content distribution network or route to in a peer-to-peer system [17], reducing inter-ISP communication [5,12], reducing the amount of state stored in routers [1], performing byzantine leader elections [6], and detecting Sybil attackers [2,8].

Several schemes [10,16,18,19] have been developed to protect network coordinate systems against the attacks in [11], where malicious peers report randomly chosen coordinates, report random but consistent coordinates, or add random delay in their messages to other peers; all of the schemes were shown to effectively mitigate the known attacks. Recently, however, a new type of attack [3] – the frog-boiling attack – was introduced, and it was shown that some of these schemes fail to protect against this attack. The frog-boiling attacker reports small but consistent lies that are not detected by any of the security mechanisms, but which cumulatively introduce unacceptable errors; for example, it was shown that this technique can randomly partition an overlay using a secure network coordinate system [19]. One of the issues is that the current schemes aimed only to "patch" against the known attacks. This could lead to an *arms race* where new attacks bypass existing security mechanisms, resulting in new improved schemes to defend against the new attack, and so on.

To avoid this arms race, we evaluate a network coordinate system in terms of an explicit *security goal* – an invariant that should hold despite the presence and actions of an attacker – under a concrete threat model that states what resources the attacker can marshall. The two goals are 1) an attacker's influence on either the network distance or coordinate distance between two honest nodes is limited, and 2) the coordinate distance between a malicious peer and an honest peer cannot be smaller than the true network distance between these two nodes. The first goal limits an attacker's influence on honest nodes' coordinates while the second goal prevents an attacker from appearing closer than it actually is. Our security model will be described in more detail in Section 2.

Our main contribution is describing a completely decentralized network coordinate system, KoNKS, which is secure under our stated security model. KoNKS – consensus-style network coordinate system – modifies the objective function that each peer follows to update its coordinates. In current network coordinate systems, a peer's goal is to minimize the sum of the prediction errors for all of its neighbors. In contrast, using KoNKS, a peer's goal is to minimize the number of neighbors whose individual relative error is unacceptable

– KoNKS puts an upper bound on each neighbor's relative error. The relative error determines how accurate the coordinate system is, thus when there are no attackers, minimizing the sum of errors should lead to more accurate distance predictions. However, minimizing the sum of prediction errors allows each neighbor to have a significant influence on the position of its peers. This is one of the reasons why the frog-boiling attack works.

We show in Section 5 that KoNKS is as accurate as Vivaldi [7], one of the most popular decentralized network coordinate system (Vivaldi is implemented in Vuze [17] and is the basis for previous "secure" network coordinate systems [10,16,19]), and is secure against all the current attacks, including the network-partition frog-boiling attack. More specifically, KoNKS puts an upper bound on the amount of influence an adversary can have on the honest nodes. For example, 10% of attackers can partition a network using KoNKS only so much before their lies do not have any effect anymore because they are outside of the threshold, or the other honest neighbors' influence equals the malicious neighbors' influence. KoNKS with no attack can achieve a median relative error as low as 12%, which is comparable to Vivaldi's median relative error of 10%. KoNKS also incurs a very low overhead, similar to Vivaldi as coordinates can be piggybacked on top of application messages. The processing overhead of each node updating its coordinates is also small.

2. SECURITY MODEL

The network consists of n nodes, of which $0 \leq m < \frac{n}{4}$ are malicious. The number of malicious peers needs to be limited to at most 1/4 of the network so that satisfying a majority of the nodes guarantees that the median node is honest. Clearly, a botnet or Sybil attacker [8] could invalidate this assumption, but dealing with such attacks is beyond the scope of this paper[1]. We allow for a powerful adversary that can compromise any node in the network (up to 1/4 of the network), but cannot adaptively compromise nodes, so that if a peer chooses its neighbors randomly, with high probability only 1/4 of them will be malicious. Moreover, we allow the adversary to have global knowledge of the network, that is, it knows every peer's coordinate, real network distance to other peers, and neighbors. The adversary can also inflate its network distance to other peers by delaying its responses, and can report any coordinate it chooses. However, we do limit the adversary's power in the following sense: we assume that a malicious peer cannot decrease its real network distance to another peer. This can be achieved by using an unpredictable unique nonce in every message. Thus, a malicious peer cannot reply to a message faster than the network conditions allow.

We next describe how a network coordinate system functions. Every node computes its own k-dimensional Euclidean coordinates (for simplicity we assume Euclidean space). The real network distance between any two nodes A and B is denoted as $\mathsf{rtt(A,\ B)}$, and the coordinate distance between any two nodes A and B is denoted by $\mathsf{distance(A,\ B)}$. Embedding a higher-dimensional space (pairwise network distances) into a low-dimensional Euclidean space (for example 5-dimensional Euclidean coordinates) inherently produces *errors*, that is, it is in general not possible to obtain a perfect embedding from a high-dimensional space to a low-dimensional space. A good embedding minimizes the error produced. The prediction error between two peers A and B is defined as $|\mathsf{distance(A,\ B)} - \mathsf{rtt(A,\ B)}|$. We define the relative error as $\frac{|\mathsf{distance(A,\ B)} - \mathsf{rtt(A,\ B)}|}{\mathsf{rtt(A,\ B)}}$ A low relative error implies a low prediction error, but the converse is not necessarily true. We thus use relative error for our evaluation results. To calculate the accuracy of the whole network, the median relative error for all the nodes is usually used. The lower the relative error, the more accurate the network coordinates as the coordinate distance closely matches the real network distance. We define the network as having *converged* when the median relative error for all the nodes remains unchanged, that is, peers' coordinates are relatively unchanged.

We now define the two goals that any network coordinate system should meet in order to be secure.

1. The attacker cannot cause the median relative error between pairs of honest peers to exceed the threshold T.
2. The attacker cannot reduce its apparent distance to an honest peer by a factor of more than $(1 - T)$.

The first security goal is to prevent a "cascading" effect, where an attacker can greatly influence one honest node, which in turn, influences another honest node. Two honest nodes should be able to "embed" each other and calculate their real coordinate distance even if malicious nodes are present. If an attacker can affect the coordinate distance between two honest nodes, then a honest node will not be able to accurately calculate the coordinate distance to any other honest node.The second security goal is to ensure that an adversary cannot appear closer to another peer than it actually is. This prevents the attacker from being close to all the peers in the network. In certain applications such as a peer-to-peer system or closest-server selection, the closest peer is usually queried.

3. RELATED WORK

Existing secure network coordinate systems use various methods as security mechanism. An anomaly detection system such as the Kalman filter [10] uses a statistical method to determine if a reported coordinate is acceptable. Outlier detection mechanisms such as the Mahalanobis distance [19] reject reported coordinates that do not conform to past accepted coordinates. Veracity [16] is a distributed reputation system, where a peer's coordinate is verified by other peers in the network. All these schemes have been previously shown to be insecure by a new attack [3,4].

Treeple [4] is a provably secure and accurate network coordinate system, using centralized and trusted "vantage points". Treeple provides strong, worst-case security guarantees against a realistic adversary model, but requires a set of trusted authorities to achieve these guarantees. In contrast, KoNKS requires no central authorities and provides average case security, guaranteeing that each peer will have accurate latency estimates to the majority of its peers. Furthermore, Treeple generates "positions" that are not Euclidean coordinates whereas KoNKS generates Euclidean coordinates and can be used in place of the standard insecure schemes for any application that expects Euclidean network coordinates.

4. KONKS DESIGN

The objective function that each node in a current network coordinate system seeks to minimize is total prediction

[1]We note that an adversary controlling a botnet can likely cause more serious problems for an overlay than disrupting its network coordinate system.

error to all nodes. This is required for the network coordinates to be accurate, as a low prediction error implies that the coordinate distance is close to the network distance. In these schemes, every node maintains a list of neighbors – a subset of all the peers in the network. Each node will, regularly, pick a node from that list of neighbors to “contact” so as to update its own coordinate. For each coordinate update, a node will update its coordinate so that it eventually **minimizes the sum of prediction errors for all of its neighbors** as the end goal. Although this objective function allows a node to select a coordinate that is accurate (low relative error), it allows for “outliers” in the list of neighbors. For example, the sum of prediction errors might be small, with most neighbors' prediction error being very small and one neighbor's prediction error being large. In trying to minimize the sum of errors, the large error of the one neighbor can be decreased. However, this might double the small errors of all the other neighbors. Thus, an adversarial neighbor might disproportionately influence the updated coordinate of a peer.

To mitigate possible attacks and to reduce the influence of any one neighbor, we modify the objective function so that each node will **minimize the number of neighbors whose individual relative error is greater than a threshold** T. All the neighbors are considered at each location update. However, let's say a location C can be chosen so that most of the neighbors' individual relative error is less than T, except one neighbor N whose error is greater than T. If another location can be computed such that this neighbor's (N) error is less than T, but if this location change would make other neighbors' individual error greater than T, then the previous location C would be chosen instead. This objective function limits the influence each neighbor can have on a peer. More specifically, each neighbor has the same amount of influence as each other.

```
For each $P_i \in \mathcal{P}$
Choose $N_j \in \mathcal{N}_i \subseteq \mathcal{P}$
Define $\mathsf{Pair}(N_j) = (\mathsf{Coord}_{N_j}, \mathsf{rtt}(P_i, N_j))$

Procedure SendUpdate($N_j$):
Send $\mathsf{Coord}_{P_i}$ to $N_j$
Procedure ReplyUpdate($N_j$):
Send $\mathsf{Coord}_{P_i}$ to $N_j$
Send $\mathsf{rtt}(P_i, N_j)$
Procedure ReceiveUpdate($N_j, \mathsf{Coord}_{N_j}, \mathsf{rtt}(P_i, N_j)$):
$\mathsf{Pair}(N_j) = (\mathsf{Coord}_{N_j}, \mathsf{rtt}(P_i, N_j))$
Update()

Procedure Update():
Set $bestCoordinate \leftarrow null$
Set $bestNumSat \leftarrow 0$
for C $\in$ CoordSpace:
    Set $numSat \leftarrow$ NumSatisfied($C$)
    if $numSat > bestNumSat$:
        $bestNumSat \leftarrow numSat$
        $bestCoordinate \leftarrow C$
Output: $bestCoordinate$

Procedure NumSatisfied(coordinate):
Set $satisfiedNeighbors \leftarrow 0$
For each $N_j \in \mathcal{N}_i$ do:
    if relative_error(coordinate, $\mathsf{coord}_{N_j}$) $\leq T$:
        Increment $satisfiedNeighbors$ by 1
Output: $satisfiedNeighbors$
```

Figure 1: KoNKS algorithm

4.1 Algorithm

Figure 1 shows the algorithm for KoNKS. The set of n nodes is represented as $\mathcal{P}$. Every node $P_i \in \mathcal{P}$ maintains a list of neighbors $\mathcal{N}_i$. The bottom half of the figure outlines how a peer will update its coordinate.

4.2 Why is KoNKS Secure?

We argue that KoNKS is secure – it meets our two security goals from Section 2. We assume that it is possible to obtain a T-embedding for the honest nodes in the network. This means that it is possible to assign coordinates to honest nodes such that the coordinate distance between any pair of honest nodes will be different from the real network distance between that pair of nodes by a factor of at most T. In other words, the error for the network latency estimate will be at most T. In Section 5.2, we show that experimentally, $T = 0.2$ is adequate for the Internet. This means that all honest neighbors can be satisfied – the individual relative error of every honest neighbor will be less than T.

KoNKS satisfies the first security goal due to multiple reasons. First, each peer is influenced by its list of neighbors and if the attacker is not in that list, it cannot influence the peer's coordinate. Second, even if the neighbor list contains some adversarial nodes, the honest neighbors outnumber the malicious neighbors. Thus, the influence that the malicious neighbors can exert on the peer is limited. In the worst case, 1/4 of a peer's neighbors are malicious. Thus, the peer can satisfy $\frac{3}{4}$ of its neighbors. Even if all malicious neighbors are satisfied, at least half of the honest neighbors are also satisfied, and thus the median relative error of honest peers will be less than the threshold T.

KoNKS also satisfies our second security goal. A dishonest node N can claim to have coordinates that make its distance to peer P arbitrarily small. If N is on P's neighbor list, P will try to find coordinates that make its relative error to N acceptable, while also making its relative error for the other neighbors acceptable. Thus, N cannot reduce its apparent network distance to P by a factor of more than $1 - T$.

5. EVALUATION RESULTS

5.1 Setup

We implemented a simulator for our KoNKS algorithm and evaluated its accuracy and resistance to known attacks. For our simulation we used the King dataset [9] for latencies between nodes. Each peer in the network computes its own 5-dimensional coordinate. Every 10 seconds, each node will randomly select another peer in the network to SendUpdate. Each peer maintains 50 neighbors. Neighbors are randomly chosen from all the peers in the network. All attacks start after time 80 ticks, to allow the network to stabilize – that is, reach a stable median relative error. This translates to about 13 minutes. All of our simulations were run on a quad-core 2.67GHz Intel Xeon W3550 processor.

5.2 Simulation

Picking the threshold T: In the previous section, we mentioned the threshold T that all neighbor's individual relative error must satisfy. Recall that the relative error between two peers A and B is defined as $\frac{|distance(A,B) - rtt(A,B)|}{rtt(A,B)}$. A low relative error means that the system is more accurate. We now experimentally set the value of T and explain why this value produces accurate latency predictions. Figure 2(a) shows the median relative error for all the nodes in the net-

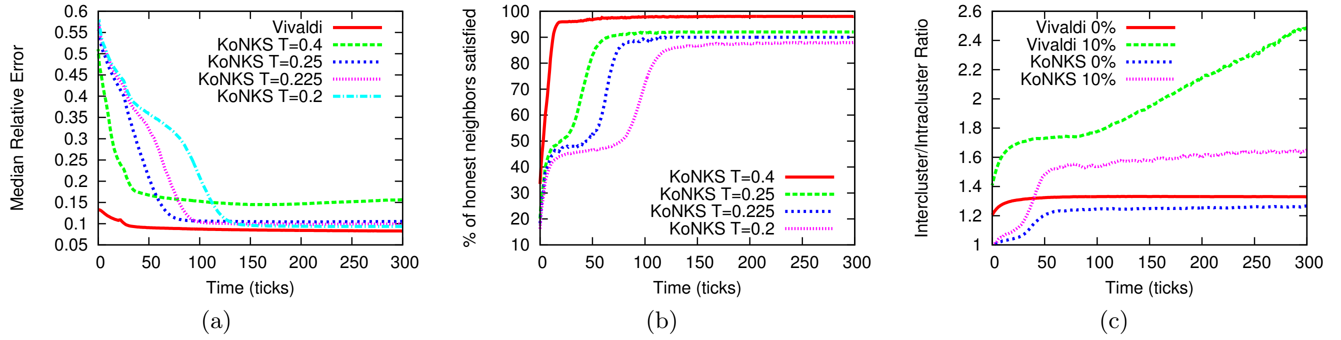

Figure 2: (a) The median relative error for Vivaldi and KoNKS with different thresholds. (b) The average percentage of honest neighbors whose individual relative error is less than T. (c) The intercluster/intracluster ratio for both Vivaldi and KoNKS with 0% and 10% of frog-boiling attackers.

work over time. The different lines indicate the different values of T for KoNKS. The figure shows that decreasing the threshold improves the accuracy of KoNKS – when $T = 0.2$, KoNKS's accuracy is comparable to Vivaldi's accuracy. This is expected as the lower the threshold, the lower the neighbor's relative error, and this implies that the median relative error should be lower than T. We note that the median relative error is **much lower** than T. When $T = 0.4$, the median relative error converges to 0.15. This means that the network latency prediction has an error of 15%, that is, the coordinate distance differs from the real network distance by 15%. The figure also shows that the lower the threshold, the longer KoNKS takes to converge to a stable equilibrium. From these observations, we picked $T = 0.25$ as our threshold – each neighbor's coordinate distance can differ from the real network distance by a factor of 0.25. Intuitively, a 25% error when estimating network latencies on the Internet is acceptable in practice. Moreover, from Figure 2(a), we see that when $T = 0.25$, the median error is actually only 12%.

The median relative error does not show the whole picture. Figure 2(b) shows the percentage of neighbors satisfied – their individual relative error is less than T. The higher the threshold, the more likely it is to satisfy neighbors, and the lower the threshold, the harder it is to choose a coordinate that can satisfy all neighbors. At $T = 0.25$, 90% of the neighbors can be satisfied. Although 10% of the neighbors cannot be satisfied and are not considered when computing the optimal coordinate, this does not mean that these neighbors are "bad". These same peers might be satisfied neighbors for another node.

5.3 Security

To empirically support our security argument, we implemented three of the previously proposed attacks – the "random" attack, the "inflation/deflation" attack, and the frog-boiling attack [3]. Due to space constraints, we only show our results for the frog-boiling attacks.

We implemented the frog-boiling attack from [3] in an attempt to partition the KoNKS network into two independent subnetworks. The malicious nodes pick the first "half" of the network N_1 to "move" to the coordinate $[-1000, -1000, -1000, -1000, -1000]$ and the second "half" of the network N_2 to move to $[1000, 1000, 1000, 1000, 1000]$. If successful, this attack partitions the whole network into two independent subnetworks N_1 and N_2. The malicious nodes behave normally and compute their best coordinate just like any honest peer. However, they lie about their location when they report it to the honest nodes. When a malicious node receives a SendUpdate request from an honest node, it will first determine which "half" that node falls into. If the honest node has not previously contacted the attacker, it will reply with its current best coordinate. If the honest node has previously contacted the attacker, it will reply with the last reported coordinate to that honest node $\pm\delta$ ($-\delta$ if the honest node falls into N_1 and $+\delta$ if the honest node falls into N_2). As [3] showed, each lie is small and not detected by anomaly detection, but cumulatively, the lies add up such that the network is effectively partitioned. We set $\delta = 20$ in our experiments. The higher δ is, the faster the attack, but the higher the chance of being detected. We expect that KoNKS will not be affected by this attack.

Figure 2(c) shows the intercluster/intracluster ratio for both KoNKS and Vivaldi with no attackers and 10% of malicious nodes. The intercluster/intracluster ratio indicates how far apart the two networks are. A ratio of two means that on average, nodes are twice as far from the center of the opposite cluster as they are from the center of their own. The higher the ratio, the further apart the two networks are. With no attack, both KoNKS and Vivaldi stabilize to a ratio of 1.2. The ratio is not 1 as we always use the same partition of the network to be N_1 and the same other half to be N_2. With 10% of attackers, the ratio for Vivaldi keeps increasing over time. Although the intercluster/intracluster ratio for KoNKS increases from 1.2 to 1.6, the ratio remains stable over time. This reinforces our argument that the attacker's influence on honest KoNKS nodes is limited. The malicious neighbors can affect their honest peers only so much before they stop having a malicious effect. Figure 3(a) shows the corresponding median relative error. Vivaldi's median relative error keeps increasing with 10% of malicious nodes, whereas KoNKS' median relative error remains mostly unchanged even under attack. We obtained a similar result for higher percentages of frog-boiling attackers.

So far, we have shown that KoNKS is secure against all the known attacks. Figure 2(b) shows the percentage of neighbors which can be satisfied. We also show that even under attack, honest KoNKS peers can still satisfy a high percentage of honest neighbors, while not satisfying the malicious neighbors. Figure 3(b) shows the percentage of honest neighbors satisfied for varying percentages of malicious nodes in the network. We observe that whichever attack is used or the percentage of malicious peers, honest KoNKS peers can still satisfy most honest neighbors.

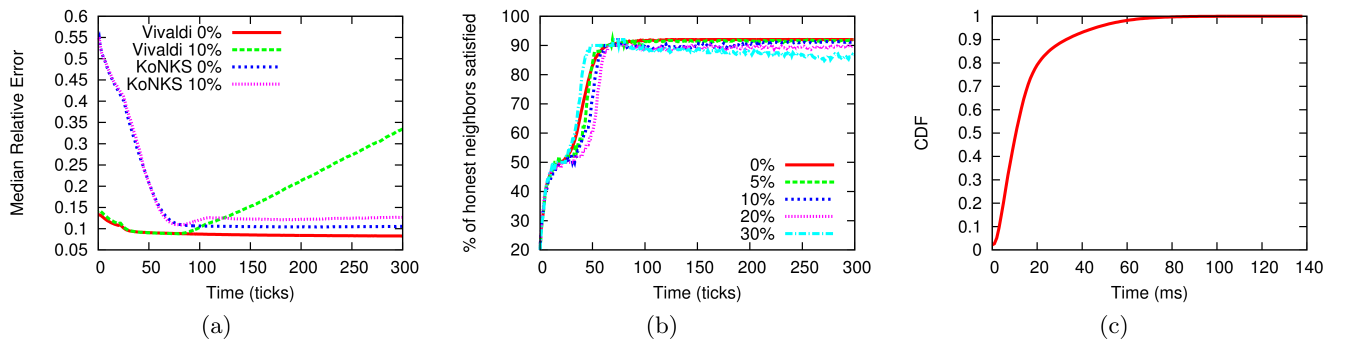

Figure 3: (a) The median relative error for both Vivaldi and KoNKS with 0% and 10% of frog-boiling attackers. (b) The average percentage of honest neighbors satisfied for honest nodes under attack in KoNKS with $T = 0.25$. (c) The CDF for the time required to complete the search algorithm to find the best coordinate for each update.

5.4 Overhead

As mentioned before, the communication overhead is small. Coordinates can be piggybacked on top of application messages. The processing overhead (the search algorithm to find the optimal coordinate) is also small. Figure 3(c) shows that the median time for a peer to pick its best coordinate at each update is 10ms. The code is not multi-threaded, thus it is expected that the update time can be further reduced.

5.5 Experiments

We also performed our experiments on PlanetLab [15], with the same implementation details as for the simulator. Similar results were obtained on PlanetLab and are omitted due to space constraints.

6. CONCLUSION

Although network coordinate systems can accurately predict the network latency between two peers, current systems are not secure in the sense that an adversary can disrupt the whole network by increasing the error, which in turn means that the network latency prediction is no longer accurate. It was previously shown that even the "secure" schemes are vulnerable to the frog-boiling attack. We introduce a new decentralized network coordinate system, KoNKS, and argue that it will be secure not only against known attacks, but also against future attacks that work within our threat model. KoNKS aims to achieve consensus among all the neighbors of a peer, such that, the individual relative error of each neighbor is less than the threshold $T = 0.2$.

We found experimentally that setting the threshold to 0.2 produced a low relative error, comparable to Vivaldi, and allows the network to converge quickly. The median relative error for KoNKS with no attacker is 0.12, compared to 0.10 for Vivaldi. This means that the network distance prediction differs from the real network distance by 12%. With 10% of malicious nodes, the intercluster/intracluster ratio – a measure of how far away from each other the two networks are – increases from 1.2 to 1.6, but remains **stable**, contrary to the frog-boiling attack on Veracity or Vivaldi, where the two networks keep getting further apart over time.

Acknowledgments: This work was supported by the National Science Foundation under grant CNS-0716025.

7. REFERENCES

[1] I. Abraham and D. Malkhi. Compact routing on euclidian metrics. In *PODC: ACM symposium on Principles of distributed computing*, 2004.

[2] R. A. Bazzi and G. Konjevod. On the establishment of distinct identities in overlay networks. In *PODC: ACM symposium on Principles of distributed computing*, 2005.

[3] E. Chan-Tin, D. Feldman, Y. Kim, and N. Hopper. The Frog-Boiling Attack: Limitations of Anomaly Detection for Secure Network Coordinates. *SecureComm*, 2009.

[4] E. Chan-Tin and N. Hopper. Accurate and Provably Secure Latency Estimation with Treeple. *NDSS*, 2011.

[5] D. R. Choffnes and F. E. Bustamante. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. *SIGCOMM Comput. Commun. Rev.*, 38(4):363–374, 2008.

[6] J. Cowling, D. Ports, B. Liskov, R. A. Popa, and A. Gaikwad. Census: Location-Aware Membership Management for Large-Scale Distributed Systems. *In proceedings of USENIX Technical Conference*, 2009.

[7] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM*, 2004.

[8] J. R. Douceur. The sybil attack. In *IPTPS '01: International Workshop on Peer-to-Peer Systems*, 2002.

[9] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: estimating latency between arbitrary internet end hosts. In *IMW: ACM SIGCOMM Workshop on Internet measurment*, 2002.

[10] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti, and W. Dabbous. Securing internet coordinate embedding systems. *SIGCOMM Comput. Commun. Rev.*, 37(4):61–72, 2007.

[11] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous. Real attacks on virtual networks: Vivaldi out of tune. In *LSAD: SIGCOMM workshop on Large-scale attack defense*, 2006.

[12] C. Lumezanu, D. Levin, and N. Spring. Peer wise discovery and negotiation of faster path. *HotNets*, 2007.

[13] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *IEEE INFOCOM*, pages 170–179, 2001.

[14] T. S. E. Ng and H. Zhang. A network positioning system for the internet. In *USENIX Technical Conference*, 2004.

[15] PlanetLab. `http://planet-lab.org`, Accessed 2011.

[16] M. Sherr, M. Blaze, and B. T. Loo. Veracity: Practical Secure Network Coordinates via Vote-based Agreements. In *USENIX Annual Technical Conference*, 2009.

[17] Vuze. `http://azureus.sourceforge.net`, Accessed 2011.

[18] G. Wang and T. E. Ng. Distributed algorithms for stable and secure network coordinates. In *IMC: ACM SIGCOMM conference on Internet measurement*, 2008.

[19] D. J. Zage and C. Nita-Rotaru. On the accuracy of decentralized virtual coordinate systems in adversarial networks. In *CCS: Proceedings of the ACM conference on Computer and communications security*, 2007.