The Frog-Boiling Attack: Limitations of Anomaly Detection for Secure Network Coordinate Systems

Eric Chan-Tin, Daniel Feldman, Nicholas Hopper, Yongdae Kim
{dchantin, feldman, hopper, kyd}@cs.umn.edu
University of Minnesota

Abstract. A network coordinate system assigns Euclidean "virtual" coordinates to every node in a network to allow easy estimation of network latency between pairs of nodes that have never contacted each other. These systems have been implemented in a variety of applications, most notably the popular Azureus/Vuze BitTorrent client. Zage and Nita-Rotaru (CCS 2007) and independently, Kaafar *et al.* (SIGCOMM 2007), demonstrated that several widely-cited network coordinate systems are prone to simple attacks, and proposed mechanisms to defeat these attacks using outlier detection to filter out adversarial inputs. We propose a new attack, Frog-Boiling, that defeats anomaly-detection based defenses in the context of network coordinate systems, and demonstrate empirically that Frog-Boiling is more disruptive than the previously known attacks. Our results suggest that a new approach is needed to solve this problem: outlier detection alone cannot be used to secure network coordinate systems.

Key words: Vivaldi, Anomaly Detection, Network Coordinate Systems

1 Introduction

Network coordinate systems assign virtual coordinates to every node in a network. These coordinates allow efficient estimation of the latency between any pair of nodes in the network: instead of directly measuring the $O(n^2)$ pairwise latencies, each of the n nodes computes its coordinates based on the round-trip time to a few other nodes and their coordinates, greatly reducing the communication costs. Several possible uses of network coordinate systems include choosing peers to download from in a filesharing network [1], choosing peers for routing in a DHT [2], or finding the closest node in a content-distribution network. A popular BitTorrent client, Azureus (now called Vuze [3]), is currently using a network coordinate system to prioritize lookups based on network distance and to find closer nodes [4].

There have been several network coordinate systems proposed in the literature; these schemes can be categorized into centralized or "landmark"-based systems [1, 5, 6] that depend on a small set of "trusted" nodes, and decentralized systems [7, 8]. A widely-implemented and studied example of decentralized coordinate systems is Vivaldi [7], which has been shown to produce accurate estimations and converge quickly under various network conditions. Although it is decentralized, Vivaldi can be easily disrupted by spurious or malicious nodes, rendering the network coordinate system useless and impractical since the nodes never reach a stable coordinate. Zage and Nita-Rotaru [9] proposed a mechanism, based on real-time statistical analysis of nodes' coordinates, to detect and discard adversarial inputs. A similar mechanism was proposed and evaluated by Kaafar *et al.* [10]. Both methods rely on outlier detection using statistical models – respectively, the Mahalanobis distance and Kalman filters – of coordinate evolution.

In this paper, we demonstrate the inherent challenge in designing a secure network coordinate system using outlier detection. We propose the *Frog-Boiling attack*, where

an adversary disrupts the network while consistently operating within the threshold of outlier detection. This is analogous to the popular account that a frog put in hot water will quickly jump out but a frog placed in cold water that is gradually brought to a boil will not notice the change and boil to death. The adversary sends "small-step" fake updates (fake RTTs or self-reported error or coordinate) [1] to nodes in the network. The "step" is small enough that it does not trigger the anomaly detection but the nodes attacked are still affected. Thus, the coordinates of the nodes in the attacked network quickly become very different from the coordinates of the same nodes in the original network. The effectiveness of the attack can also be significantly increased when conducted in conjunction with a *Sybil* attack.

We implement, and empirically evaluate, three variants of the Frog-Boiling attack to demonstrate its effectiveness against outlier-detection based defenses. All three attacks rely on a simple concept: lying can be harmful but telling consistent, believable lies is even more harmful. Our evaluation on a PlanetLab deployment of Vivaldi shows that even the basic frog-boiling attack is *more disruptive* against the security mechanism proposed in [9] than the attacks they defend against. In particular, with only 5% of attackers in the network, Frog-boiling causes a median relative error of 0.28 after two hours and 0.57 after 14. The same network with no attackers has a median relative error of 0.11, and under Zage and Nita-Rotaru's "random" attack, the insecure coordinate scheme has a maximum median relative error of 0.22, even when the fraction of attackers is above 10%. Thus the outlier detection mechanism is completely ineffective against frog-boiling. We note that while the step size of the attack is small – nodes are pushed "little by little" – the result of the attack is neither slow nor small, resulting in similar errors just as quickly as previously known attacks but causing greater damage over time. See Section 5.3 for more details.

While similar attacks on outlier detection mechanisms appear in the literature (including [12, 13]), to our knowledge we are the first to demonstrate the effectiveness of frog-boiling in the context of network coordinate systems. Furthermore, we demonstrate that the attacks are *more disruptive* than previous work and are completely unmitigated by the existing approaches to securing network coordinate systems. These results suggest that new approaches and/or stronger assumptions are needed to construct secure network coordinate systems.

The remainder of the paper is organized as follows. We give a brief background on network coordinate systems, existing attacks, and the outlier detection mechanisms in Section 2. A detailed description of the attacks outlined above is given in Section 3. The evaluations of our experiments on a wide area network are shown in Section 4 and Section 5. Finally, we conclude in Section 6.

2 Background

2.1 Network Coordinate Systems

The first network coordinate systems developed were centralized – trusted infrastructure nodes compute coordinates for all other nodes. Centralized systems typically require a

[1] This is possible since updates are usually done via the application level, and an adversary can easily delay or hasten [11] replies

significant fraction of all network nodes to act as trusted servers, which is not possible for large networks. Centralized network coordinate systems include IDMaps [6], GNP [1] and NPS [5].

To improve the ease of deployment of network coordinate systems, decentralized network coordinate systems were introduced. A decentralized network coordinate system has no infrastructure nodes. Instead, normal nodes pick peers out of the set of all nodes, and compute their own coordinates with respect to those peers only. Finding potential peers is delegated to the underlying network. Decentralized network coordinate systems are attractive for P2P applications, since they can be deployed alongside the client software. Moreover, decentralized network coordinate systems are scalable as there are no centralized servers which could become overloaded.

Vivaldi Vivaldi [7] is a decentralized network coordinate system. It is based on a spring model. Its behavior is analogous to a physical model made of springs and balls, in which each ball represents a network node and the spring connecting any two balls is longer when the latency between those nodes is larger. Over time, such a model reaches a stable equilibrium. A Vivaldi node begins by selecting an arbitrary set of peers, and sets its initial coordinate to the origin. It then begins an iterative algorithm that pulls it closer to peers with lower latencies, and pushes it away from peers with higher latencies. After many iterations, the coordinate system reaches an equilibrium, and subsequent changes are due only to the changing latency between nodes. Each node will pick 64 other nodes in its reference set – 32 nodes are "close" and 32 nodes are "far". On each iteration, a Vivaldi node sends a probe packet (which could be piggybacked on top of application-level messages) to each of its peers. It receives a response to each probe packet containing the peer's current coordinate and self-reported error estimate (can also be piggybacked on top of application-level messages), and learns its latency to that peer from the RTT of the transaction. It then computes a new position that is closer to the peer if the estimated latency is too large, and farther from the peer if the estimated latency is too small. Vivaldi's coordinate system is *n*-dimensional. It was shown in [7] that 2 dimensions plus height work well for most cases. Moreover, Vivaldi boasts a low convergence time, a low reported error, and an accurate mapping of the virtual coordinate network. Vivaldi also deals well with *churn* – the constant change in membership of a P2P network due to its public nature – because of its low convergence time. However, Vivaldi was not designed for an adversarial environment and it is simple for an attacker to disrupt the whole network.

Pyxida Pyxida [14] implements a virtual coordinate network. It is being used in both academia and commercially – to track the coordinates of all the PlanetLab [15] nodes; in the Azureus [3] BitTorrent client; and to study selfish neighbor selection in P2P networks [16]. It is designed to work on a P2P network and implements the Vivaldi algorithm. Pyxida coordinates use 4 dimensions plus height. Moreover, it is open-source, enabling easy modification to implement the countermeasures and attacks. We used Pyxida in our experiments since it implements the Vivaldi algorithm, provides a stable network coordinate system, and has been used in a large-scale deployment [17]. A detailed description of Pyxida is given in [18].

2.2 Existing Attacks

Several attacks have been proposed [9, 10, 19]. They are the *Disorder attack*, *Repulsion attack*, *Colluding Isolation attack*, *Inflation/Deflation attack*, and the *Oscillation attack*.

The Repulsion and Colluding Isolation attacker sends the same coordinates each time in an attempt to move the victim nodes to some coordinate space. The other attacks consist of the attacker reporting random coordinates and a low error. The reader is referred to those papers for a more detailed description of the attacks.

2.3 Countermeasures

Several mechanisms, based on outlier detection, have recently been proposed to secure network coordinate systems.

Kalman Filter. Kaafar *et al.* [10] propose to implement a Kalman filter [20] to detect outlier hosts in the network, that is, hosts that are lying or behaving strangely. The Kalman filter works by comparing the previous trajectory of a node's coordinates with its coordinates after an update. If the distance between the expected coordinates and the update is larger than the threshold for the Kalman filter, then the update is rejected. The authors estimate that in order to resist the disorder attack, about 10% of the network must be trusted "surveyor" nodes.

Mahalanobis Distance. Zage *et al.* [9] proposed a countermeasure that uses two statistical filters to ignore peers that report unusually large or rapidly changing coordinates. The first filter is called the *spatial filter*, while the second is called the *temporal filter*. Each node applies both filters to incoming data from its peers, and discards data that do not pass both filters. The Mahalanobis outlier detection function used by the spatial filter determines if the new spatial vector falls inside an ellipsoid defined by previously-seen vectors. The temporal filter looks at the change in the last iteration. Since the data set is much larger, a constant-time and constant-space but slightly less accurate variant of the Mahalanobis function is used for this filter. Since the cost of a false positive is small, nodes can afford to set their thresholds very low. However, if the thresholds are too low, nodes will only accept data points that fit into a small range, leading to inaccurate coordinates. To our knowledge, the correct choice of thresholds to maximize security vs correctness has not been studied. When a peer's data fails either the spatial or temporal filter, there are two consequences. First, that peer's data is not used to update the node's current coordinate. Second, that peer's data is not used as history for the filters in the next iteration. However, there is no permanent blacklist of nodes which failed the filters. For a more detailed description, see [9].

In this paper, we attack Pyxida with Mahalanobis distance-based outlier detection. However, because the Kalman filter approach also features a threshold region in which updates will be accepted (and incorporated into the filter) we do not expect the Kalman filter to offer any significant defense against frog-boliing.

3 Proposed Attacks

Recall that the ellipsoid used to determine whether a new data point falls within acceptable bounds has axes with lengths that are multiples of the variances of the variables used in each filter. New data points are accepted if they fall inside this ellipsoid, and rejected otherwise. This mechanism correctly identifies a small number of spurious nodes that return random coordinates with low error. Since correctly operating nodes are unlikely to change coordinates much faster than average while still reporting low error, nodes that do so must be spurious.

However, an intelligent adversary can send "random" data points that still fall inside the Mahalanobis ellipsoid. Thus, the data points will be accepted although they are

"wrong". We call this approach the *Frog-Boiling attack*. If the adversary lies too much, its peers won't accept its updates. If it lies too little, the attack won't succeed in disrupting the network. The Frog-Boiling attack can be used to disrupt the whole network by continuously lying to all the nodes.

As a simple example, assume there are only two nodes A and B in the network and they have converged to stable coordinates. An attacker node C is introduced and obtains its coordinates from both A and B. However, each time C receives a request (say from A), it replies with $Coord_C = Coord_C + \delta$, where δ is a small offset. For example, if its coordinates in 2-dimensions (Pyxida uses 4-dimensions with height) are $(120, 100)$, the reported coordinate will be $(120.5, 100.5)$. Since the coordinate reported is not outside of the Mahalanobis thresholds, A will accept the coordinate and update its own coordinate accordingly. Then whenever B queries A, the response will be a coordinate that is slightly higher than what the "real" coordinate should have been. Thus, B's coordinate changes slightly as well. This process continues with the attacker continuously lying in small increments about its own coordinate. This whole process might just shift the coordinates, but not affect the estimated distance between any two nodes. Thus a targeted attack can be performed and as we show in Section 5, our attack effectively renders the network coordinate ineffective.

The targeted frog-boiling attack works as follows. The attacker attempts to move some victim nodes (a fraction of the whole network) to some arbitrary network coordinates. The targeted location in this case is far from the rest of the network. Although those nodes can still communicate with the rest of the network, they will not be able to calculate a correct coordinate for themselves and will report a "false" coordinate and error to the rest of the network. The Mahalanobis distance will flag those nodes as outliers and will not accept their updates. This effectively isolates the victim nodes from the rest of the network.

One way of performing this attack is for the attacker to consistently report its coordinates to the victim nodes so that the latter end up to coordinate space A. Note that the attacker will not be able to pull the victim nodes all the way to A, but the victims will be closer to A than the rest of the network. This is because, although the rest of the network might not accept updates from the victim nodes, the latter will still accept updates from the rest of the network. Thus, the victims are pushed to A by the attacker but also pulled back to the rest of the network. The success of the attack is for the attacker nodes to exert a greater force on the victim nodes than the rest of the network.

In this paper we evaluate three variants of this attack against Zage and Nita-Rotaru's secure network coordinate system. All three attacks rely on the same concept of consistently and progressively lying:

- The **Basic-Targeted** attack is as described above.
- The **Network-Partition** attack is an extension of the previous attack,where the whole network is partitioned into two subnetworks or clusters.
- The **Closest-Node** attacker tries to become the closest node (in terms of coordinate space) to the victim nodes.Becoming the closest node might not be important by itself. However, if the network coordinate system is used with an application such as in Azureus, then the closest node could be used to initiate file transfer. If the attacker becomes the closest node to a victim node, it will then be the first node that the victim contacts for a file. This can have various implications such as preventing

any node in a file-sharing network from being able to download a file. This attack is performed in a similar way to the targeted attack. Instead of pulling the victim node to a certain coordinate space, the attacker pushes itself close to the victim node. One way of doing this is for the attacker (after learning the victim's coordinate) to report its network coordinates as being very close to that of the victim's.

4 Experimental Setup

To evaluate the impact of our attacks on a secure network coordinate system, we deployed a standalone Pyxida service (see Section 2) on PlanetLab [15]. Since the original Pyxida code implements the basic Vivaldi coordinate system, the Mahalanobis distance outlier detection mechanism proposed in [9] was added to the Pyxida code using a third-party library [21].

We made some small modifications to Pyxida before deploying it. The neighbor list was modified to contain a maximum of 32 nodes (due to an estimated PlanetLab network size of 400). We used 50 nodes as the common "bootstrap" nodes, that is, all the Pyxida nodes contact those nodes when they first start. We wait until the network stabilizes before introducing any adversaries in the network.

The metric we used is the median relative error (henceforth just called error). It is calculated as $\frac{|RTT_{estimated} - RTT_{actual}|}{RTT_{actual}}$, where RTT_{actual} is the actual RTT between two nodes and $RTT_{estimated}$ is the RTT obtained by taking the difference in the coordinates of the two nodes. The lower this number is, the more accurate the network coordinate system is (each node believes it has the right coordinate). This is the same metric used in various other papers [9, 17, 18].

We use both a spatial and temporal threshold of 5 for our experiments.The network starts to stabilize after only 2 hours, indicating a low convergence time. The median relative error was 0.1. The attackers join the network at time 2 hours. The experiments for determining the best thresholds, as well as the other metrics used (such as relative rank loss [22]), will be described in the full version of this paper. We note that most of the experiments were also performed using a simulated network to verify implementation correctness. The results of these simulations are consistent with experimental results and are thus omitted due to space constraints.

5 Attack Evaluations

5.1 Previous Attacks

To establish a baseline for comparison with the effectiveness of our attacks, we implemented the previously proposed "coordinate oscillation" attack [9] (in which attacker nodes report completely random coordinates with low relative error) and measure the performance of the attack against our Pyxida deployment (without the Mahalanobis distance filter). The progress over time of the median relative error with 11% attacker nodes is shown below.

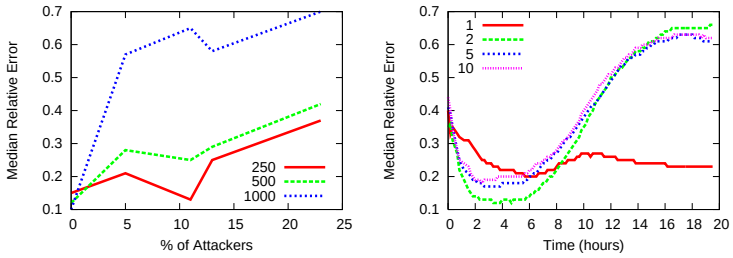
Time (mins)	100	250	500	750	1000
Relative Error	0.23	0.21	0.23	0.22	0.2

5.2 Basic-Targeted Attack

The Basic-Targeted attacker targets a victim node and attempts to change the victim's coordinate in small steps. We attempt to change the coordinate of the victim nodes to be $Loc_T = (2000, 2000, 2000, 2000)$ with height 2000. Initially, for each victim node (say coordinate C), the attacker node will report its coordinate to be $C' = C + \delta$. For

each subsequent time that victim node contacts our attacker node, the latter reports its coordinate as $C'' = C' + \delta$, until $C'' = Loc_T$. Thus, the victim's coordinate is moved in small steps to the target coordinate.

Recall from Section 2 that a Pyxida node only updates its coordinate when it has sent a "ping" request. Thus, the victim nodes have to contact the attacker nodes for the attack to work. With 10% of attackers, the victim will contact one attacker node 10% of the time. Once an attacker node becomes a neighbor of the victim, it will stay in the neighbor's list for at least the next 32 iterations, which is long enough for another attacker to be contacted and added to the list. The probability of an attacker node being part of the neighbor list after 32 iterations is $1 - 0.9^{32} = 96.5\%$. Thus, there is a very high probability that a victim node will have at least one attacker node in its neighbor list. Recall that the neighbor list is used every 10 seconds in Pyxida to calculate the current force. Since the attacker is updating its coordinate to be closer to the target coordinate at each time step, the victim will thus go closer to the target coordinate progressively. The Mahalanobis distance does not work in this case because the attacker is within the thresholds (since δ is small). The attacker only attacks the victim nodes and does not respond to other nodes in the network. Since there is no gossiping in Pyxida, this does not affect the attack.



(a) (b)

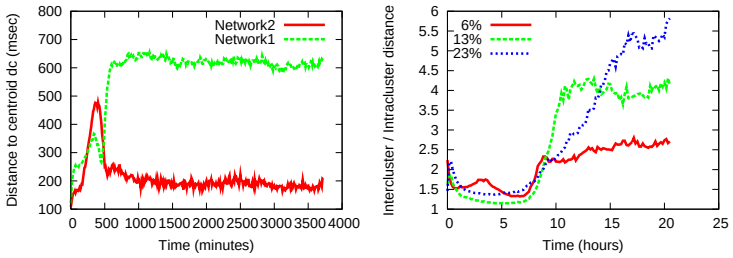
Fig. 1. The average median relative error for (a) varying % of attackers at different timestamps, (b) the targeted nodes with 11% of attackers over time and with different values of δ

Figure 1(a) shows the error with varying percentage of attackers. (We note that 20% of attacker nodes may seem high, but many of the applications that implement network coordinate systems are vulnerable to Sybil attacks that make it trivial to control a large fraction of the nodes) The different lines show the error at different times (250 minutes, 500 minutes, and 1000 minutes). Adding more adversaries significantly increases the error (by more than 100% with only 11% of attackers). The error is increased from 0.12 with no attackers to 0.25 with 11% of attackers, an increase of 108%. After 1000 minutes (a little over 16 hours), it can be seen that the network coordinate is unusable even with only 5% of the network being malicious – the error is greater than 0.5.

The frog-boiling attack on the secure network coordinate system is as effective as a random attack on the original network coordinate system. At time 500 minutes, the error for the random attack is 0.23 while the error for the frog-boiling attack is 0.25 with 11% of attackers. This means that the Mahalanobis distance does not provide any extra protection to a network coordinate system. This reinforces our belief that an outlier detection system is not suitable to secure a network coordinate system.

5.3 Aggressive Frog-Boiling

Our attack works by moving the victims in small steps to some coordinate. In the previous section, the step size δ was $2ms$. In this section, we varied the value of δ to test the effect of a more aggressive attack, which will produce an impact on the network earlier – in other terms, we show how quickly our attack can have an impact on the network. Figure 1(b) shows the errorwith 11% of attackers in the network. The different lines show the different δ values used – 1, 2, 5, and 10. With δ equal to 1 and 2, the error stays the same until time 6 hours, so it take 4 hours for the attack to start having an effect. On the other hand, with δ equal to 5 or 10, the relative error starts to increase at time 4 hours – after only 2 hours, the victim's network coordinates start to be disrupted. Thus, out attack is fast and efficient.



(a) (b)

Fig. 2. (a) The coordinate distance to the centroid and (b) the intercluster / intracluster ratio for the Network-Partition attack

5.4 Network-Partition Attack

The Network-Partition attack is similar to the Basic-Targeted attack. Instead of just moving the victim nodes (*Network1*) to some far-away coordinate, the rest of the network (*Network2*) is also moved to some other location. This effectively partitions the network into two subnetworks. The targeted coordinate for Network1 was set to $P_1 = (1000, 1000, 1000, 1000)$ with height 1000 and the targeted coordinate for Network2 was set to $P_2 = (-1000, -1000, -1000, -1000)$ with height -1000.

In our experiment, 6% of the nodes were adversaries, 37% of the network was assigned to Network1 and 57% of the network was assigned to Network2. Figure 2(a) shows the distance to the origin of the network for Network1 and Network2. At the beginning, the two clusters are close together. At time 500 minutes, which is how long it takes for the attack to have an effect, the two networks start to diverge. Network1 is pushed toward P_1 while Network2 is pushed toward P_2. Since the two clusters continue to exert some pull on each other, the intended coordinates are not reached, but the network is still effectively partitioned.

Figure 2(b) shows the ratio of the intercluster distance to the intracluster distance. The intercluster distance is the average of the distance from Network1 to the centroid of Network2 and the distance from Network2 to the centroid of Network1. The intracluster distance is the average of all the nodes in a cluster to the centroid of that cluster. The ratio shows how far apart the two clusters are moving from each other. The figure shows that over time, the two networks are getting pulled further apart from each other. The different lines show different fractions of attackers. This shows that our attack effectively partitions the whole network into two smaller networks far apart from each

other. We note that this attack could easily be extended to support partitioning into an arbitrary (constant) number of clusters with arbitrary membership ratios.

5.5 Closest-Node Attack

An adversary tries to become the closest node (in terms of coordinate space) to a victim in the Closest-Node attack. The attacker node queries the victim nodes constantly to obtain their coordinates. When a victim node queries the attacker node, it will reply back with that victim node's coordinate $+\delta$. The attacker node does not reply to other nodes in the network. We took a snapshot at 500 minutes and determine how many times one of the attacker nodes was reported as being the closest neighbor of a victim node (this reporting is done every 10 minutes). With only 11% of attackers, we find that an attacker is able to become the closest neighbor to a victim node 41% of the time.

6 Conclusion

A stable, decentralized network coordinate system could potentially provide a beneficial service for many Internet applications. However, existing systems provide no protection against malicious participants: even a single adversary can cause the entire coordinate system to fail. The apparent solution to such a dilemma is to add an anomaly detection mechanism to the coordinate system. Previous studies have shown that such a mechanism can prevent adversaries from disrupting the network. However, protection against more complicated adversaries is fraught with difficulty.

Consider a node in a network coordinate system that has some outlier detection mechanism. In order for the node to determine its coordinates, it must learn about the coordinates of its peers – it must accept some updates. The range of updates it accepts must be based on recent history, since network topologies and conditions vary widely. However, under these two assumptions an adversary can slowly expand the range of data accepted by the node by influencing the node's recent history. We call this attack the *Frog-Boiling* attack. In this paper we have introduced three variants of the frog-boiling attack and empirically demonstrated that the attack effectively disrupts the Vivaldi network coordinate system to a greater extent than previous attacks, and that the attack is completely unmitigated by Mahalanobis distance-based outlier detection. There is no reason to believe that Frog-Boiling would not be equally effective against Kalman filter-based outlier detection; we leave the evaluation of this claim for future work.

The task of securing a distributed network coordinate system against adversaries seems very challenging. The problem is that the current distributed network coordinate system mechanisms (secure or not) rely only on a node's local view of the network. Because of this, it is a challenge for a node to know whether a reported coordinate and RTT is correct or faked. Thus, a secure network coordinate system will need to provide some mechanism to verify a node's reported coordinates and/or RTTs. The success of the Frog-Boiling attack demonstrates that outlier detection is not a secure mechanism to provide this service. Recent work based on reputation or trust mechanisms [23, 24] may provide an alternative approach, but the difficulty of constructing secure reputation systems suggests that these schemes will also require careful evaluation.

Acknowledgments. We thank Jonathan Ledlie and Peter Pietzuch for their help with Pyxida, and Eugene Vasserman for pointing out the analogy to "boiling a frog." This work was supported by the NSF under grant CNS-0716025. No frogs were harmed in the writing of this paper.

References

1. Ng, T.S.E., Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches. Proceedings of IEEE (INFOCOM) (2002)
2. Dabek, F., Li, J., Sit, E., Robertson, J., Kaashoek, M.F., Morris, R.: Designing a DHT for low latency and high throughput. In: Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI). (2004)
3. Azureus: `http://azureus.sourceforge.net`
4. Vuze Forums: `http://forum.vuze.com/thread.jspa?threadID=80764`
5. Ng, T.S.E., Zhang, H.: A network positioning system for the internet. Proceedings of the USENIX annual technical conference (2004)
6. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A Global Internet Host Distance Estimation Service. IEEE/ACM Trans. Netw. **9**(5) (2001) 525–540
7. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: A Decentralized Network Coordinate System. In: Proceedings of ACM SIGCOMM. (2004)
8. Costa, M., Castro, M., Rowstron, A., Key, P.: PIC: Practical Internet Coordinates for Distance Estimation. Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS) (2004)
9. Zage, D.J., Nita-Rotaru, C.: On the accuracy of decentralized virtual coordinate systems in adversarial networks. In: Proceedings of the 14th ACM conference on Computer and communications security (CCS). (2007)
10. Kaafar, M.A., Mathy, L., Barakat, C., Salamatian, K., Turletti, T., Dabbous, W.: Securing Internet Coordinate Embedding Systems. Proceedings of ACM SIGCOMM (2007)
11. Su, A.J., Choffnes, D.R., Kuzmanovic, A., Bustamante, F.E.: Drafting Behind Akamai (Travelocity-Based Detouring). Proceedings of ACM SIGCOMM (2006)
12. Denning, D.E.: An Intrusion-Detection Model. IEEE Transactions on Software Engineering, Vol. SE-13, No. 2 (1987)
13. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can Machine Learning Be Secure? In: Proceedings of the ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS). (2006)
14. Pyxida: `http://pyxida.sourceforge.net`
15. PlanetLab: `http://planet-lab.org`
16. Selfish Neighbor Selection: `http://csr.bu.edu/sns`
17. Ledlie, J., Pietzuch, P., Seltzer, M.: Network coordinates in the wild. Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI) (2007)
18. Ledlie, J., Pietzuch, P., Seltzer, M.: Stable and accurate network coordinates. Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS) (2006)
19. Kaafar, M.A., Mathy, L., Turletti, T., Dabbous, W.: Real attacks on virtual networks: Vivaldi out of tune. Proceedings of the SIGCOMM workshop on Large-scale Attack Defense (2006)
20. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering **82**(Series D) (1960) 35–45
21. CommonSense: `http://www.kimvdlinde.com/professional/programming/statistics/commonSense/body.html`
22. Lua, E.K., Griffin, T., Pias, M., Zheng, H., Crowcroft, J.: On the Accuracy of Embeddings for Internet Coordinate Systems. In: Proceedings of ACM SIGCOMM-Usenix Internet Measurement Conference (IMC). (2005)
23. Sherr, M., Blaze, M., Loo, B.T.: Veracity: Practical Secure Network Coordinates via Vote-based Agreements. In: USENIX Annual Technical Conference. (2009)
24. Zhao, X., Lua, E.K., Chen, Y., Song, X., Deng, B., Li, X.: Sniper: Social-link Defense for Network Coordinate Systems. IEEE Conference on Computer Communications (INFOCOM) (poster) (2009)