2007

On the Accuracy of Decentralized Virtual Coordinate Systems in Adversarial Networks

David Zage

Cristina Nita-Rotaru
Purdue University, crisn@cs.purdue.edu

Report Number:
07-012

ON THE ACCURACY OF DECENTRALIZED VIRTUAL COORDINATE SYSTEMS IN ADVERSARIAL NETWORKS

David Zage
Cristina Nita-Rotaru

Department of Computer Science
Purdue University
West Lafayette, IN 47907

CSD TR #07-012
May 2007

On the Accuracy of Decentralized Virtual Coordinate Systems in Adversarial Networks

ABSTRACT

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 nodes in the network. Many of the proposed virtual coordinate systems were designed with the assumption that all of the nodes in the system 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 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 demonstrate the attacks and mitigation techniques in the context of a well-known distributed virtual coordinate system using simulations based on three representative, real-life Internet topologies of hosts and corresponding round trip times (RTT).

1. INTRODUCTION

A wide range of applications taking advantage of peer-to-peer systems have emerged in recent years, including file download and distribution (e.g. BitTorrent [1], Emule [2]), voice over IP (e.g. Skype [3]), and video broadcasting (e.g. ESM [4], Coolstreaming [5]). 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. One basic approach to learn network locality is to probe all hosts in the network to determine attributes such as latency. The cost associated with active monitoring to estimate such attributes is non-negligible [4, 6], being exacerbated by the presence of multiple applications performing this task on a common network infrastructure.

Virtual coordinate systems [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17] 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 distance estimates to a subset of *reference nodes.* By comparing the 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 distance between any two hosts. The set of landmarks can be pre-determined [7, 11, 16] or randomly selected [9, 17]. 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 decentralized virtual coordinate systems include PIC [12], Vivaldi [10], and PCoord [15, 18].

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 [19] 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 [10, 15]. 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 [20, 21] conducted by attackers that infiltrate such systems or compromise some of their nodes. Since virtual coordinate systems are network services providing support for a wide variety of peer-to-peer applications and more recently routing [22], they would likely be a prime candidate for attack. It is critical that such systems are designed to be robust to attackers that influence the accuracy of the coordinates.

Previous work focused very little on mitigating vulnerabilities of virtual coordinate systems with the notable exception of [12], which uses the triangle inequality to detect malicious nodes. The results based on synthetic networks presented in [12] show that the method does improve the accuracy of the PIC coordinate system in adversarial networks. However, as shown in [23, 24, 25] violations of the triangle equality are very frequent for real networks, resulting in the inaccuracy and fragility of virtual coordinate systems even when deployed in non-adversarial networks. Previous work [20, 21] also pointed out the susceptibility of Vivaldi to attacks, without proposing any solution.

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 attacks. To the best of our knowledge, we provide the first 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 increase the communication in the system, complying with the virtual coordinate system design goal of maintaining a low communication cost. We summarize our key contributions:

• We classify attacks against virtual coordinate systems, based on the impact on the coordinates, as *coordinate inflation, deflation*, and *oscillation*. The attacks are conducted by insiders that infiltrated the virtual coordinate 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 the attacker can be constrained by correlating dependent metrics.

• We demonstrate the impact of the attacks and the effectiveness of our defense mechanisms through p2psim [26] simulations, in the context of the well-studied Vivaldi virtual coordinate system[10] using three representative real-world topologies of hosts and corresponding RTTs: King [13], Meridian [27], and AMP [28]. We found through analytical and empirical studies that a spatial threshold of 1.5 and a temporal threshold of 4.0 provided a low system error under attack while maintaining an acceptable false positive rate. Our experiments also show that the method starts to degrade when the coalition size of malicious nodes in the reference set of a node increases over 30% of the reference set size.

The rest of the paper is organized as follows: We provide an overview of decentralized virtual coordinate systems and attacks against them in Section 2. We propose mitigation mechanisms in Section 3. We present experimental results demonstrating the impact of the attacks and the effectiveness of our solutions in Section 4. We discuss related work in Section 5 and conclude our work in Section 6.

2. ATTACKS AGAINST VIRTUAL COORDINATE SYSTEMS

In this section, we give an overview of the main components of decentralized virtual coordinate systems and describe how they can be exploited by attackers to influence their accuracy.

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 a decentralized coordinate systems are (1) the *reference or neighbor set*, (2) the *distance 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, with identifying a set of close and set of distant network nodes and selecting a random subset of each being one of the most promising [10, 12]. 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 [20], PCoord uses 10 nodes [18], and PIC uses 32 nodes [12].

Once a reference set has been selected, a node determines its coordinate based on a predefined *distance prediction mechanism*, such as the Euclidean distance. Each system typically maintains coordinates in either low dimensional (usually 2 to 8 dimensions) Euclidean space [12], an augmented Euclidean space [10], or non-Euclidean (e.g. hyperbolic) space [29]. In general, it has been shown that none of the embedding spaces dominates the others in performance [30] and lower dimensionality Euclidean spaces are often sufficient [10]. A node determines its position 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 provide accurate latency prediction, achieved through *error minimization techniques* of a chosen distance error function. Examples include:

• Generic multi-dimensional minimization designed to minimize a relative system error measure (such as logarithmic transformed error) using techniques such as the downhill simplex method [12].

• Minimizing coordinates by simulating Newtonian mechanics. Each node in the system is simulated as a particle influenced by the field force induced between nodes. Each pair of particles (nodes) either pulls or repulses each other, thereby reducing the total system error [29].

• Minimizing coordinates by simulating spring relaxation, where the state of the springs at rest is the optimal embedding. The system minimizes the squared system error by iteratively finding the low-energy point of the spring-based system [10].

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 [12]. Systems such as

Vivaldi [10], PCoord [15], and PIC [12] stabilize at an average system latency estimation error of ten milliseconds for large scale simulations and deployments.

2.2 Attacker Model

We consider a constrained-collusion Byzantine adversary model similar to that proposed in [31], with a system size of N and a bounded percentage of malicious nodes f $(0 \leq f < 1)$ behaving arbitrarily. 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. 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.

2.3 Attacks Description

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 any query 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.

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 position by either pushing the node away from or pulling it closer to the malicious node's reported coordinates. For example, a malicious node can attract a victim node towards a random position and away from the victim's correct position by reporting false virtual coordinates and a low error. Also, since many of the minimization techniques rely on the measured RTT of queries, a malicious node can push a victim node away from itself by delaying its query responses. The larger the induced delay, the farther the victim node will re-calculate its positions away from the malicious node's reported coordinates to possibly more erroneous locations. An attacker may also take advantage of the error minimization techniques to repel a victim node away from specific virtual coordinates by making its queried responses appear worse than actuality by advertising coordinates with high error. 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 positions similar to the current position of that victim node. A malicious node may also report false coordinates where the distance between the victim and the attacker reflects the RRT between the nodes, once again rendering the victim immobile. We refer to such attacks in which the victim nodes are prevented from performing necessary, correct coordinate changes as *coordinate deflation.*

Any attack against the coordinate system may target a particular node, subset of nodes, or region of the coordinate space. The final goal of manipulating the coordinate system can include isolating subsets of nodes from the network, creating general disorder in the system, and rendering the coordinate system unusable due to high estimation error. We refer to attacks which result in nodes not converging to a virtual coordinate and continuously changing their positions as *coordinate oscillation.*

While all of the attacks have different goals, 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 position, this has an epidemic, detrimental effect on many of the nodes in the system as the victim node will push/pull nodes away from their correct coordinates by reporting its now incorrect coordinates. That is because a correct node that computed its coordinates based on incorrect information may serve as a reference set for other nodes in the system, thus 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. In addition, as the attacks exploit the semantics of the information contained on the packet, they do not add a noticeable change 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 to decrease the communication cost involved in 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 [32, 33]. The identification of outliers can lead to discovering important trends and information, such as the presence of malicious activities. Outlier detection, also known as anomaly or deviation detection, has been used in a variety of different fields including intrusion detection [34, 35], fraud detection [36], medical analysis [37], and business trend analysis [38].

Many of the techniques for outlier detection utilize a statistical based or distance-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 [37, 39], each having its own benefits given the type of analysis being performed.

Malicious activity can lead to spatial and temporal inconsistencies. *Spatial outlier detection* identifies observations which are inconsistent with their surrounding neighbors, while *temporal outlier detection* identifies inconsistencies in the metrics of the observation space of a system over time. The use of both temporal and spatial outlier detection allows for the identification of multiple types of attacks with better accuracy than either alone.

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 through the use of statistical outlier detection. Since the evidence of malicious activity is distributed across space and time, we propose to detect them using both temporal and spatial correlations among metrics in the system.

Each node independently performs outlier detection before changing its coordinate in order to identify and filter out outliers in the received metrics. Spatial outlier detection compares the recently received metrics from each of the queried nodes in a node's reference set and forces a node to report metrics consistent with what other reference peers are currently reporting. Temporal outlier detection examines the consistency of the metrics received from an individual queried node over time and forces a node to report metrics consistent with what it has reported in the past.

To avoid adding communication cost, we use metrics already reported by the nodes in the reference set. We use the 3-tuple of <*remote error, change in remote coordinates, latency*> to generate the spatial outlier statistics and the 5-tuple of <*remote error, local error, latency, change in remote coordinates, change in local coordinates*> to generate the temporal outlier statistics. The metrics were chosen on the basis that while each of them represents a different measure of system performance, changes in one measure will result in a correlated change in other metrics. For example, as the system stabilizes to low overall error, the local error reported by each node and correlated magnitude of the change in coordinates will both change less. 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 as 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 [40] distance to detect outliers. We selected this distance function because it has been shown effective at detecting outliers with multiple attributes [41], scales each variable based on its standard deviation and covariance, and takes into account how the measured attributes change in relation to each other [42].

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 <*remote error, change in remote coordinates, latency*>. The node records this response and tracks the most recent u updates in a queue-like fashion, where the oldest responses are replaced by newer ones and u is equal to the size of the reference set. 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 [4], 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 highlight 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 [40]:

$$d(\vec{x},\vec{y}) = \sqrt{((\vec{x}-\vec{y})^T C^{-1}(\vec{x}-\vec{y}))} \tag{1}$$

where $\vec{x}$ and $\vec{y}$ are the feature vectors consisting of error, latency, and distance from the last virtual coordinate. $\vec{x}$ is the value from the query response and $\vec{y}$ is the average value that was calculated. C^{-1} is the inverse covariance matrix computed from the stored observation tuples. Finally, this distance 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 tuple consisting of <*remote error, local error, latency, change in remote coordinates, change in local coordinates*>. 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 the metrics received from that member's query response, similar to [40], using the technique described in [43]. In order to compare newly received values with the temporal centroid, we use the "simplified Mahalanobis distance" presented in [40]:

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

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 over-fitting and reduce false positives [40]. 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*

We combine the two outlier detection mechanisms described above by using a codebook technique similar to [44]. Each reference set node response that is not a spatial or temporal outlier is utilized in updating the receiver node's coordinates. 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 be dropped from the most recent u updates.

4. EXPERIMENTAL RESULTS

In this section we demonstrate the impact of attacks against virtual coordinate systems through simulations based on real-life 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 [10], which is simulated in the p2psim simulator [26]. 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 [10].

4.1 Evaluation Methodology

We use three different RTT data sets collected from real-life Internet topologies. Table 1 and Fig.1 summarize the characteristics of each data set. The data sets are:

- *King:* The King data set contains the pair-wise RTT of 1740 nodes measured using the King method [13].
- *Meridian:* The Meridian data set, obtained from the Cornell Meridian project [27], contains the pair-wise RTT of 2500 nodes measured using the King method [13].
- *AMP:* The AMP data set, collected from the NLANR Active Measurement Project [28] on March 1, 2007, contains complete information for 90 high-speed nodes contained mostly in North America.

Table 1: 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

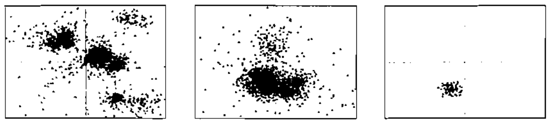

(a) King (b) Meridian (c) AMP

Figure 1: Node placement chosen by Vivaldi for various data sets

We selected the King and Meridian data sets because they are representative of larger scale peer-to-peer 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. 1(a). The average RTT of Meridian is approximately half that of King since it contains many nodes a short distance from one another, as seen in Fig. 1(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. 1(c). We do not consider synthetic topologies since they do not capture important network properties such as violations of the triangle inequality.

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

$$Error_{pred} = |Act_{RTT} - Est_{RTT}| \tag{3}$$

where the Act_{RTT} is the actual measured RTT and Est_{RTT} is the predicted RTT by the virtual coordinate system. This metric provides an intuition of how the overall system is performing. The lower the system prediction error is, the more accurate the predicted RTTs are.
Relative error is defined as

$$Error_{rel} = \frac{Error_{attack}}{Error_{no\_attack}} \tag{4}$$

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.

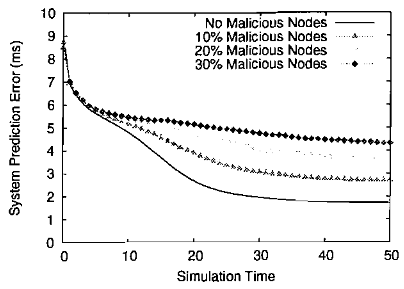
For each of the error measures, the 5^{th}, 50^{th}, and 95^{th} percentile error are analyzed. These values are obtained by selecting the corresponding entries from a sorted array of prediction error and are averaged over multiple simulation runs. Intuitively, the 5^{th} percentile represents low error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

We ran one million tick long simulations, using the King data 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. 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 such as the adaptive timestep were initialized to the optimal values discussed in [10]. 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 average over all of the simulation.

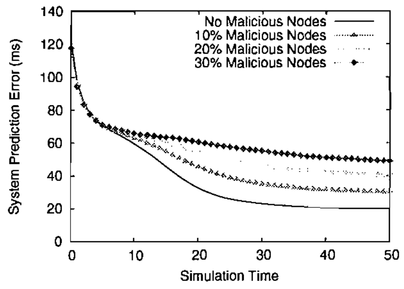
4.2 Attacks Against Distributed Virtual Coordinate Systems

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.

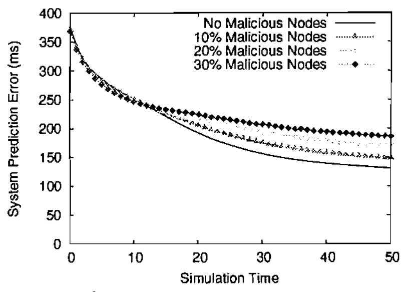
Inflation and deflation attacks. We first demonstrate



(a) 5th Percentile Prediction Error

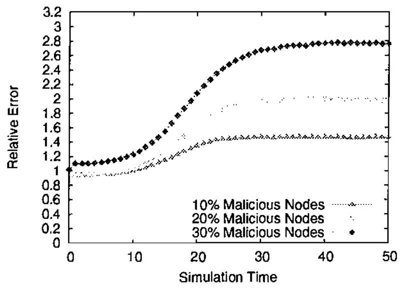


(b) 50th Percentile Prediction Error

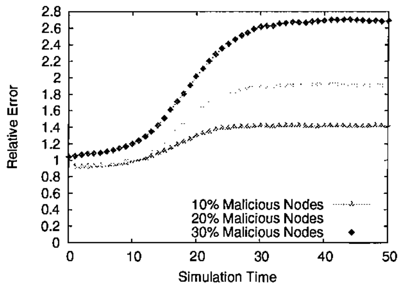


(c) 95th Percentile Prediction Error

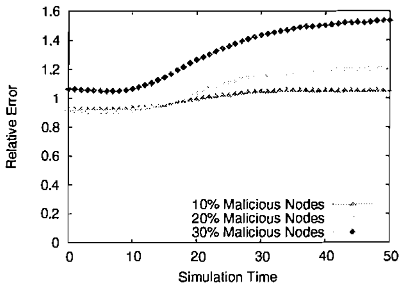
Figure 2: System prediction error under different percentages of attackers (King)



(a) 5th Percentile Relative Error



(b) 50th Percentile Relative Error



(c) 95th Percentile Relative Error

Figure 3: Relative error under different percentages of attackers (King)

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 represents the number of malicious nodes in a reference set, N be 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

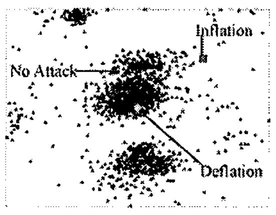
$$f(k; N, D, n) = \frac{\binom{D}{k}\binom{N-D}{n-k}}{\binom{N}{n}} \qquad (5)$$

By summing the discrete probability distributions for given values of k, we can determine the probability of having a certain percentage of malicious nodes in 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. 4 presents the location and associated prediction error of a victim node under non-attack conditions and under the two attacks. The correct location of the victim node is in the upper left quadrant. For the deflation attack, note the circle at the origin representing a victim node which did not move to its correct position. In this scenario, the attackers send the victim node coordinates that minimize the difference between the actual RTT and estimated RTT (the Euclidean distance between the attacker and victim). As a result, the victim stays at its current coordinate while believing it has a very low estimation error. Fig. 4(b) also depicts an inflation attack, where the attackers send the victim node chosen coordinates along with an artificially high RTT by delaying query responses. Note the square in the upper right quadrant representing the victim node forced to move away from the origin and towards a location chosen by the attacker. As it can be seen in the Table 4(a), the attacks greatly increase the prediction error of the victim node from 10ms to 60ms for the deflation attack and to 70ms for the inflation attack.

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

(a) Prediction Error

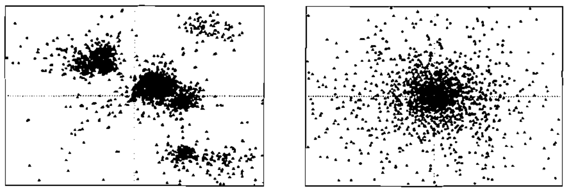


(b) Node Placement

Figure 4: Victim node error and placement for a deflation and inflation attack.

Oscillation attacks. We demonstrate an oscillation attack in Fig. 5. In this scenario, the attacker sends the victim nodes erroneous random positions selected over the coordinate space with a low error value, causing the victim nodes to make multiple incorrect coordinate changes. As it can be seen in Fig. 5(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. When the system is under attack as seen in Fig. 5(b), the virtual coordinate system looses its structure and hence also looses 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.



(a) No attack (b) Oscillation attack

Figure 5: Virtual coordinate system node placement under an oscillation attack.

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 position 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.

Fig. 2 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 position, allowing the system as a whole to achieve lower prediction error. Once the system stabilizes about halfway through the simulation, the system prediction error remains roughly constant. After this point, each of the nodes continues to refine its position, 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. 3, 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. 3(a) and Fig. 3(b) than in Fig. 3(c). When a low error node moves in response to malicious data, it is prone to make large, erroneous changes to its own position and experience a higher estimation error.

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. 6 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. 6(a)) showing the greatest degradation in accuracy due to the fact its has more variation in RTT and is prone to excessive over and under estimation in response to an attack. Meridian (Fig. 6(b)) shows less degradation due to the fact it has less variation in its link latencies. AMP (Fig. 6(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 to identify the potentially malicious query responses and eliminate them from the coordinate computation process. We consider the same attack scenario with a percentage of attackers as in Section 4.2 to experimentally determine our outlier detection thresholds since it is one of the most difficult in which to identify malicious responses. When a malicious node selects a coordinate to respond with, the selected coordinate is from a range in which many altruistic nodes reside as well as the majority of malicious nodes actual coordinates lie within. 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 Section 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.

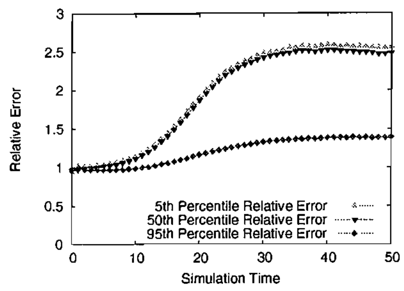
Temporal threshold selection. We used a threshold of 4.0 for our temporal outlier detection to allow for the four features: remote error, local error, change in remote coordinates, and change in local coordinates to vary by at most one standard deviation over each feature from their temporally developed mean. The value was chosen based on the formula of the simplified Mahalanobis distance as in [40].

Spatial threshold selection. The threshold for our outlier detection can be mathematically derived as in [45, 46], 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:

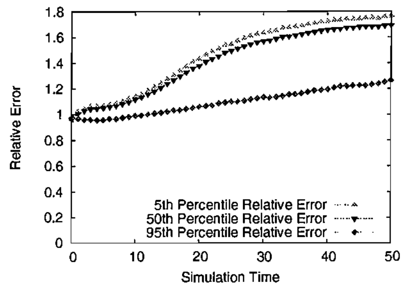
$$P = -\frac{1}{\sqrt{2}\pi} + 2\left(\frac{1}{\sqrt{2\pi}}\int_0^k e^{\frac{y^2}{2}}dy\right) - \sqrt{\frac{2}{\pi}}ke^{\frac{-k^2}{2}} \quad (6)$$

We initially analytically selected a k of 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 6. A node may select smaller spatial threshold values for stronger security guarantees, with the drawback that it may find its coordinate less accurate to discarding valid updates.

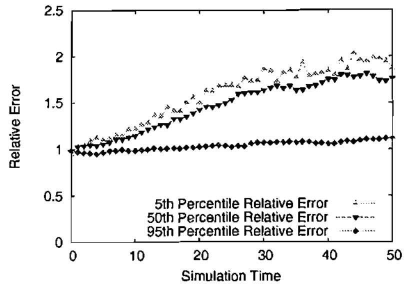
Fig. 7 presents the relative error for the King data set in which the temporal outlier threshold was set to 4.0 and various spatial outlier detection threshold were tested. Table 2 presents corresponding false positive rate and median system prediction error for the different thresholds. Although



(a) King



(b) Meridian



(c) AMP

Figure 6: Relative error under 30 percent malicious nodes for three real-life Internet latency data sets

Table 2: False Positive Rate (Percentage) and Median Prediction Error for Different Spatial Outlier Thresholds (King data set)

% Mal.	*Spatial Outlier 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

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 low false positive rate with the trade-offs that the prediction error raises to 36ms, with 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 service and hence the presence of a small percentage of false positive will not hinder the system. Based on the results in Fig. 7 and Table 2 we conclude that a spatial threshold of 1.5 worked 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 re-examining 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, as shown in Fig. 4. The error is similar to a system under non-attack conditions (10ms), and nearly six times less than the unprotected system.

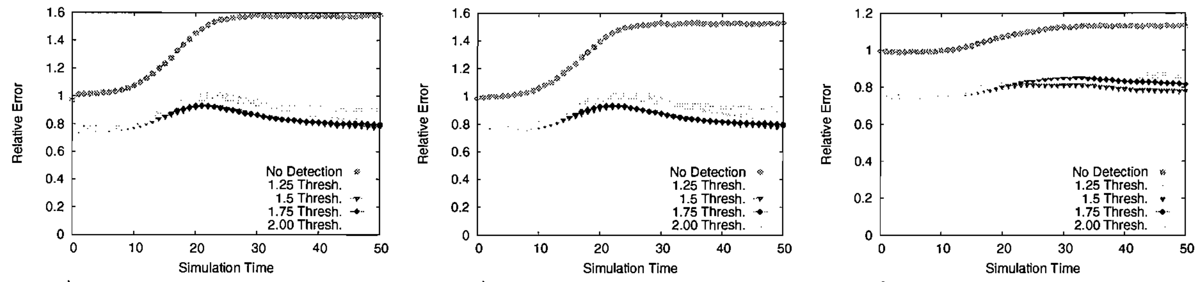
Different percentage of malicious nodes. Fig. 7 presents the relative error for the King data set for different percentages of malicious nodes. Note that for a spatial threshold of 1.5, our solution mitigates the system instability caused by the malicious nodes and even helps the system to stabilize 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.

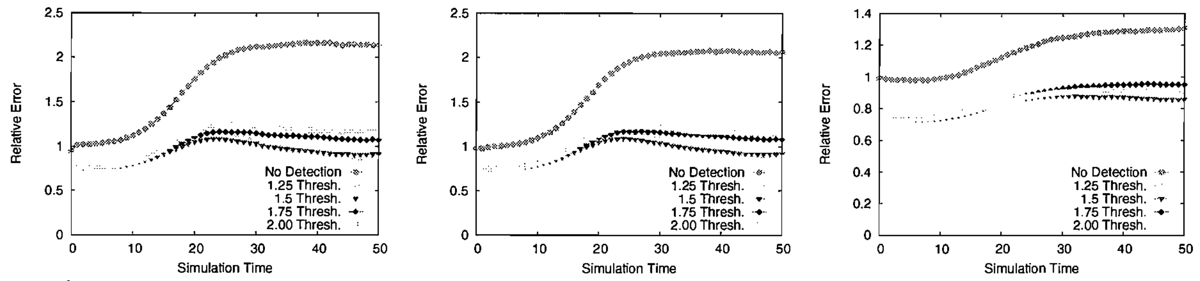
Table 3: False Positive Rate (Percentage) and Median Prediction Error for Different Data Sets Using A Spatial Outlier Threshold of 1.5

% Mal.	*Topology*		
Nodes	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

Different network topologies. Fig. 8 and Table 3 show the results for the King, Meridian and AMP topologies with and without outlier detection, where the attack scenario is the same as the one in Section 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. 8(a)) 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 (Table 3). In Table 3, 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. 8(b)). While our solution is able to offer protection to the smaller scale AMP data set from malicious nodes, it can be seen from Fig. 8(c) that larger percentages of malicious begin to overwhelm the system. This occurs since the percentage of malicious nodes is high ($\geq$ 30%), 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 for King or Meridian under the same conditions due to AMP's much smaller size (see Table 1).

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. Table 4 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 Section 2.3 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


(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.

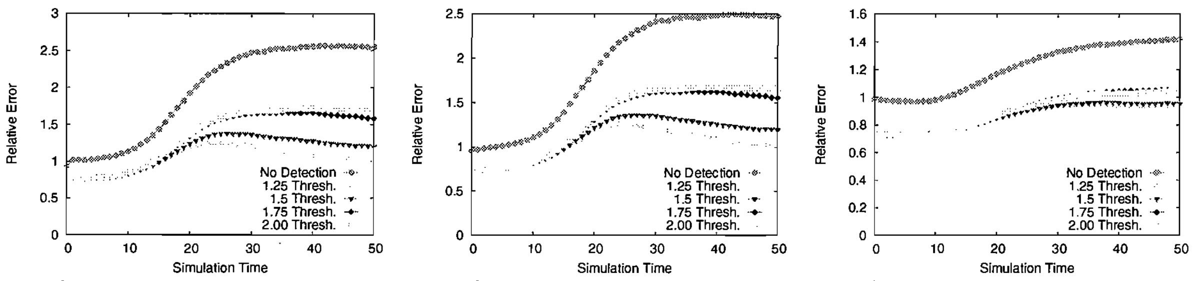

(g) 5th Pct. Relative Error, 30% Mal. (h) 50th Pct. Relative Error, 30% Mal. (i) 95th Pct. Relative Error, 30% Mal.

Figure 7: Relative error under different percentage of attackers using different spatial outlier thresholds with the King data set

Table 4: Number of Colluding Nodes Tolerated by Spatial Outlier Detection for Different Data Sets Using A Spatial Outlier Threshold of 1.5 (Reference set size is 64)

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

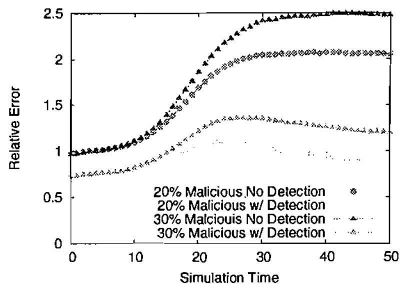
across the three data sets. The deflation attack is more successful for AMP since the RTTs are less variable and the virtual coordinate system creates one main cluster (Fig. 1(c)) that contains all of the nodes. This also explains why high percentages of malicious nodes ($\geq 30\%$) were able to overwhelm 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 can 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. This bound is in line with the requirements of other methods that tolerate malicious insiders.

System overhead. Our defense mechanisms do not introduce any extra link stress since they utilize information that is 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 bound by the number of nodes in the reference set which is constant. The computation of the temporal outliers is a constant time calculation performed for each of the nodes when deciding to update its coordinate. The calculation of the spatial correlation is also computed in constant time.

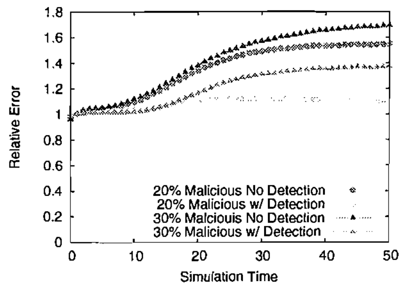
5. RELATED WORK

In this section, we review previous work in three areas:

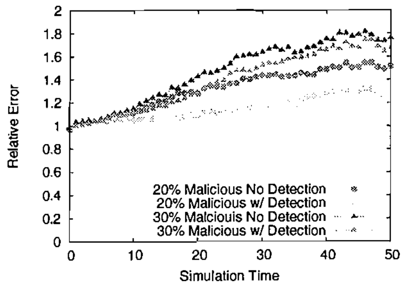
Attacks in virtual coordinate systems. One of the



(a) King



(b) Meridian



(c) AMP

Figure 8: Relative error under different percentage of attackers using a spatial outlier threshold of 1.5 with three real-life Internet latency data sets

few systems to consider actual malicious behavior is the PIC [12] virtual coordinate system which uses a security test based on the triangle inequality. Any node which violates the triangle inequality above some margin of error is ignored and designated as malicious. However, it has been shown in [23, 24, 25] that RTT measurements often violate this inequality and thus solutions based solely on such inequalities may degrade system performance when no attack is occurring. In [20, 21] the authors demonstrate the susceptibility of the Vivaldi to attacks. However, no solution is proposed.

Coordinate system error and landmark selection. An important area of research orthogonal to the security of the system is the minimization of error in the system. The accuracy of such systems is greatly effected by landmark placement for centralized schemes and neighbor selection in decentralized schemes. In [47], it is shown that a hierarchical approach can lead to better performance over non-hierarchical solutions. Work such as [25] and [48] demonstrate shortcomings of current systems and propose possible new metrics and measurements to more accurately embed the distance 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.

Use of spatial and temporal correlations. Recently the benefits of the Mahalanobis distance for statistical anomaly detection have been demonstrated in the context of network intrusion detection [40, 49]. In [49], the authors present a comparative study of detection schemes based on data mining techniques for network based intrusion detection. In [40] the authors discuss an unsupervised, payload-based network anomaly detector based on the Mahalanobis distance which was used to detect attacks like worms.

Spatial and temporal correlations were previously used in the context of network security. A notable work in this aspect is [44] where authors use temporal and spatial correlations to trace back attacks and detect attack scenarios, using a large amount of information from intrusion detection systems, firewalls, and different software logs. Unlike the approach in [44], which was more general, our work focuses on virtual coordinate systems.

Correlations have also been used in wireless networks for the detection of attacks [50, 51]. The work in [50] uses correlations between different features to identify attacks against wireless ad hoc routing protocols while the work in [51] shows how to augment sensor networks with spatio-temporal correlation to detect misinformation being injected into the sensor streams. In our work, the correlation is incorporated in-line with the coordinate computation and analysis is performed on real Internet data sets.

6. 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 even a small number of attackers can severely degrade coordinate accuracy due do the epidemic nature of the attacks. We proposed to use spatial-temporal correlation to perform outlier detection on metrics received from malicious nodes and eliminate them from the coordinate computation process. By using analytical and empirical methods we found that a spatial temporal of 1.5 and a temporal threshold of 4 produced a low system error and maintained an acceptable false positive rate. Finally, we examined the limitations of outlier detection when a significant percentage of nodes are malicious and found that the method starts degrading when more than 30% of the nodes in a reference set form a malicious coalition.

Future work includes analyzing the relation between reference set size and the system size and the effect of our mechanisms on upper level applications using virtual coordinate systems to estimate network measurements.

7. REFERENCES

[1] "Bittorrent." http://www.bittorrent.com/.

[2] Y. Kulbak and D. Bickson, "The eMule Protocol Specification," *eMule project, http://sourceforge. net.*

[3] "Skype." http://www.skype.com/.

[4] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast (keynote address)," in *SIGMETRICS '00*, 2000.

[5] X. Zhang, J. Liu, B. Li, and T. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," *Proc. IEEE Infocom*, 2005.

[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Tech. Rep. UCB/CSD-01-1141, UC Berkeley, Apr. 2001.

[7] T. Ng and H. Zhang, "A network positioning system for the internet," *Proc. USENIX Conference*, 2004.

[8] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location

information," 2003.

[9] L. Tang and M. Crovella, "Virtual landmarks for the internet," 2003.

[10] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *Proceedings of SIGCOMM '04*, 2004.

[11] E. Ng and H. Zhang, "Predicting internet network distance with coordiantes-based approaches," 2002.

[12] M. Costa, M. Castro, R. Rowstron, and P. Key, "PIC: practical Internet coordinates for distance estimation," *Proc. of the ICDCS '04*, 2004.

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

[14] H. Lim, J. Hou, and C. Choi, "Constructing internet coordinate system based on delay measurement," '03.

[15] L. wei Lehman and S. Lerman, "A decentralized network coordinate system for robust internet distance," in *ITNG '06*, 2006.

[16] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "Idmaps: A global internet host distance estimation service," 2000.

[17] M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris, "Lighthouses for scalable distributed location," 2003.

[18] L. wei Lehman and S. Lerman, "Pcoord: Network position estimation using peer-to-peer measurements," in *Proc. of IEEE NCA '04*, 2004.

[19] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth, "Bar fault tolerance for cooperative services," in *SOSP '05*, Dec. 2005.

[20] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous, "Real attacks on virtual networks: Vivaldi out of tune," in *Proc. of LSAD '06*, 2006.

[21] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous, "Virtual networks under attack: Disrupting internet coordinate systems," 2006.

[22] J. Ledlie, P. Pietzuch, M. Mitzenmacher, and M. Seltzer, "Wired geometric routing," in *Proc. of IPTPS '07*, 2007.

[23] J. Ledlie, P. Gardner, and M. Seltzer, "Network coordinates in the wild," in 4th *USENIX NSDI*, 2007.

[24] H. Zheng, E. Lua, M. Pias, and T. Griffin, "Internet routing policies and round-trip-times," *Proceedings of the Passive Active Measurement 2005.*

[25] E. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft, "On the accuracy of embeddings for internet coordinate systems," *Proc. of IMC*, 2005.

[26] "p2psim: A simulator for peer-to-peer protocols." http://pdos.csail.mit.edu/p2psim/.

[27] B. Wong, A. Slivkins, and E. Sirer, "Meridian: a lightweight network location service without virtual coordinates," *ACM SIGCOMM '05*, 2005.

[28] "Nlanr active measurement project." http://amp.nlanr.net/.

[29] Y. Shavitt and T. Tankel, "Big-bang simulation for embedding network distances in euclidean space," 2002.

[30] C. Lumezanu and N. Spring, "Playing Vivaldi in Hyperbolic Space," *Proc. of Internet Measurement Conference, Rio de Janeiro, Brazil, Oct*, 2006.

[31] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proc. of OSDI '02.*

[32] V. Barnett and T. Lewis, "Outliers in statistical data," 1978.

[33] R. Johnson and D. Wichern, *Applied multivariate statistical analysis.* Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1988.

[34] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. 13, no. 2, pp. 222–232, 1987.

[35] C. Sargor, "Statistical anomaly detection for link-state routing protocols," in *Proc. of ICNP '98*, 1998.

[36] Z. Ferdousi and A. Maeda, "Unsupervised outlier detection in time series data," in *Proc. of IEEE ICDEW '06*, 2006.

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

[38] E. M. Knorr and R. T. Ng, "Algorithms for mining distance-based outliers in large datasets," in *Proc. 24th Int. Conf. on VLDB '98.*

[39] D. Birant and A. Kut, "Spatio-temporal outlier detection in large databases," 2006.

[40] K. Wang and S. J. Stolfo, "Anomalous Payload-based Network Intrusion Detection," in *Proceedings of (RAID)*, 2004.

[41] C. Lu, D. Chen, and Y. Kou, "Multivariate spatial outlier detection," *Intl. Journal on Artificial Intelligence Tools, World Scientific*, vol. 13, no. 4, '04.

[42] A. Walters, D. Zage, and C. Nita-Rotaru, "Mitigating attacks against measurement-based adaptation mechanisms in unstructured multicast overlay networks," in *Proc. of ICNP '06*, November 2006.

[43] D. E. Knuth, *The Art of Computer Programming, 2nd Ed.* Boston, MA, USA: Addison-Wesley Longman Publ. Co., Inc., 1978.

[44] G. Jiang and G. Cybenko, "Temporal and spatial distributed event correlation for network security," in *American Control Conference*, 2004.

[45] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Intl. Journal of Robotics Research*, vol. 5.4, 1986.

[46] M. I. Ribeiro, "Gaussian probability density functions: Properties and error characterization," 2003.

[47] R. Zhang, C. Hu, X. Lin, and S. Fahmy, "A hierarchical approach to internet distance prediction," in *Proc. of IEEE ICDCS*, 2006.

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

[49] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proc. of 3rd SIAM Conf. on Data Mining*, 2003.

[50] Y. an Huang, W. Fan, W. Lee, and P. S. Yu, "Cross-feature analysis for detecting ad-hoc routing anomalies," in *ICDCS '03*, 2003.

[51] S. Tanachaiwiwat and A. Helmy, "Correlation analysis for alleviating effects of inserted data in wireless sensor networks," in *MobiQuitous*, 2005.