Robust Decentralized Virtual Coordinate Systems in Adversarial Environments

DAVID ZAGE and CRISTINA NITA-ROTARU
Department of Computer Science and CERIAS, Purdue University
305 N. University St., West Lafayette, IN 47907 USA
{zagedj,crisn}@cs.purdue.edu

Virtual coordinate systems provide an accurate and efficient service that allows hosts on the Internet to determine the latency to arbitrary hosts without actively monitoring all of the nodes in the network. Many of the proposed systems were designed with the assumption that all of the nodes are altruistic. However, this assumption may be violated by compromised nodes acting maliciously to degrade the accuracy of the coordinate system. As numerous peer-to-peer applications come to rely on virtual coordinate systems to achieve good performance, it is critical to address the security of such systems.

In this work, we demonstrate the vulnerability of decentralized virtual coordinate systems to insider (or Byzantine) attacks. We propose techniques to make the coordinate assignment robust to malicious attackers without increasing the communication cost. We use both spatial and temporal correlations to perform context-sensitive outlier analysis to reject malicious updates and prevent unnecessary and erroneous adaptations. We demonstrate the attacks and mitigation techniques in the context of a well-known virtual coordinate system using simulations based on three representative, real-life Internet topologies of hosts and corresponding round trip times (RTT). We show the effects of the attacks and the utility of the mitigation techniques on the virtual coordinate system as seen by higher-level applications, elucidating the utility of deploying robust virtual coordinate systems as network services.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and System Design—*Network Topology*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms: Design, Reliability, Security

Additional Key Words and Phrases: Attack Mitigation, Network Coordinates, Reliability, Security, Virtual Coordinate Systems

1. INTRODUCTION

A wide range of applications taking advantage of peer-to-peer (P2P) systems have emerged in recent years, including file download and distribution (*e.g.*, BitTorrent), voice over IP (*e.g.*, Skype), and video broadcasting (*e.g.*, ESM [Chu et al. 2000]). Many of these applications optimize their performance based on network topology. For example, the construction of multicast trees or the selection of a replica for file sharing applications can be greatly improved by taking advantage of network locality [Pietzuch et al. 2006]. One basic approach to learn network attributes such as locality is to probe all of the hosts in the network. The cost associated with actively monitoring to estimate such attributes is non-negligible [Chu et al. 2000; Zhao et al. 2004], being exacerbated by the presence of multiple applications performing this task on a common network infrastructure.

Virtual coordinate systems [Francis et al. 2001; Ng and Zhang 2002; Rao et al. 2003; Tang and Crovella 2003; Pias et al. 2003; Ng and Zhang 2004; Dabek et al. 2004; Costa et al. 2004; Gummadi et al. 2002; Lim et al. 2003; Lehman and Lerman 2006] have been proposed as a low communication cost service to accurately predict latencies between arbitrary hosts in a network. These systems allow a node to map itself to a *virtual* coordinate based on a small number of actual network latency estimates to a subset of *reference nodes*. By comparing virtual coordinates, nodes can trivially estimate the latency between them.

Two main architectures for virtual coordinate systems have emerged: landmark-based and decentralized. Landmark - based systems rely on infrastructure components (such as a set of landmark servers) to predict the latency between any two hosts. The set of landmarks can be pre-determined [Francis et al. 2001; Ng and Zhang 2002; 2004] or randomly selected [Tang and Crovella 2003; Pias et al. 2003]. Decentralized virtual coordinate systems do not rely on explicitly designated infrastructure components, requiring any node in the system to act as a reference node. Examples of such systems include PIC [Costa et al. 2004], Vivaldi [Dabek et al. 2004], and PCoord [Lehman and Lerman 2004; 2006].

The accuracy and stability of virtual coordinate systems rely on the assumption that the reference set nodes on which the virtual coordinate computation relies on are altruistic and correctly participate in the system. Under this assumption, many of the proposed systems have been shown to be accurate, often achieving an overall latency prediction error of less than ten percent [Dabek et al. 2004; Lehman and Lerman 2006]. While this assumption may be ensured for landmark-based virtual coordinate systems by securing the small set of infrastructure nodes, it is not easily achieved for decentralized systems where any node can act as a reference node for other nodes in the system. As a result, decentralized virtual coordinate systems are vulnerable to insider attacks [Kaafar et al. 2006a; 2006b] conducted by attackers that infiltrate such systems or compromise some of their nodes.

Since virtual coordinate systems are designed to be network services providing latency estimation for a wide variety of P2P applications, such as distributed hash tables (DHTs) [Dabek et al. 2004] and routing [Ledlie et al. 2007], they are likely to be prime candidates for attack. In order to maintain upper-level application performance, it is critical that virtual coordinate systems are designed to be robust to attackers that influence the accuracy of the coordinates.

Previous work has focused little on mitigating the vulnerabilities of virtual coordinate systems or their effect on upper-level applications. Costa et al. [2004] proposed to use the triangle inequality to detect malicious nodes. The results based on synthetic networks show that the method does improve the accuracy of the PIC coordinate system in adversarial networks. However, as demonstrated by Zheng et al. [2005], Lua et al. [2005], and Ledlie et al. [2007], violations of the triangle equality are very frequent for real networks, resulting in the inaccuracy and fragility of such detection mechanisms even when deployed in non-adversarial networks. More recent solutions proposed by Kaafar et al. [2007] and Saucez et al. [2007] mitigate malicious activity assuming the use of trusted network components, which may be impractical in many systems.

In this paper, we study the vulnerability of decentralized virtual coordinate systems to insider attacks and propose mechanisms to make the accuracy of such systems resilient to attack. We provide a solution for mitigating attacks against virtual coordinate systems that is based on realistic assumptions about network topology and demonstrate its effectiveness using real-life Internet data sets. Our solution does not induce extra communication in the system or use trusted components, complying with the virtual coordinate system design goals of flexibility and low communication overhead. We also analyze the effect of the attacks and our solution on higher-level applications utilizing the virtual coordinate system for latency estimation. We summarize our key contributions:

—We classify three types of attacks against virtual coordinate systems based on their impact on the coordinates: *coordinate inflation, deflation*, and *oscillation*. The attacks are conducted by insiders that have infiltrated the system or compromised some of the nodes. The low-rate nature of the attacks (*i.e.*, they do not require the attacker to generate a noticeable amount of traffic) makes them difficult to detect while their epidemic nature makes them very dangerous, as a small number of attackers can significantly influence the accuracy of the system.

—We propose techniques to reduce incorrect coordinate mappings by using spatial and temporal correlations to perform context-sensitive outlier analysis. A key component of our solution is based on the observation that the behavior of attackers can be constrained by correlating dependent metrics.

—We demonstrate the impact of the attacks and the effectiveness of our defense mechanisms through p2psim [p2p] simulations, in the context of the well-studied Vivaldi virtual coordinate system [Dabek et al. 2004] using three representative real-world topologies of hosts and corresponding RTTs: King [Gummadi et al. 2002], Meridian [Wong et al. 2005], and AMP [AMP]. We find through analytical and empirical studies that a spatial threshold of 1.5 and a temporal threshold of 4.0 maintain a low system error under attack while incurring a low false positive rate. Our experiments also show that the method begins to degrade when a coalition of malicious nodes accounts for over 30% of a node's reference set.

—We demonstrate the effect of the attacks and our defense mechanisms on higher-level applications through p2psim [p2p] simulations of the Chord DHT [Stoica et al. 2003] utilizing the Vivaldi virtual coordinate system to provide latency estimations for the King topology [Gummadi et al. 2002]. We show that the higher-level application performance severely degrades as the number of attackers increases. We find through empirical studies that our technique using a spatial threshold of 1.5 and a temporal threshold of 4.0 is able to preserve the upper-level application performance, even when 30% of the network is malicious.

The rest of the paper is organized as follows: We provide an overview of decentralized virtual coordinate systems and attacks against them in Sec. 2. We propose mitigation mechanisms in Sec. 3. We present experimental results demonstrating the impact of the attacks and the effectiveness of our solutions in Sec. 4. We analyze the effect of the attacks and our solution on a distributed hash table utilizing the virtual coordinate system for latency estimation in Sec. 5. We discuss related work in Sec. 6 and conclude our work in Sec. 7.

2. ATTACKS AGAINST VIRTUAL COORDINATE SYSTEMS

In this section, we provide an overview of the main components of decentralized virtual coordinate systems, a description of a representative virtual coordinate system, Vivaldi, and describe how virtual coordinate systems can be exploited by attackers.

2.1 Decentralized Virtual Coordinate Systems

The design goal of decentralized virtual coordinate systems is to efficiently create and maintain a stable set of virtual coordinates that accurately predict the latency between nodes without using fixed infrastructure nodes. Although each specific virtual coordinate system differs in some details, most of them follow a common design. The most important characteristics that define decentralized coordinate systems are (1) the *reference or neighbor set*, (2) the *latency prediction mechanism*, and (3) the *error minimization technique*.

In a decentralized virtual coordinate system, each node calculates its coordinates based on the information obtained from a small set of nodes in the network, which we refer to as the *reference set*. There are several methods used to select the reference set. One of the most promising methods identifies a set of close and a set of distant network nodes and selects a random subset of each [Dabek et al. 2004; Costa et al. 2004]. Nodes may have different reference sets. Different systems use different sizes of the reference set due to the frequency of actual network measurements, the number of nodes queried per measurement interval, and the error minimization technique utilized. For example, Vivaldi uses a reference set size of 64 nodes [Kaafar et al. 2006a], PCoord uses 10 nodes [Lehman and Lerman 2004], and PIC uses 32 nodes [Costa et al. 2004].

Once a reference set has been selected, a node determines its coordinate based on a predefined *latency prediction mechanism*, such as the Euclidean distance. Each system typically maintains coordinates in a low dimensional (usually 2 to 8 dimensions) Euclidean space [Costa et al. 2004], an augmented Euclidean space [Dabek et al. 2004], or a non-Euclidean (*e.g.*, hyperbolic) space [Shavitt and Tankel 2004]. In general, it has been shown that none of the embedding spaces dominates the others in performance [Lumezanu and Spring 2006] and lower dimensionality Euclidean spaces are often sufficient [Dabek et al. 2004]. A node determines its location and then successively refines it by periodically querying nodes in its reference set. Queried nodes respond with metrics that can include local error, perceived system error, local coordinates, and RTT.

Virtual coordinate systems achieve accurate latency prediction through *error minimization techniques* of a chosen latency error function. Examples include:

—Generic multi-dimensional minimization designed to minimize a relative system error measure such as the logarithmic transformed error using techniques such as the downhill simplex method [Costa et al. 2004].

—Minimizing coordinate error by simulating Newtonian mechanics. Each node in the system is simulated as a particle influenced by the field force induced between

Algorithm 1: Vivaldi Coordinate Update

```
Input: Remote node observation tuple ($\langle x_j, e_j, RTT_{ij} \rangle$)
Output: Updated Node Coordinate
1 $w = e_i/(e_i + e_j)$;
2 $e_s = |\|x_i - x_j\| - RTT_{ij}|/RTT_{ij}$;
3 $\alpha = c_e \times w$;
4 $e_i = (\alpha \times e_s) + ((1 - \alpha) \times e_i)$;
5 $\delta = c_c \times w$;
6 $x_i = x_i + \delta \times (RTT_{ij} - \|x_i - x_j\|) \times u(x_i - x_j)$;
```

nodes. Each pair of particles (nodes) either pulls or repulses each other, thereby reducing the total system error [Shavitt and Tankel 2004].

—Minimizing system error by simulating spring relaxation, where the state of the springs at rest is the optimal embedding. The system minimizes the squared system latency estimation error by iteratively finding the low-energy point of the spring-based system [Dabek et al. 2004].

While each technique has benefits, systems based on multi-dimensional minimization are often slow to converge, sensitive to initial system conditions, and sensitive to high error measurements. Simulation techniques such as spring relaxation are computationally inexpensive, less sensitive to high error nodes, and more amenable to general decentralized system design.

In general, virtual coordinate systems achieve the overall goals of accuracy and stability while reducing traffic by as much as two orders of magnitude when compared with active monitoring to estimate RTT [Costa et al. 2004]. Systems such as Vivaldi [Dabek et al. 2004], PCoord [Lehman and Lerman 2006], and PIC [Costa et al. 2004] stabilize at an average system latency estimation error of ten milliseconds for large scale simulations and deployments.

2.2 Vivaldi Virtual Coordinate System Overview

Vivaldi is a fully decentralized virtual coordinate system which assigns each host synthetic coordinates in a multi-dimensional Euclidean coordinate space. Conceptually, Vivaldi attaches a spring between each pair of neighbor nodes, where the length of the spring is the estimated RTT between the nodes. By minimizing the tension on these springs across the entire network, the protocol minimizes the error in the system.

When a node joins the system, it establishes its reference set and initializes its coordinate to the origin of the Euclidean coordinate space. Each node i then periodically probes a reference set member j to obtain j's current coordinate and perceived error. As i receives the coordinate (x_j), the remote error estimate (e_j), and measured RTT from j, i uses this information to update its coordinate using Algorithm 1. First, node i calculates the reliability of the observation based on the local and remote error (line 1) and relative error of the observation tuple (line 2). Next, node i updates its local error (line 4) and the movement dampening factor (line 5). Finally, node i updates its coordinate in the last line of the algorithm. As the Vivaldi virtual coordinate system stabilizes, the average system error is on the order of ten milliseconds. Once the coordinate system has stabilized, the latency

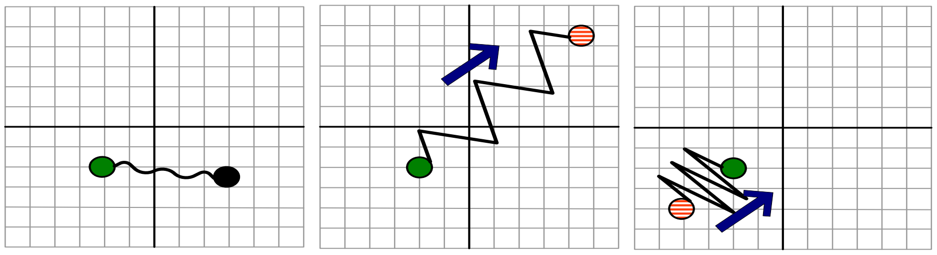

(a) No Attack
(b) Coordinate Inflation, Pull
(c) Coordinate Inflation, Push

Fig. 1. Example inflation attack scenarios against an individual victim node. Fig. 1(a) represents the system in a benign scenario where the tension on the logical spring connecting the two nodes is at a minimum and not inducing movement. Fig. 1(b) represents the coordinate inflation attack in which the malicious node is pulling the victim away from its correct location. Similarly, Fig. 1(c) represents the coordinate inflation attack in which the malicious node is pushing the victim away from its correct location.

between two nodes is estimated by computing the Euclidean distance between their coordinates. For further details of Vivaldi, we refer the reader to [Dabek et al. 2004].

2.3 Attacker Model

We consider a constrained - collusion Byzantine adversary model similar to that proposed by Castro et al. [2002], with a system size of N and a bounded percentage of malicious nodes f $(0 \leq f < 1)$. The malicious nodes behave arbitrarily and are only limited by the constraints of the cryptographic methods deployed [Dolev and Yao 1981]. The set of malicious nodes may collude. We assume a malicious adversary has access to all data at a node as any legitimate user would (insider access), including cryptographic keys stored at a node. This access can be the result of the adversary bypassing the authentication mechanisms or compromising a node through other means. As malicious nodes have insider access, nodes cannot be completely trusted although they are authenticated. We assume that data authentication and integrity mechanisms are deployed and we focus only on attacks directed at the accuracy of the virtual coordinates.

2.4 Attacks Description

As noted previously, the correct operation of virtual coordinate systems is dependent on the assumption that the reference set nodes are altruistic and respond with correct metrics to queries from any node computing its corresponding coordinates. An attacker controlling reference set nodes has the ability to influence the coordinate maintenance process by manipulating the information, such as remote node error and coordinates, returned in response to a query. By blindly accepting this malicious information, a correct node computes incorrect coordinates. While we examine the attacks in the context of the Vivaldi virtual coordinate system, the attacks generalize to any virtual coordinate system which uses external information to maintain the correctness of the system.

A malicious node is able to indirectly take advantage of the error minimization techniques and chosen error function by manipulating the metrics it reports as a reference set node. In doing so, an attacker is able to make a victim node move away from its correct location by either pulling it closer or pushing it towards a location specified by the attacker. As we can see from Algorithm 1, if the malicious node reports a low error (used to calculate the movement dampening factor in

Lines 1 and 5) in conjunction with an actual RTT less than the estimated RTT (the Euclidean distance between the victim node's coordinates and the reported location), the victim will be pulled towards the reported location and away from its correct location (Fig 1(b)). Similarly, if a malicious node reports a location close to the victim node that is in the opposite direction of the desired malicious location along with an artificially high RTT (caused by delaying probe responses) the node will be pushed towards the desired location (Fig 1(c)). The larger the induced delay, the farther the victim node will re-calculate its location away from the reported location. We refer to such attacks that result in coordinate mappings farther from the correct location as *coordinate inflation*.

An attacker may cause a victim node to remain immobile by reporting coordinates that minimize the difference between the actual and estimated RTT. Examining line 6 of Algorithm 1, if the attacker is able to force $(RTT_{ij} - \|x_i - x_j\|) = 0$, the victim node will remain stationary. To achieve this, the malicious node can either report coordinates which cause the difference between the measured and estimated RTT to be zero or influence the measured delay so it matches the delay estimated by the coordinates. In addition, since the difference between the actual and estimated RTTs is used to update the victim's local error estimate, the estimate will be artificially low. We refer to such attacks in which the victim nodes are prevented from performing necessary, correct coordinate changes as *coordinate deflation*.

The final type of attack we examine is the *coordinate oscillation* attack, which results in nodes continuously changing their locations and not converging to a stable coordinate. The attack consists of each malicious node attempting to maximize system error by randomly reporting erroneous coordinates, low, random local error, and actively delaying probe responses. Depending on the attack, each time a malicious node is used as a reference set node, it can report the same random coordinates or an entirely new location.

While we have described in detail each of the attacks against individual nodes, any attack against the coordinate system may target one or more nodes, a subset of nodes, or a region of the coordinate space. Also, as each node reports false information, it attempts to lie within the normal operating characteristics of the system. That is, the attacker reports coordinates selected over the coordinate area where the majority of nodes are located and it reports a low error in line with low-error benign nodes to minimize the risk of being identified as malicious.

The final goal of manipulating the coordinate system can include isolating subsets of nodes from the network, creating general disorder in the system, or rendering the coordinate system unusable due to high estimation error. While each of these goals serves a different purpose, in the end, they all distort the coordinate space and can make using the computed coordinates worse than using randomly assigned coordinates. Even short-lived, localized attacks have a long-lasting effect on the overall system. For example, even when a single victim node is displaced from its correct location, this has an epidemic, detrimental effect on the system as the victim node will push/pull other nodes away from their correct coordinates by reporting its now incorrect coordinates. That occurs since the victim node will serve as a reference set member for other nodes in the system, negatively influencing their coordinate computation. Besides degrading the accuracy of the coordinate system, the attacks will also adversely impact any application using the coordinate system to estimate network measurements. Additionally, as the attacks exploit the semantics of the information contained in the packet, they do not create noticeable changes in traffic load and thus are difficult to detect by traditional mechanisms.

3. LEVERAGING OUTLIER DETECTION TO ADD ROBUSTNESS TO VIRTUAL COORDINATE SYSTEMS

In this section, we discuss how techniques used in network security can be used in the context of virtual coordinate systems to make them more robust to attacks from compromised nodes. As such systems were proposed with the intention of decreasing the communication cost associated with active monitoring, our goal is to propose mitigation techniques that do not add any communication to the system. We propose to prevent incorrect coordinate updates by detecting and filtering out outliers in the metrics reported by queried nodes. Our method evaluates temporal and spatial correlations among data in the system. Below, we provide an overview of outlier detection and describe how we apply it to virtual coordinate systems.

3.1 Overview of Outlier Detection

The usability of a data set and the quality of statistical measures derived from it are integrally related to the number of outliers present. Outliers are data points which deviate so much from the rest of the data set as to arouse suspicion that they were generated by a different mechanism [Barnett and Lewis 1994]. The identification of outliers can lead to discovering important trends and information, such as the presence of malicious activity. Outlier detection, also known as anomaly or deviation detection, has been used in a variety of different fields including intrusion detection [Denning 1987; Sargor 1998], fraud detection [Ferdousi and Maeda 2006], medical analysis [Tan et al. 2006], and business trend analysis [Knorr and Ng 1998].

Many of the techniques for outlier detection utilize a statistical - based approach in which an outlier is any point which lies beyond a specified distance threshold. The Euclidean, Manhattan, Minkowski, and Mahalanobis distance functions are the most commonly used functions in determining distance [Tan et al. 2006; Birant and Kut 2006], each having its own benefits given the type of analysis being performed.

3.2 Applying Outlier Detection in Virtual Coordinate Systems

We leverage techniques from outlier detection to identify malicious behavior and take defensive actions to mitigate its effects. Instead of allowing malicious coordinate mappings to occur and then trying to detect them, we focus on reducing the likelihood of a node computing incorrect coordinates by filtering out malicious updates using statistical outlier detection. Each node independently performs outlier detection before updating its coordinates in order to identify and remove outliers in the received metrics.

Since the evidence of malicious activity is distributed across space and time, we detect malicious activity using both temporal and spatial correlations among metrics in the system. *Spatial outlier detection* identifies observations which are inconsistent with their surrounding neighbors, forcing nodes to report metrics consistent with what other reference peers are reporting. *Temporal outlier detection* identifies inconsistencies in the metrics over time, forcing a node to report metrics consistent with what it has reported in the past.

To minimize communication cost and maintain a low-overhead system, we focus on utilizing metrics used by nodes to update their coordinates using Algorithm 1. We use the 3-tuple of $\langle RTT,\, e_{remote},\, \Delta_{remote}\rangle$ to generate the spatial outlier statistics and the 5-tuple of $\langle RTT,\, e_{remote},\, \Delta_{remote},\, e_{local},\, \Delta_{local}\rangle$ to generate the temporal outlier statistics. Here, RTT is the measure of the actual two-way latency between the nodes, e_{remote} is the remote error estimate received in the observation tuple, and e_{local} is the most recent error estimate calculated at the local node using Algorithm 1. Δ_{remote} is the Euclidean distance the remote node has moved due to its last update. For example, if node j's location prior to an update was located at (4, 5) and moved to (4, 3) after the update, then Δ_{remote} is 2. Δ_{local} is the analogous measure at the local node. Both Δ_{remote} and Δ_{local} are the only metrics used in addition to the original Vivaldi update algorithm. Δ_{remote} is incorporated as part of the observation tuple and Δ_{local} is easily calculated at the local node.

Each metric was chosen on the basis that while each represents a different measure of system performance, changes in one measure will result in correlated changes in other metrics. For example, as the system stabilizes to low overall error, the local error reported by each node and magnitude of the change in coordinates will both decrease. An attacker must therefore report a high error with greatly changing coordinates in order to not be identified as malicious. Our solution also forces an attacker to lie consistently with other peers. This is difficult to achieve since an attacker does not have perfect knowledge of the observation space, must accurately predict the random subset of reference nodes that will be queried, and only has a finite amount of time to coordinate with other attackers.

Our approach uses the Mahalanobis [Wang and Stolfo 2004] distance to detect outliers. We selected this distance function because it has been shown to be effective at detecting outliers in data sets with multiple attributes [Lu et al. 2004], scales each variable based on its standard deviation and covariance, and takes into account how the measured attributes change in relation to each other [Walters et al. 2006].

3.2.1 *Spatial outlier detection.* We use spatial outlier detection to examine the consistency of recently received metrics from queried nodes. A node queries a random node from its reference set and receives an *observation tuple* which consists of $\langle RTT,\, e_{remote},\, \Delta_{remote}\rangle$. The node records this response and tracks the most recent u updates in a queue. The size of the history queue, u, is equal to the size of the reference set which allows the queue, on average, to contain one entry from each reference set nodes. Unlike more message-intensive distributed systems where a new set of responses from all nodes queried (in this case nodes in the reference set) are collected in response to one query [Chu et al. 2000], virtual coordinate systems collect these responses sequentially. Our approach requires a node to perform outlier detection every time it receives a new tuple, considering the most recent u updates. We note that this technique is an instance of spatial outlier detection since we examine metrics across various system nodes and not time.

Once a node receives an observation tuple, the node first computes the centroid of the data set consisting of observation tuples from the stored u updates. The node then computes the Mahalanobis distance between the received observation tuple and the centroid as follows [Wang and Stolfo 2004]:

$$d(a, \bar{b}) = \sqrt{((a - \bar{b})^T C^{-1} (a - \bar{b}))} \tag{1}$$

where a and $\bar{b}$ are two feature vectors whose elements consist of an *observation tuple*. a is the feature vector representing the newly received observation tuple from a remote node and $\bar{b}$ is the averaged feature vector (the centroid) computed from the u most recently received observation tuples which have been used to update the current node's coordinates. C^{-1} is the inverse sample covariance matrix computed from the u most recent tuples. After the distance is calculated, it is compared against a *spatial threshold*. We discuss spatial threshold selection in Sec. 4.3.

3.2.2 *Temporal outlier detection.* We use temporal correlations to detect inconsistencies in the metrics reported over time by a reference set node. We use the 5-tuple consisting of $\langle RTT,\ e_{remote},\ \Delta_{remote},\ e_{local},\ \Delta_{local} \rangle$. Using incremental learning, we compute a temporal centroid for each of the members of a node's reference set. We assume each of the reported metrics is statistically independent, necessitating the storage of just the mean, standard deviation, and sample count computed from the received query responses over time. The stored values for a reference set member are incrementally updated with metrics received from its query response, similar to Wang and Stolfo [2004], using the technique described by Knuth [1978]. In order to compare newly received values with the temporal centroid, we use the "simplified Mahalanobis distance" presented by Wang and Stolfo [2004]:

$$d(x, \bar{y}) = \sum_{i=0}^{n-1} \frac{|x_i - \bar{y_i}|}{\bar{\sigma_i} + \alpha} \tag{2}$$

where n is the number of metrics, five in our case (remote error, local error, latency, change in remote coordinates, and change in local coordinates), $\bar{\sigma_i}$ is the standard deviation, and α is a smoothing factor empirically set to .001 to help to avoid overfitting and reduce false positives [Wang and Stolfo 2004]. Once a query response is received, the latest observation tuple is compared with the corresponding temporal centroid using the simplified Mahalanobis distance, based on a *temporal threshold* that decides if the tuple is an outlier or not. We discuss temporal threshold selection in Sec. 4.3.

3.2.3 *Spatio-temporal outlier detection.* As seen in Algorithm 2, we combine the two outlier detection mechanisms described above by using a codebook technique similar to Jiang and Cybenko [2004]. As a node receives observation tuples from its reference set members, it checks each one to ensure that it is not a spatial or temporal outlier. If the reference node is found to be an outlier, the query response will not be used in future temporal centroid calculations since it will not be incorporated into the temporal mean, temporal standard deviation, or sample count. Also, it will not be used in future spatial centroid calculations since it will not be added to the queue of the most recent u updates. If the observation tuple is not an outlier, it is used to update the receiver node's coordinates using Algorithm 1.

Algorithm 2: Procedure to exclude malicious observation tuples from the coordinate update process.

```
Input: $x_j$, $RTT$, $e_{remote}$, $\Delta_{remote}$, $e_{local}$, $\Delta_{local}$
Output: Updated coordinate if the observation tuple was not malicious
// Calculate the centroid of the most recent tuples and check
   spatial consistency using the Mahalanobis distance
centroid = calcCentroid($u$ observation tuples);
if (spatialOutlier(centroid, $RTT$, $e_{remote}$, $\Delta_{remote}$) == true) then
    return false;
// Check temporal consistency using the simplified Mahalanobis
   distance
if (temporalOutlier($RTT$, $e_{remote}$, $\Delta_{remote}$, $e_{local}$, $\Delta_{local}$) == true) then
    return false;
// Data is not malicious, perform update operations using
   Algorithm 1
updateCoordinates($x_j$, $e_j$, $RTT_{ij}$);
store $\langle RTT, e_{remote}, \Delta_{remote}\rangle$ tuple in the queue of $u$ most recent tuples;
update stored temporal statics for $RTT$, $e_{remote}$, $\Delta_{remote}$, $e_{local}$, $\Delta_{local}$;
return true;
```

3.3 Potential Limitations of Outlier Detection

While outlier detection is an extraordinarily powerful tool, it has potential limitations that must be addressed. First, the system must start in a clean state free from attack, allowing the system to establish a correct centroid and correct running averages of historical data [Barreno et al. 2006]. Next, without a proper response to the attacks identified by outlier detection, the attack data may eventually become incorporated in the corpus of data used to create the view of normal system functionality, allowing the attacks to go unnoticed and rendering the system ineffective [Chan-Tin et al. 2009]. One possible response method is the removal of nodes identified as malicious through the use of a reputation or consensus protocol, such as that proposed by Walters et al. [2008]. Finally, an adversary can attempt to bypass the outlier detection by reporting metrics that are marginally incorrect [Chan-Tin et al. 2009]. However, there are several techniques that can be employed in conjunction with our solution to create solutions robust to attackers specifically targeting the defense mechanisms, such as using regularization to bias hypothesis testing, dynamically adapting thresholds of the system, introducing randomization into data collection and hypothesis calculations, and providing disinformation to potentially malicious nodes [Barreno et al. 2006].

4. EXPERIMENTAL RESULTS

In this section, we demonstrate the impact of attacks against virtual coordinate systems through simulations using actual Internet topologies. In addition, we demonstrate that our proposed mechanisms enhance the robustness of decentralized virtual coordinate systems to such attacks. We examine their effect on a representative decentralized virtual coordinate system, Vivaldi, which is simulated in the p2psim simulator [p2p]. We selected Vivaldi to demonstrate the attacks and defense mechanisms because it is a mature system, conceptually easy to understand and visualize, and has been shown to produce low error embeddings [Dabek et al. 2004].

4.1 Evaluation Methodology

We use three different RTT data sets collected from real-life Internet topologies. Tab. I and Fig. 2 summarize the characteristics of each data set:

– *King:* The King data set contains the pair-wise RTT of 1740 nodes measured using the King method [Gummadi et al. 2002].

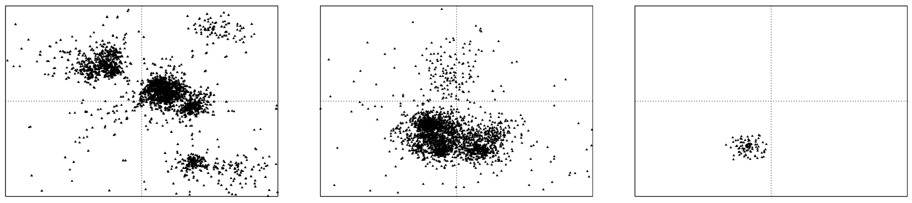
– *Meridian:* The Meridian data set, obtained from the Cornell Meridian project [Wong et al. 2005], contains the pair-wise RTT of 2500 nodes measured using the King method [Gummadi et al. 2002].

– *AMP:* The AMP data set, collected from the NLANR Active Measurement Project [AMP] on March 1, 2007, contains complete information for 90 high-speed nodes contained mostly in North America.

Table I. Data Sets Characteristics

Data Set	# Nodes	Avg. RTT	Max. RTT	Std. Dev. RTT
King	1740	180ms	800ms	66ms
Meridian	2500	80ms	1000ms	69ms
AMP	90	70ms	453ms	51ms

We selected the King and Meridian data sets because they are representative of larger scale P2P systems and were used in validating many virtual coordinate systems. They have very different data characteristics. The King data set contains a variety of link latencies, allowing nodes in the virtual coordinate system to form a structure in which nodes with small RTTs between them converge into clusters, as seen in Fig. 2(a). The average RTT of Meridian is approximately half that of King since it contains many nodes close to each other in terms of network latency. This can be seen visually in Fig. 2(b), where the system forms fewer, but larger clusters. The final data set, AMP, was used since it represents a smaller, high speed system, such as a corporate network. In AMP, 100ms or less links account for nearly 90% of all links, resulting in one main cluster, as seen in Fig. 2(c). We did not consider synthetic topologies since they do not capture important network properties such as violations of the triangle inequality.



(a) King (b) Meridian (c) AMP

Fig. 2. Node placement chosen by Vivaldi for various data sets

In order to quantitatively compare the effect of attacks on the accuracy of the system, we evaluate two error metrics:

• *System prediction error* is defined as

$$Error_{pred} = |RTT_{Act} - RTT_{Est}|$$

where the RTT_{Act} is the measured RTT between two nodes and RTT_{Est} is the RTT predicted by the virtual coordinate system. This metric provides an intuition of how the overall system is performing. The lower the system prediction error, the more accurate are the predicted RTTs.

• *Relative error* is defined as

$$Error_{rel} = \frac{Error_{attack}}{Error_{no\_attack}}$$

where $Error_{attack}$ is the system prediction error measured in the presence of malicious nodes and $Error_{no\_attack}$ is the system prediction error without malicious nodes. This metric captures the impact an attacker has on the coordinate system. A relative error greater than one indicates a degradation in accuracy and a value less than one indicates a better estimation accuracy than the baseline.

For each of the error measures, the 5^{th}, 50^{th}, and 95^{th} percentile error are analyzed. These values are obtained by selecting the corresponding entries from a sorted array of prediction error and are averaged over multiple simulation runs. Intuitively, the 5^{th} percentile represents low error nodes, the 50^{th} percentile corresponds to average or median error nodes, and the 95^{th} percentile represents high error nodes.

We ran each simulation for 200 time units, where each time unit is 500 seconds in length. The King data was set as our default topology unless otherwise noted. The nodes join in a flash-crowd scenario in which all nodes join simultaneously and are each initially placed at the origin of the logical coordinate space. All nodes that join the network are physically stationary and are present for the duration of the experiment. Each node proceeds independently of other nodes in the network and chooses a reference set of 64 nodes using the Vivaldi method where half of the nodes are selected as the closest nodes based on network latency and the rest are selected at random. All other Vivaldi parameters were initialized to the optimal values discussed by Dabek et al. [2004]. Each of the experiments utilizes a two-dimensional coordinate space $\{(x, y)|x, y \in [-300000, 300000]\}$. Every simulation was run ten times with the reported metrics averaged over all of the simulation. In each of the figures, only the first 50 time units are displayed since the virtual coordinate system has converged to a stable state beyond that point.

4.2 Attacks Against Distributed Virtual Coordinate Systems

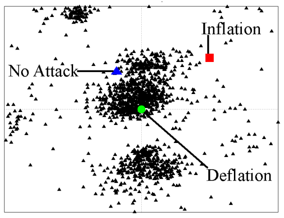
In this section, we demonstrate several attacks against the Vivaldi coordinate system. Vivaldi was designed to tolerate high-error, *benign* nodes, but it has no built-in mechanisms to defend against malicious nodes.

Inflation and deflation attacks. We first demonstrate how a coalition of f=30% malicious nodes can target one particular victim node and conduct an inflation or a deflation attack. Note that the actual number of attackers which directly influence the victim is the number of malicious nodes that are selected to be in the reference set of the victim node. Using the hypergeometric distribution, we can determine the probability of having a given number of malicious reference set members. If we let k represent the number of malicious nodes in a reference set, N be the number of nodes in the system, D is the total number of malicious nodes, and n is the size of the reference set, then the probability of having exactly k malicious nodes in a reference set is given by

$$f(k; N, D, n) = \frac{\binom{D}{k}\binom{N-D}{n-k}}{\binom{N}{n}} \tag{3}$$

By summing the discrete probability distributions for values from 0 to k, we can determine the probability of having a certain percentage of malicious nodes in the reference set. In the King data set, given that 30% of the total nodes are malicious, the probability that at least 30% of the nodes in a reference set (about 20 nodes) are also malicious is only about 35%.

Fig. 3 presents the location and associated prediction error of a single victim node under non-attack conditions and under the two attacks. The correct location of the victim node is represented by the triangle in quadrant II. Under a deflation attack, all of the attackers send the victim node coordinates that minimize the difference between the actual RTT and estimated RTT (minimizing the Euclidean distance between the attacker and victim). As a result, since the attackers force the estimated RTT to artificially match the actual RTT, the victim node remains stationary at the origin of the cartesian space while believing it has low estimation error. Fig. 3(a) also depicts an inflation attack, where all of the attackers send the victim node coordinates from a small chosen area along with RTTs influenced by delaying query responses, causing the node to move rapidly towards the desired area. The square representing the victim node was forced towards the area in quadrant I chosen by the attackers. As can be seen in the Tab. 3(b), the different attacks greatly increase the prediction error of the victim node from 10ms to 60ms for the deflation attack and 10ms to 70ms for the inflation attack.



(a) Node Placement

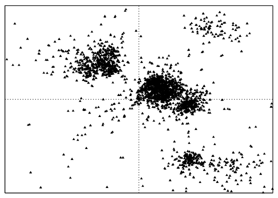
Attack	Pred. Error
None	10 ms
Deflation	60 ms
Inflation	70 ms
w/defense	11 ms

(b) Prediction Error

Fig. 3. Victim node error and placement for a deflation and inflation attack (King)

Oscillation attacks. We demonstrate an oscillation attack in Fig. 4. In this scenario, the malicious nodes work together to cause general disorder in the system by sending the victim nodes erroneous random coordinates selected over the coordinate space with a low error value, causing the victim nodes to make multiple incorrect coordinate changes. As seen in Fig. 4(a), the system under non-attack conditions has an easily identifiable structure in which nodes with small RTTs between them converge into clusters in the coordinate space. However, when the system is under attack as seen in Fig. 4(b), the virtual coordinate system loses its structure and its ability to yield a *low error* embedding. This attack also exemplifies the epidemic nature of such attacks. As correct nodes computing incorrect coordinates are later used as reference nodes for other nodes, the entire system destabilizes.

Impact of percentage of malicious nodes. We investigate the effect of the number of malicious nodes on the accuracy of the system by varying the percentage of malicious nodes. Each queried malicious node returns erroneous metrics in the form of a random location selected over the coordinates $\{(x, y)|x, y \in [-100000, 100000]\}$ and a low, non-zero error value. A malicious node also randomly delays its response between 100ms and 1000ms in order to induce greater variability in its responses in an attempt to expand the coordinate space.


(a) No attack

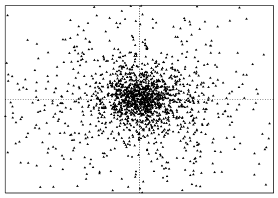
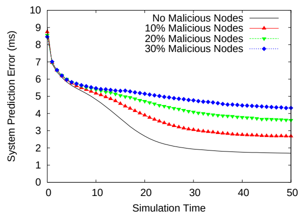
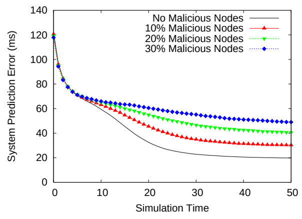

(b) Oscillation attack

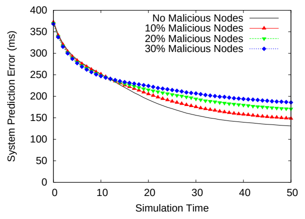
Fig. 4. Virtual coordinate system node placement under an oscillation attack (King)



(a) 5th Percentile Prediction Error

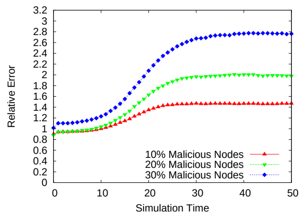


(b) 50th Percentile Prediction Error

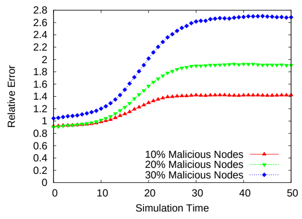


(c) 95th Percentile Prediction Error

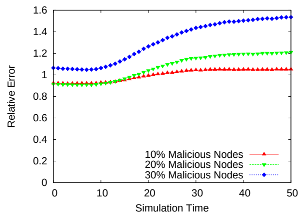
Fig. 5. System prediction error under different percentages of attackers (King)



(a) 5th Percentile Relative Error



(b) 50th Percentile Relative Error

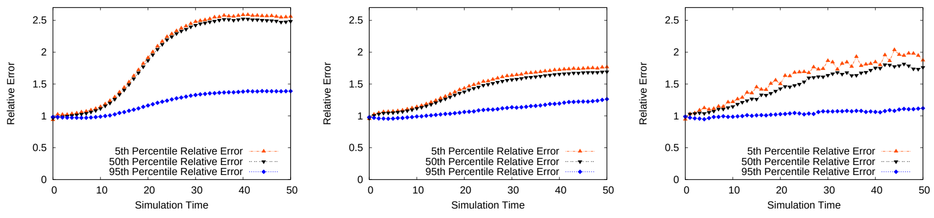


(c) 95th Percentile Relative Error

Fig. 6. Relative error under different percentages of attackers (King)

Fig. 5 presents the prediction error for the King data set for several percentages of malicious nodes. Under non-attack conditions, a node joining the coordinate system is initially placed at the origin of the logical coordinate space. As time passes, each node receives query responses from its reference set and is able to refine its location, allowing the system as a whole to achieve lower prediction error. Once the system stabilizes, the system prediction error remains roughly constant. After this point, each of the nodes continues to refine its location, but the overall sum of these movements yields little change in the prediction error. While the system under attack may initially start with similar prediction errors since nodes are initially placed at the origin, it is never able to effectively refine its coordinates and achieve the desired low estimation error found in the non-attack scenario. As the percentage of attackers increases, the ability of the system to accurately estimate latency significantly degrades.

Similar trends are also evident in Fig. 6, where the system can be seen to stabilize at a much higher relative error than the baseline of one. Having even a small percentage of attackers incurs *double* or *triple* the estimation error when compared with the non-malicious scenario. Malicious nodes have a greater negative impact on the lower error nodes, as can been seen from the higher relative errors in Fig. 6(a) and Fig. 6(b) than in Fig. 6(c). When a low error node moves in response to malicious data, it is prone to make large, erroneous changes to its own coordinates and experience a higher estimation error.



(a) King (b) Meridian (c) AMP

Fig. 7. Relative error under 30% malicious nodes for three real-life Internet latency data sets

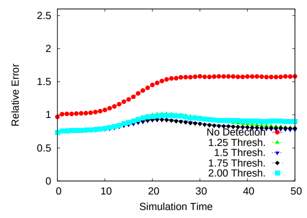
Impact of attacks on different network topologies. We examine the impact of the attacks on different network topologies with different sizes and variabilities by using three representative data sets. Fig. 7 shows the relative error for these data sets when f=30% of the nodes are malicious. Each of the topologies is adversely effected, with the King data set (Fig. 7(a)) showing the greatest degradation in accuracy due to the fact it has more variation in RTT and is prone to excessive over and under estimation in response to an attack. Meridian (Fig. 7(b)) shows less degradation due to the fact that it has less variation in its link latencies. AMP (Fig. 7(c)) shows more variability in the relative error due to its small size and frequent, large-scale node coordinate changes.

4.3 Threshold Selection for Spatial-Temporal Outlier Detection

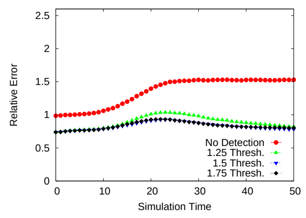
An important aspect of our approach is selecting the temporal and spatial thresholds that allow for the identification and elimination of potentially malicious query responses from the coordinate computation process. We consider the same attack scenario with a percentage of attackers as in Sec. 4.2 to experimentally determine our outlier detection thresholds since this scenario is one of the most difficult in which to identify malicious responses. When a malicious node selects a coordinate to respond with, this coordinate is selected from an area in which many altruistic nodes reside. The malicious nodes also report low but variable error inline with low-error altruistic nodes. These factors help disguise the malicious nodes actions and make them much harder to detect.

We use a slightly modified version of the method proposed in Sec. 3.2. Specifically, we do not use latency in the outlier detection due to the fact the latencies are predetermined in the simulator and thus show little variability.

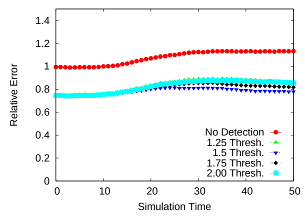
Temporal threshold selection. We used a threshold of 4.0 for our temporal outlier detection to allow for the four features: remote error, local error, change in remote coordinates, and change in local coordinates to vary at most one standard deviation over each feature from their temporally developed mean. This value was chosen based on the formula of the simplified Mahalanobis distance as discussed by Wang and Stolfo [2004].



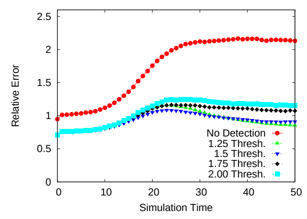
(a) 5th Pct. Relative Error, 10% Mal.



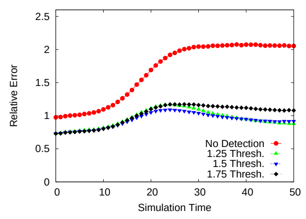
(b) 50th Pct. Relative Error, 10% Mal.



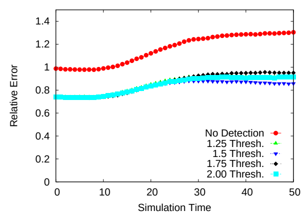
(c) 95th Pct. Relative Error, 10% Mal.



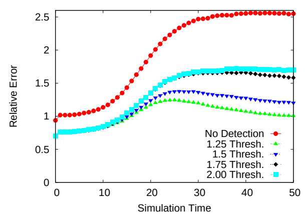
(d) 5th Pct. Relative Error, 20% Mal.



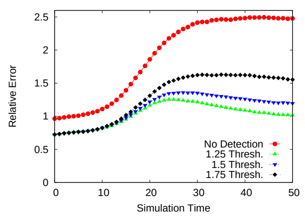
(e) 50th Pct. Relative Error, 20% Mal.



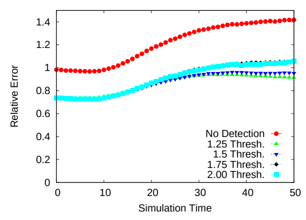
(f) 95th Pct. Relative Error, 20% Mal.



(g) 5th Pct. Relative Error, 30% Mal.



(h) 50th Pct. Relative Error, 30% Mal.



(i) 95th Pct. Relative Error, 30% Mal.

Fig. 8. Relative error under different percentage of attackers using different spatial thresholds (King)

Spatial threshold selection. The threshold for our spatial outlier detection can be mathematically derived as demonstrated by Smith and Cheeseman [1986] and Ribeiro [2004], assuming a multivariate Gaussian distribution for the metrics vector. The contours of equal probability of this distribution create a 2-dimensional ellipse and the outlier threshold reflects the probability of a vector being within the ellipse whose semi-axes are determined by k. The probability that a random vector lies within the ellipse increases with the size of k. Thus, for a given value of k the probability that a probed tuple lies within the ellipse can be computed as:

$$P = 1 - e^{\frac{-k^2}{2}} \tag{4}$$

We initially analytically selected a $k = 1.5$, in theory creating a threshold through which 53% of the coordinate updates would successfully pass. Through empirical testing of over 200,000 coordinate updates over multiple simulations, we found an ellipse determined by this threshold will allow approximately 79% of the updates to pass. This variation from the mathematically derived value can be attributed to the fact that the used metrics do not form a perfect normalized distribution and have a smaller variance than assumed in Equation 4. A node may select smaller spatial threshold values for stronger security guarantees, with the drawback that it may find its coordinate less accurate due to discarding valid updates.

Table II. False Positive Rate (Percentage) and Median Prediction Error for Different Spatial Thresholds (King)

% Mal.	***Spatial Threshold***			
Nodes	1.25	1.50	1.75	2.00
0	28, 16ms	21, 16ms	17, 16ms	13, 16ms
10	17, 17ms	13, 18ms	10, 19ms	5, 20ms
20	21, 18ms	15, 21ms	7, 23ms	6, 26ms
30	27, 20ms	11, 22ms	10, 33ms	9, 36ms

Fig. 8 presents the relative error for the King data set in which the temporal threshold was set to 4.0 and various spatial thresholds were tested. Tab. II presents the corresponding false positive rate and median system prediction error for the different thresholds. Although higher thresholds provide a smaller false positive rate, they do induce a higher error rate. For example, as malicious nodes are introduced into the system, a threshold of 2.00 maintains a lower false positive rate than other thresholds, but allows the prediction error to reach 36ms. This is 14ms more than the threshold of 1.5 which maintains a prediction error of 22ms when 30% of the nodes are malicious. We note that virtual coordinate systems are designed to be long-running services and hence the presence of a small percentage of false positive will not hinder the system. Based on the results in Fig. 8 and Tab. II we conclude that a spatial threshold of 1.5 works well for different percentages of attackers while having an acceptable false positive rate.

4.4 Mitigating Attacks Against Virtual Coordinate Systems

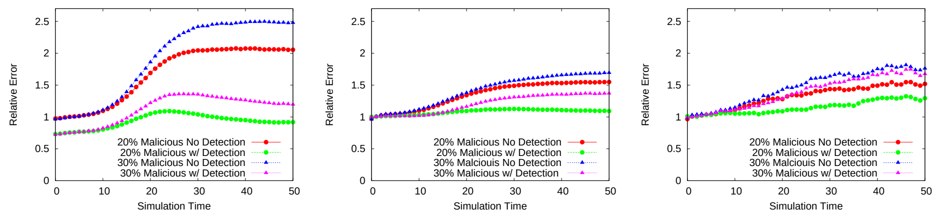
In this section, we demonstrate the effectiveness of our defense mechanisms at mitigating the effects of malicious nodes and sustaining the usability of the system.

Inflation and deflation attacks. We begin by reexamining the inflation and deflation attacks against a victim node, this time with a system using our defense mechanisms. The victim node is able to identify and mitigate the effect of the malicious nodes, achieving a prediction error of 11ms. The error is similar to a system under non-attack conditions (10ms), and nearly six times less than the unprotected system.

Different percentage of malicious nodes. Fig. 8 presents the relative error for the King data set for different percentages of malicious nodes. Note that for a spatial threshold of 1.5, our solution mitigates the system instability caused by the malicious nodes and even helps the system to stabilize at a more accurate local minimum than the initial protocol design to tolerate benign errors. While each node may occasionally accept erroneous data from malicious nodes due to a short temporal history or a skewed spatial history with updates from only a few nodes (as can be seen by the brief rise in error before coming back down), over time the system is able to avoid many malicious updates.

Different network topologies. Fig. 9 and Tab. III show the results for the King, Meridian and AMP topologies with and without outlier detection, where the attack scenario is the same as the coalition attack in Sec. 4.2. Applying the spatial threshold of 1.5 which was tested on the King data set, we find our solution is able to mitigate the system instability in all three data sets.

The King data set (Fig. 9(a)), previously seen in Fig. 8(e) and Fig. 8(h)), maintains a low relative error for various percentages of the attackers. We also note it is able to maintain a low system prediction error and low number of false positives (Tab. III). In Tab. III, the less the system prediction error increased with the number of attackers, the more resiliently the system performed under attack. Similar trends can also be observed for the Meridian data set (Fig. 9(b)). While our solution is able to offer protection from malicious nodes to the smaller scale AMP data set, it can be seen from Fig. 9(c) that larger percentages of malicious nodes begin to overwhelm the system. This occurs since the percentage of malicious nodes is high ($\geq$ 30%) and each benign node will have many malicious reference set members. For example, given that 30% of the total nodes are malicious, the probability that at least 30% of the nodes in a reference set of AMP are also malicious is about 67%. This is nearly double the probability of the King or Meridian data sets under the same conditions due to AMP's much smaller size.



(a) King (b) Meridian (c) AMP

Fig. 9. Relative error under different percentage of attackers using a spatial threshold of 1.5 with three real-life Internet latency data sets

Table III. False Positive Rate (Percentage) and Median Prediction Error for Different Data Sets Using A Spatial Threshold of 1.5

% Mal. Nodes	***Topology***		
	Meridian	AMP	King
0	23, 30ms	21, 18ms	21, 16ms
10	13, 30ms	15, 20ms	13, 18ms
20	12, 32ms	14, 25ms	15, 21ms
30	11, 40ms	12, 36ms	11, 22ms

Malicious coalition size tolerated by outlier detection. All defense mechanisms and protocols resilient to insiders have limitations regarding the number of attackers they can tolerate. We analyze the number of malicious colluding nodes that can be tolerated by our outlier detection mechanism using a reference set size of 64. Tab. IV presents the number of malicious nodes in a reference set which by colluding can influence the spatial centroid calculation enough to allow the attack types discussed in Sec. 2.4 to bypass the detection mechanism. Nearly twenty malicious nodes (or 30% of the reference set size) are required for nearly all of the identified attack types across the three data sets. The deflation attack is more successful for AMP since the RTTs are less variable and the virtual coordinate system creates one main cluster (Fig. 2(c)) that contains all of the nodes. This also explains why high percentages of malicious nodes ($\geq$ 30%) were able to overwhelm our solution in the AMP scenarios. In these cases, the benign nodes were likely to have twenty or more malicious nodes in their reference set, which could cause the spatial centroid to shift and allow malicious updates to pass undetected. We conclude that our defense method works well when the size of the malicious coalition is smaller than one third of the total number of nodes in the reference set.

Table IV. Number of Colluding Nodes Tolerated by Spatial Outlier Detection for Different Data Sets Using A Spatial Threshold of 1.5

Attack Type	***Data Set***		
	King	Meridian	AMP
Inflation	19.7	21.6	19.8
Deflation	20.2	19.8	12.6
Oscillation	19.6	20.3	19.3

System overhead. Our defense mechanism adds minimal link stress as it uses one additional numerical value in addition to the information already being exchanged between nodes. The memory utilization for spatial correlation requires maintaining the most recent u updates. In the case of the temporal outlier detection, the memory usage consists of maintaining the temporal centroid. By incrementally updating the centroid, we do not need to maintain the entire history for each probed node but only need to store the mean, standard deviation, and count for each of the metrics. The additional computational complexity is bounded by the number of nodes in the reference set which is constant.

5. ATTACK EFFECT ON A DHT USING THE VIVALDI VIRTUAL COORDINATE SYSTEM FOR LATENCY ESTIMATION

As the Internet continues to increase in popularity, an ever increasing number of distributed applications is being created and deployed. Many of these applications use network latency information to optimize their structure and performance. It has been shown in previous work that virtual coordinate systems can provide an efficient RTT estimation framework which can be leveraged by multiple systems across a network [Dabek et al. 2004]. In the following section, we analyze the effects malicious attacks against the Vivaldi virtual coordinate system have on an overlying application, the Chord distributed hash table. We demonstrate the decrease in the performance of Chord and how these effects can be mitigated by deploying a robust virtual coordinate system conferring resiliency to attack.

5.1 The Chord Distributed Hash Table

Distributed Hash Tables are a class of distributed application which provide the nodes in a network with the ability to efficiently store and locate <*key,data*> pairs. Over the past few years, DHTs have generated a great deal of research interest, resulting in the creation of systems such as Chord [Stoica et al. 2003], Kademlia [Maymounkov and Mazieres 2002], Pastry [Rowstron and Druschel 2001], and Tapestry [Zhao et al. 2004]. Along with the creation of multiple DHTs with differing properties, they are employed as building blocks in a variety of systems, ranging from file distribution protocols such as BitTorrent, to secure communications platform such as CSpace [CSP], to content distribution networks such as Coral [Freedman et al. 2004]. Given the general applicability of DHTs, these systems and the subcomponents on which they rely, including virtual coordinate systems such as Vivaldi, will increasingly be exposed to attack.

5.1.1 *General Chord Operation.* In a DHT, every node is assigned a n-bit identifier based on the hash of its IP address and every <*key,data*> pair is assigned a n-bit identifier based on the hash of the key. When a node joins the DHT, it is placed into a logical ring based upon its identifier, connected to the node with the pervious identifier and the next highest identifier (the predecessor and successor, respectively). For example, in Fig. 10, the predecessor of node 211 is 205 and the successor of node 211 is 230. In addition, a node with an identifier a maintains a table of neighbors known as a finger table, where the i^{th} finger points to the node closest to the identifier $a + 2^i$. These fingers can logically be thought of as "shortcuts" across the ring, allowing for shorter, more efficient lookup paths. As these values are assigned at join time and may change due to churn, a node will periodically run a maintenance protocol to ensure its predecessor, successor, and fingers are accurate and reachable.

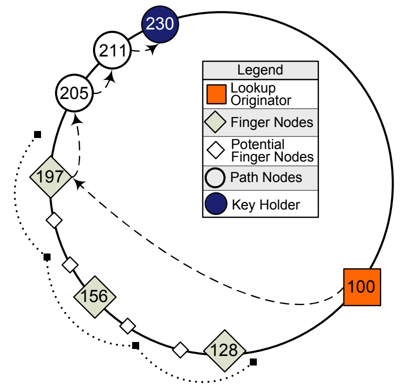


Fig. 10. An example lookup for the key 228 in the Chord DHT using an 8-bit ($2^8 = 256$) identifier space. The square represents the origin of the lookup while the large, filled diamond nodes are entries in its finger table. The smaller, unfilled diamonds represent possible finger choices in the identifier range denoted by the dotted arcs, which are both described in further detail in Sec. 5.1.2. The unfilled circles represent nodes on the lookup path and the final filled circle represents the node that holds the desired key k. The lookup path is denoted by the arrows.

When a <*key,data*> pair is added to the system, it is assigned to the first node with an identifier greater than or equal to the hash of the key k. When a client attempts to locate the key k in the DHT, it can query any node in the DHT to start the lookup process. In Fig. 10, node 100 has been queried for a key $k = 228$. The query will be iteratively forwarded through the finger and successor links around the ring until the query reaches the node 230 which holds the key k. For further details on the protocol, we refer the reader to [Stoica et al. 2003].

5.1.2 *Utilizing Latency Estimation to Improve Chord Performance.* In the original design of Chord and many ring-based DHTs, the selection of the node with which to fill the i^{th} position of the finger table of node a is restricted to the node closest to the identifier $a+2^i$. However, it has been realized that this "requirement" is not fundamental to the correct operation of the protocol and any node which falls in the range $[a + 2^i,\ a + 2^{i+1}]$ will satisfy the system requirements [Gummadi et al. 2003]. As seen in Fig. 10, the lookup originator has multiple choices for each finger node. This flexibility allows the system to utilize the latency estimation provided by virtual coordinate systems such as Vivaldi to optimize the finger selection, reducing the average lookup time by nearly one-half [Dabek et al. 2004].

When a node updates the i^{th} finger, it will check up to x nodes, where x is some small constant defined by the system, in the desired range of $[a + 2^i,\ a + 2^{i+1}]$ to determine which one has the lowest latency and will thus result in the best performance for lookups. It has been shown in previous work that $x = 16$ provides near optimal results [Dabek et al. 2004].

5.2 Experimental Evaluation

While the inclusion of the Vivaldi virtual coordinate system as a component of the Chord DHT can significantly improve the performance of the system, it also opens a new avenue of attack. If the latency estimations provided by the virtual coordinate system are not robust to malicious compromise, the operation of the DHT can be significantly impaired. In essence, the system is only as strong as its "weakest link". We demonstrate the impact attacks against the virtual coordinate system have on an overlying DHT and how these effects can be mitigated by adding robustness to the underlying virtual coordinate system.

5.2.1 *Evaluation Methodology.* In order to quantitatively compare the effect of attacks on the overlying DHT, we evaluate two system metrics:

• *Lookup Latency* is defined as the time required to find the address of the node holding a key k in the DHT and return the location to the requestor.

• *Normalized Lookup Delay* is defined as

$$Lat_{norm} = \frac{Lat_{attack}}{Lat_{no\_attack}}$$

where Lat_{attack} is the difference between the optimal and actual lookup latency measured in the presence of malicious nodes and Lat_{no_attack} is the difference between the optimal and actual lookup latency measured without malicious nodes. The optimal lookups occurs when nodes fill their finger table with the lowest latency nodes from the specified range, as discussed in Sec. 5.1.2. This metric captures the impact of malicious activity on the performance of the DHT. A normalized lookup delay greater than one indicates a degradation in performance (larger lookup times) while a value less than one indicates performance closer to optimal.

We ran each simulation for 200 time units, where each time unit is 500 seconds in length using the King data set topology unless otherwise noted. We present the results for the King data set as it is representative of larger scale P2P systems for which most DHTs are designed, the data set was used to validate several virtual coordinate system and DHT designs, and the results are representative of the results experienced with the other data sets. We enable the system to quickly form and stabilize by having the nodes join the DHT in a flash-crowd scenario in which all nodes join simultaneously, allowing us to focus on the lookup performance of the DHT. We used a finger table with the optimal size of $x = 16$, as shown by Dabek et al. [2004]. All other Chord parameters and were initialized to the optimal values discussed by Gummadi et al. [2003] and Dabek et al. [2004].

5.2.2 *Utilizing a Non-Robust Virtual Coordinate System for Latency Estimation.* We investigate the effects malicious nodes attacking the underlying Vivaldi virtual coordinate system have on the lookup latency of Chord by varying the percentage of malicious nodes using the attack described in Sec. 4.2.

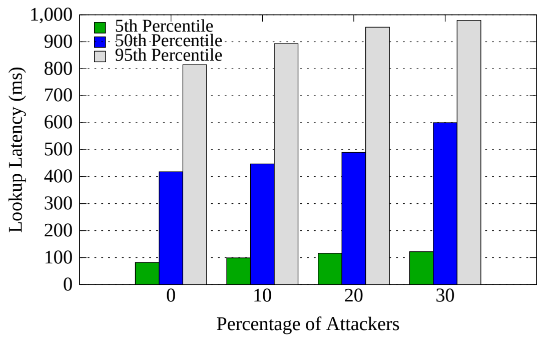


Fig. 11. Chord lookup latency under different percentages of attackers targeting the Vivaldi virtual coordinate system over the King topology

Fig. 11 presents the average lookup latencies for the Chord DHT over Vivaldi being attacked by several percentages of malicious nodes. As the number of malicious nodes increase, the average lookup performance of the system degrades. Under non-attack conditions, the DHT has an average lookup latency of 418ms. When 30% of the nodes in the network are attacking the virtual coordinate systems, the lookup latency increases to 600ms. In this case, an average lookup for the DHT utilizing a non-robust virtual coordinate system took nearly 50% longer when attackers were present. Lookups in the 5^{th} percentile are less affected as these lookups have shorter path lengths ($<$ 3 hops) while lookups in the average and 95^{th} percentile cases have longer paths ($\geq$ 5 hops), increasing the chance the paths will traverse a higher latency hop.

5.2.3 *Adding Robustness to the Virtual Coordinate System to Maintain DHT Performance.* In this subsection, we demonstrate the effectiveness of our defense mechanisms employed at the level of the Vivaldi virtual coordinate system at mitigating the effects of malicious nodes as seen by the higher-level application, the Chord DHT. Before presenting the results, we motivate the need to protect the virtual coordinate system subsystem and not just the higher level DHT.

If defense mechanism are only placed at the high-level application, both the detection and efficient response to attacks targeting specific subsystems become much more difficult. First, while attacking the virtual coordinate system has a significant impact on the lookup latency of Chord, other system performance metrics such as lookup success rate and hops per lookup are unaffected by these attacks. Under both benign and attack conditions, a random key lookup succeeded on average 97% of the time with an average path length of 5 hops, with almost no variation seen between different scenarios. This suggests that attacks against the underlying components produce little “visible” effect at the upper-level of the system, making them difficult to identify. Secondly, even if an attack is detected, the upper-level system is unable to take preventative actions against the malicious nodes since the system implicitly trusts the underlying components and is unable to determine which nodes are incorrectly using the virtual coordinate system. These reasons necessitate that the virtual coordinate system itself be robust to attack.

Fig. 12 presents the normalized lookup delay for different percentages of malicious nodes. As the underlying system is attacked, the ability of the overlying DHT to choose optimal finger nodes degrades, causing the normalized lookup delay to

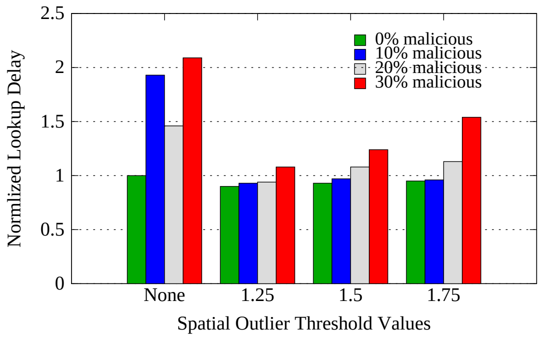


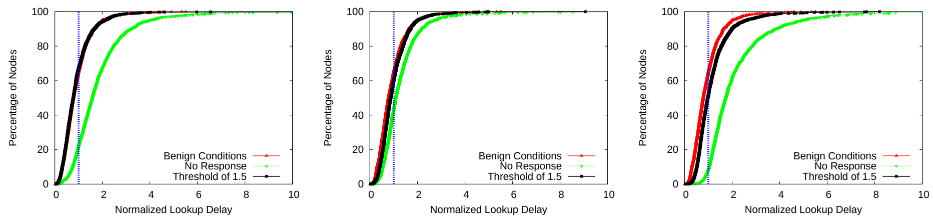
Fig. 12. Normalized lookup delay for Chord under different percentages of attackers

greatly increase. For example, with just 10% malicious nodes, the normalized lookup delay is 1.93 times greater than that of the system not under attack. A detailed explanation of the drop in attack strength for 20% malicious attackers is provided later in this section. Any user or application fetching data from the DHT will notice a marked increase in the average response time regardless of the exact percentage of malicious nodes, thus degrading the performance of the system and user experience. However, with the incorporation of a robust virtual coordinate system through the use of outlier detection, the effect of the attack on the DHT has been greatly decreased. For example, utilizing a spatial threshold of 1.5 and a temporal threshold of 4.0, the normalized lookup delay for a system experiencing attack by 10% of its nodes is .93, or 7% better than prior to our solution. As noted previously in Sec. 4.4, for smaller percentages of malicious nodes, our solution is able to reduce the error experienced in the virtual coordinate system. This allows the DHT to more accurately assess and choose the optimal finger nodes.

In order to determine if the decline in performance of the DHT under attack is due to a select group of high error nodes or a general system decline, we look at the effect of the attack on individual nodes in the network. We can see from the CDFs presented in Fig. 13, analogous to the increase in the average lookup latency error, as the number of attackers increases, the number of nodes able to maintain low lookup latency error decreases. For example, under non-attack conditions, 67% of the nodes have a lookup latency error of less than or equal to one (which indicates performance equal to or better than average) while over 95% have a lookup latency error less than 2. However, as seen in Fig. 13(a), with a network containing just 10% malicious nodes, only 22% of the nodes are able to maintain a lookup latency error of less than one and only 66% have a lookup latency error less than 2. Using a spatial threshold of 1.5 and a temporal threshold of 4.0, the system is able to return to functioning at non-attack scenario levels, with 66% of the nodes having a lookup latency error of less than one and over 94% having a lookup latency error less than 2.

For both Fig. 12 and Fig. 13, it at first appears that the attacks with 20% malicious nodes are detrimental but less effective than other percentages of malicious attackers. Upon further inspection, we determined that the random node movement caused by the attack forced a number of optimal finger node selections to have artificially low latency estimations and thus be chosen as fingers while many higher latency nodes (which would be avoided by the optimal selection) had markedly higher latency estimations. In other attack scenarios, these two cases were reversed, with many higher latency nodes having their latency estimation artificially reduced and thus being chosen as finger nodes. This artifact of the composition of the systems caused the normalized lookup delay in the 20% malicious nodes scenario to be lower than in the other attack percentages.

By using a robust virtual coordinate system to estimate network latency, the DHT is able to optimize its performance in *both* non-attack and attack scenarios.



(a) 10% Malicious (b) 20% Malicious (c) 30% Malicious

Fig. 13. Cumulative distribution functions of the normalized lookup delay for Chord under different percentages of malicious nodes in three different scenarios. The red upright triangle line plots the conditions experienced when *no* malicious nodes are present, the green downward triangle plots the conditions experienced when using a non-robust virtual coordinate system, and the black square line plots the conditions experienced when using a robust virtual coordinate system. The solid blue vertical line represents a normalized lookup delay of 1.

6. RELATED WORK

In this section, we provide an overview of previous research conducted in four areas related to our work: defense mechanisms in virtual coordinate systems, coordinate system error minimization, secure localization, and secure routing.

6.1 Defense Mechanisms in Virtual Coordinate Systems

Research has demonstrated the susceptibility of Vivaldi to attacks [Kaafar et al. 2006a; 2006b]. To address these vulnerabilities, there have been several proposed methods to maintain virtual coordinate system accuracy [Costa et al. 2004; Kaafar et al. 2007; Saucez et al. 2007; Sherr et al. 2008]. The PIC virtual coordinate system [Costa et al. 2004] uses a security test based on the triangle inequality in which any node that violates the triangle inequality above some margin of error is ignored and designated as malicious. However, it has been shown that RTT measurements often violate this inequality [Zheng et al. 2005; Lua et al. 2005; Ledlie et al. 2007] and thus solutions based solely on such inequalities may degrade system performance when no attack is occurring.

Recent work by Kaafar et al. [2007] utilizes a solution which employs a set of trusted nodes as a reference set by which to analyze all node behavior for malicious patterns and behavior. In a similar vein, the reputation-based work by Saucez et al. [2007] uses *a priori* trusted nodes to detect malicious nodes. The key difference between these techniques and our method is that we do not necessitate the need for trusted components in the network. The work by Sherr et al. [2008] uses a verification set of nodes for a node n, where n's update is considered malicious if a percentage of the verification set perceives the error of the update to be greater than a predetermined threshold. The main differences between this technique and our method is we do not require extra node sets nor network communication and we utilize outlier detection over multiple metrics. One possible research avenue to further increase the robustness of virtual coordinate systems is through the combination of techniques such as those suggested by Sherr et al. [2008] with our solution. Under such a system, each node would reach independent decisions which are then augmented using the group decisions reached by the verification set.

6.2 Coordinate System Error Minimization and Landmark Selection

An important area of research orthogonal to the security of virtual coordinate systems is the minimization of error in the systems. The accuracy of such systems is greatly effected by landmark placement for centralized schemes and neighbor selection in decentralized schemes. In the work by Zhang et al. [2006], it is shown that a hierarchical approach can lead to better performance over non-hierarchical solutions. Works by Lua et al. [2005], Zhang et al. [2006], and Narayanan and Shim [2007] demonstrate shortcomings of current systems and propose possible new metrics and measurements to more accurately embed the latency in the coordinate system. These areas provide interesting opportunities for further research since our work could possibly leverage these new metrics to place further constraints on the attackers and create a more robust, accurate, and fault-tolerant system.

6.3 Secure Localization in Sensor Networks

There have been many solutions proposed to secure localization in sensor networks. While these provide insight into securing virtual coordinate systems, the majority of the solutions rely on assumptions that are not applicable in the P2P domain. Defense mechanisms such as SPINE [Capkun and Hubaux 2005], LAD [Du et al. 2006], and TCSD [Chen et al. 2008] as well as those proposed by Lazos and Poovendran [2005] and Mathews et al. [2007] rely heavily on a pre-defined set of reference nodes to identify potentially malicious activity. ROPE [Lazos et al. 2005], SLA [Anjum et al. 2005], and the work by Mathews et al. [2007] require the reference nodes to be trusted. As decentralized virtual coordinate systems lack such infrastructure components, the proposed solutions are not applicable. Additionally, solutions such as ROPE [Lazos et al. 2005] and HiRLoc [Lazos and Poovendran 2006] rely on secure communication based on cryptography to mitigate outside attackers. Our defense mechanisms are designed to mitigate insider attacks not prevented by such schemes.

Recently, there has been growing interest in utilizing anomaly detection to mitigate attacks in sensor network localization [Srinivasan et al. 2006; Chen et al. 2008; Li et al. 2005]. DRBTS [Srinivasan et al. 2006] creates a distributed reputation-based trust protocol to detect malicious reference nodes and TCSD [Chen et al. 2008] detects temporal and spatial inconsistences in reported data caused by malicious tampering. Based on the broadcast nature of the wireless medium, both solutions require nodes to overhear each other which is not possible in our environment. Li et al. [2005] utilize statistical methods to secure sensor localization. The authors developed an attack resilient location estimator based on Least Median of Squares (LMS) designed to tolerate malicious nodes. While the work by Li et al. [2005] holds similarities to ours in that it filters out outliers in the range estimates to improve sensor location estimates, our technique utilizes multiple, correlated attributes over the observation space and over time to improve system performance.

6.4 Secure Routing

Our research has ties to secure wired and wireless routing. A large number of solutions have been proposed to secure wired routing, with particular focus on securing interdomain routing [White 2003; Hu et al. 2004; Oorschot et al. 2007; Lad et al. 2006; Hu and Mao 2007; Zheng et al. 2007]. Many of these solution rely solely on the use of cryptographic constructions [White 2003; Hu et al. 2004; Oorschot et al. 2007]. These techniques are often expensive in terms of complexity and do not offer protection from malicious insider who have compromised a machine and have access to all information on it, including the cryptographic keys.

Previous work in wireless routing has examined defending against malicious insiders [Hu et al. 2005; Papadimitratos and Haas 2003; Marti et al. 2000; Awerbuch et al. 2005]. Ariadne [Hu et al. 2005] and SDT [Papadimitratos and Haas 2003] utilize multiple routes through the network to prevent malicious nodes from dropping data. Watchdog [Marti et al. 2000] relies on properties of the wireless medium to overhear packets sent between nodes to ensure nodes are functioning correctly. ODBSR [Awerbuch et al. 2005] identifies malicious links through the use of probes and acknowledgments along the faulty path. These works use different environmental assumptions (a broadcast medium) and have much higher communication overhead than Vivaldi or our proposed solution.

Other research has examined the use of anomaly detection to identify and avoid suspicious routing behavior [Lad et al. 2006; Hu and Mao 2007; Zheng et al. 2007; Huang and Lee 2004; Patwardhan et al. 2005]. PHAS [Lad et al. 2006] allows the owners of routing prefixes to collect information about current and past BGP routing updates and check the validity of anomalous route updates based on expected patterns, similar to signature-based anomaly detection. Hu and Mao [2007] combine anomaly detection based on control-plane information with host fingerprints to detect route hijacking attempts. Zheng et al. [2007] utilize path monitors to determine if the current routing path is within some threshold distance of the expected routing path through the network. Huang and Lee [2004] and Patwardhan et al. [2005] detect malicious actions in wireless networks based on deviances from predefined patterns. Our work differs in the fact it does not require extra infrastructure components and is based on statistical anomaly detection which does not require specific sequences of events to detect malicious activity. Additionally, we use a variant of statistical anomaly detection that utilizes multiple, correlated attributes and is not based on differences determined by finite automata.

7. CONCLUSION

In this paper, we studied attacks against the accuracy of virtual coordinate systems. We classified the attacks as coordinate inflation, deflation, and oscillation and showed that a small number of attackers can severely degrade coordinate accuracy due to the epidemic nature of the attacks. We proposed the use of spatial-temporal correlations to perform outlier detection to reject malicious updates and prevent unnecessary adaptations. By using analytical and empirical methods, we found that a spatial threshold of 1.5 and a temporal threshold of 4.0 maintains a low system error under attack while incurring a low false positive rate. We examined the limitations of outlier detection when a significant percentage of nodes are malicious and found that the method begins to degrade when more than 30% of the nodes in a reference set form a malicious coalition. Finally, we demonstrate the effects of the attacks as seen by a DHT using the coordinate system for latency estimation and the utility of deploying robust virtual coordinate systems as network services.

REFERENCES

Cspace. `http://cspace.in/`.

Nlanr active measurement project. `http://amp.nlanr.net/`.

p2psim: A simulator for peer-to-peer protocols. `http://pdos.csail.mit.edu/p2psim/`.

Anjum, F., Pandey, S., and Agrawal, P. 2005. Secure localization in sensor networks using transmission range variation. In *Proc. of MASS*.

Awerbuch, B., Curtmola, R., Holmer, D., Rubens, H., and Nita-Rotaru, C. 2005. On the survivability of routing protocols in ad hoc wireless networks. In *Proc. of SecureComm*.

Barnett, V. and Lewis, T. 1994. *Outliers in statistical data*. John Wiley & Sons New York.

Barreno, M., Nelson, B., Sears, R., Joseph, A. D., and Tygar, J. D. 2006. Can machine learning be secure? In *Proc. of ASIACCS*.

Birant, D. and Kut, A. 2006. Spatio-temporal outlier detection in large databases. In *Proc. of ITI*.

Capkun, S. and Hubaux, J.-P. 2005. Secure positioning of wireless devices with application to sensor networks. In *Proc. of INFOCOM*.

Castro, M., Druschel, P., Ganesh, A., Rowstron, A., and Wallach, D. S. 2002. Secure routing for structured peer-to-peer overlay networks. In *Proc. of ACM OSDI*.

Chan-Tin, E., Feldman, D., Hopper, N., and Kim, Y. 2009. The Frog-Boiling Attack: Limitations of Anomaly Detection for Secure Network Coordinate Systems. In *Proc. of SecureComm*.

Chen, H., Lou, W., Ma, J., and Wang, Z. 2008. Tscd: A novel secure localization approach for wireless sensor networks. In *Proc. of SENSORCOMM*.

Chu, Y., Rao, S. G., and Zhang, H. 2000. A case for end system multicast (keynote address). In *Proc. of SIGMETRICS*.

Costa, M., Castro, M., Rowstron, R., and Key, P. 2004. PIC: practical Internet coordinates for distance estimation. In *Proc. of ICDCS*.

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

Dabek, F., Li, J., Sit, E., Robertson, J., Kaashoek, M. F., and Morris, R. 2004. Designing a dht for low latency and high throughput. In *Proc. of USENIX NSDI*.

Denning, D. E. 1987. An intrusion-detection model. *IEEE Trans. Softw. Eng. 13*, 222–232.

Dolev, D. and Yao, A. C. 1981. On the security of public key protocols. In *Proc. of SFCS*.

Du, W., Fang, L., and Ning, P. 2006. Lad: localization anomaly detection for wireless sensor networks. *J. Parallel Distrib. Comput. 66*, 874–886.

Ferdousi, Z. and Maeda, A. 2006. Unsupervised outlier detection in time series data. In *Proc. of ICDEW*.

Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. 2001. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Trans. Netw. 9*, 525.

Freedman, M. J., Freudenthal, E., and Mazières, D. 2004. Democratizing content publication with coral. In *Proc. of USENIX NSDI*.

Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., and Stoica, I. 2003. The impact of DHT routing geometry on resilience and proximity. In *Proc. of SIGCOMM*.

Gummadi, K. P., Saroiu, S., and Gribble, S. D. 2002. King: Estimating latency between arbitrary internet end hosts. In *Proc. of SIGCOMM-IMW*.

Hu, X. and Mao, Z. M. 2007. Accurate real-time identification of ip prefix hijacking. In *S&P*.

Hu, Y.-C., Perrig, A., and Johnson, D. B. 2005. Ariadne: a secure on-demand routing protocol for ad hoc networks. *Wirel. Netw. 11*, 21–38.

Hu, Y.-C., Perrig, A., and Sirbu, M. 2004. Spv: secure path vector routing for securing bgp. *SIGCOMM Comput. Commun. Rev. 34*, 179–192.

Huang, Y. and Lee, W. 2004. Attack analysis and detection for ad hoc routing protocols. *Lecture Notes in Comput. Sci. 3224*, 125–145.

Jiang, G. and Cybenko, G. 2004. Temporal and spatial distributed event correlation for network security. In *Proc. of ACC*.

KAAFAR, M. A., MATHY, L., SALAMATIAN, C. B. K., TURLETTI, T., AND DABBOUS, W. 2007. Securing internet coordinate embedding systems. In *Proc. of SIGCOMM*.

KAAFAR, M. A., MATHY, L., TURLETTI, T., AND DABBOUS, W. 2006a. Real attacks on virtual networks: Vivaldi out of tune. In *Proc. of LSAD*.

KAAFAR, M. A., MATHY, L., TURLETTI, T., AND DABBOUS, W. 2006b. Virtual networks under attack: Disrupting internet coordinate systems. In *Proc. of CoNext*.

KNORR, E. M. AND NG, R. T. 1998. Algorithms for mining distance-based outliers in large datasets. In *Proc. of VLDB*.

KNUTH, D. E. 1978. *The Art of Computer Programming, 2nd Ed. (Addison-Wesley Series in Computer Science and Information*. Addison-Wesley Longman Publishing Co., Inc.

LAD, M., MASSEY, D., PEI, D., WU, Y., ZHANG, B., AND ZHANG, L. 2006. PHAS: A prefix hijack alert system. In *Proc. USENIX Security*.

LAZOS, L. AND POOVENDRAN, R. 2005. Serloc: Robust localization for wireless sensor networks. *ACM Trans. Sen. Netw. 1*, 73–100.

LAZOS, L. AND POOVENDRAN, R. 2006. Hirloc: high-resolution robust localization for wireless sensor networks. *Selected Areas in Communications, IEEE Journal on 24*, 233–246.

LAZOS, L., POOVENDRAN, R., AND ČAPKUN, S. 2005. Rope: robust position estimation in wireless sensor networks. In *Proc. of IPSN*.

LEDLIE, J., GARDNER, P., AND SELTZER, M. 2007. Network coordinates in the wild. In *Proc. of USENIX NSDI*.

LEDLIE, J., PIETZUCH, P., MITZENMACHER, M., AND SELTZER, M. 2007. Wired geometric routing. In *Proc. of IPTPS*.

LEHMAN, L. AND LERMAN, S. 2004. Pcoord: Network position estimation using peer-to-peer measurements. In *Proc. of NCA*.

LEHMAN, L. AND LERMAN, S. 2006. A decentralized network coordinate system for robust internet distance. In *Proc. of ITNG*.

LI, Z., TRAPPE, W., ZHANG, Y., AND NATH, B. 2005. Robust statistical methods for securing wireless localization in sensor networks. In *Proc. of IPSN*.

LIM, H., HOU, J., AND CHOI, C. 2003. Constructing internet coordinate system based on delay measurement. In *Proc. of IMC*.

LU, C., CHEN, D., AND KOU, Y. 2004. Multivariate spatial outlier detection. *International Journal on Artificial Intelligence Tools, World Scientific 13*, 801–812.

LUA, E., GRIFFIN, T., PIAS, M., ZHENG, H., AND CROWCROFT, J. 2005. On the accuracy of embeddings for internet coordinate systems. In *Proc. of IMC*.

LUMEZANU, C. AND SPRING, N. 2006. Playing Vivaldi in Hyperbolic Space. In *Proc. of IMC*.

MARTI, S., GIULI, T. J., LAI, K., AND BAKER, M. 2000. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of MobiCom*.

MATHEWS, M., SONG, M., SHETTY, S., AND MCKENZIE, R. 2007. Detecting compromised nodes in wireless sensor networks. In *Proc. of ACIS SNPD*.

MAYMOUNKOV, P. AND MAZIERES, D. 2002. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. of IPTPS*.

NARAYANAN, S. AND SHIM, E. 2007. Performance improvement of a distributed internet coordinates system. In *Proc. of CCNC*.

NG, E. AND ZHANG, H. 2002. Predicting internet network distance with coordinates-based approaches. In *Proc. of INFOCOM*.

NG, T. AND ZHANG, H. 2004. A network positioning system for the internet. In *Proc. of USENIX*.

OORSCHOT, P. V., WAN, T., AND KRANAKIS, E. 2007. On interdomain routing security and pretty secure bgp (psbgp). *ACM Trans. Inf. Syst. Secur. 10*, 11.

PAPADIMITRATOS, P. AND HAAS, Z. J. 2003. Secure data transmission in mobile ad hoc networks. In *Proc. of WiSe*.

PATWARDHAN, A., PARKER, J., JOSHI, A., IORGA, M., KARYGIANNIS, T., AND UMBC, B. 2005. Secure routing and intrusion detection in ad hoc networks. In *Proc. of PerCom*.

Pias, M., Crowcroft, J., Wilbur, S., Bhatti, S., and Harris, T. 2003. Lighthouses for scalable distributed location. In *Proc. of IPTPS*.

Pietzuch, P., Ledlie, J., Mitzenmacher, M., and Seltzer, M. 2006. Network-aware overlays with network coordinates. In *Proc. of ICDCS*.

Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., and Stoica, I. 2003. Geographic routing without location information. In *Proc. of MobiCom*.

Ribeiro, M. I. 2004. Gaussian probability density functions: Properties and error characterization. Tech. Rep. 1049-001, Instituto Superior Tcnico, Lisboa, Portugal.

Rowstron, A. and Druschel, P. 2001. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Lecture Notes in Comput. Sci. 2218*, 329–350.

Sargor, C. 1998. Statistical anomaly detection for link-state routing protocols. In *Proc. of ICNP*.

Saucez, D., Donnet, B., and Bonaventure, O. 2007. A Reputation-Based Approach for Securing Vivaldi Embedding System. *Lecture Notes in Comput. Sci. 4606*, 78.

Shavitt, Y. and Tankel, T. 2004. Big-bang simulation for embedding network distances in euclidean space. *IEEE/ACM Trans. Netw. 12*, 993–1006.

Sherr, M., Loo, B., and Blaze, M. 2008. Veracity: A fully decentralized service for securing network coordinate systems. In *Proc. of IPTPS*.

Smith, R. C. and Cheeseman, P. 1986. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research 5*, 56–68.

Srinivasan, A., Teitelbaum, J., and Wu, J. 2006. Drbts: Distributed reputation-based beacon trust system. In *Proc. of DASC*.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. 2003. Chord: A scalable peer-to-peer lookup service for internet applications. *IEEE/ACM Trans. Netw. 11*, 17–32.

Tan, P.-N., Steinbach, M., and Kumar, V. 2006. *Introduction to Data Mining*. Addison Wesley.

Tang, L. and Crovella, M. 2003. Virtual landmarks for the internet. In *Proc. of SIGCOMM*.

Walters, A., Zage, D., and Nita-Rotaru, C. 2006. Mitigating attacks against measurement-based adaptation mechanisms in unstructured multicast overlay networks. In *Proc. of ICNP*.

Walters, A., Zage, D., and Nita-Rotaru, C. 2008. A framework for securing measurement-based adaptation mechanisms in unstructured multicast overlay networks. *IEEE/ACM Trans. Netw. 16*, 1434–1446.

Wang, K. and Stolfo, S. J. 2004. Anomalous Payload-based Network Intrusion Detection. In *Proc. of RAID*.

White, R. 2003. Securing BGP through secure origin BGP (soBGP). *Business Communications Review 33*, 47–53.

Wong, B., Slivkins, A., and Sirer, E. 2005. Meridian: a lightweight network location service without virtual coordinates. In *Proc. of SIGCOMM*.

Zhang, R., Hu, C., Lin, X., and Fahmy, S. 2006. A hierarchical approach to internet distance prediction. In *Proc. of ICDCS*.

Zhang, R., Tang, C., Hu, Y., Fahmy, S., and Lin, X. 2006. Impact of the Inaccuracy of Distance Prediction Algorithms on Internet Applications – An Analytical and Comparative Study. In *Proc. of INFOCOM*.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. 2004. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *Selected Areas in Communications, IEEE Journal on 22*, 41–53.

Zheng, C., Ji, L., Pei, D., Wang, J., and Francis, P. 2007. A light-weight distributed scheme for detecting ip prefix hijacks in real-time. *SIGCOMM Comput. Commun. Rev. 37*, 277–288.

Zheng, H., Lua, E., Pias, M., and Griffin, T. 2005. Internet routing policies and round-trip-times. In *Proc. of PAM*.

Received Dec. 2008; revised Oct. 2009; accepted Dec. 2009