Securing Application-Level Topology Estimation Networks: Facing the Frog-Boiling Attack

Sheila Becker[1], Jeff Seibert[2], Cristina Nita-Rotaru[2], and Radu State[1]

[1] University of Luxembourg - SnT, L-1359 Luxembourg
{sheila.becker, radu.state}@uni.lu
[2] Purdue University, West Lafayette, IN 47906, USA
{jcseiber, crisn}@cs.purdue.edu

Abstract. Peer-to-peer real-time communication and media streaming applications optimize their performance by using application-level topology estimation services such as *virtual coordinate systems*. Virtual coordinate systems allow nodes in a peer-to-peer network to accurately predict latency between arbitrary nodes without the need of performing extensive measurements. However, systems that leverage virtual coordinates as supporting building blocks, are prone to attacks conducted by compromised nodes that aim at disrupting, eavesdropping, or mangling with the underlying communications.
Recent research proposed techniques to mitigate basic attacks (*inflation, deflation, oscillation*) considering a single attack strategy model where attackers perform only one type of attack. In this work we explore supervised machine learning techniques to mitigate more subtle yet highly effective attacks (*frog-boiling, network-partition*) that are able to bypass existing defenses. We evaluate our techniques on the Vivaldi system against a more complex attack strategy model, where attackers perform sequences of all known attacks against virtual coordinate systems, using both simulations and Internet deployments.

1 Introduction

Several recent peer-to-peer architectures optimize underlying communication flows by relying on additional topological information in order to meet the performance requirements of real-time communication and live media streaming applications. These architectures vary from distributed approaches, where peers can independently check the traffic specific network conditions and select the most appropriate candidate [34] to more centralized approaches, where an Internet Service Provider (ISP) is actively helping this process by means of an *oracle* service [5]. Specifically, an ISP can help avoid the overlay-underlay routing clash by ranking peers according to several metrics such that peer-to-peer traffic remains largely within the same Autonomous System (AS). The latter approach is being followed by the IETF, where the *Application-Layer Traffic Optimization (ALTO)* [30] working group has defined a framework for providing a service for efficiently selecting peers with the objective of improving the performance of peer-to-peer applications without disrupting ISPs.

One way of efficiently selecting peers is to leverage an application-level topology estimation service for defining virtual coordinates for use in the peer selection process

[30]. Virtual coordinates consist of mapping each host to a multidimensional metric space, such that the distance metric between coordinates can approximate network level measurements among the original hosts. This mapping is done iteratively, as each host probes one or several other hosts and individually adjusts its virtual coordinates. Typical network level metrics are bandwidth and round-trip time (RTT) and several coordinate systems have been introduced in the past. For an extensive overview on the existing approaches, the reader is referred to [16].

Systems that leverage virtual coordinates as supporting building blocks are prone to attacks conducted by compromised nodes that aim at disrupting, eavesdropping, or mangling with the underlying communications. These attacks aim at disrupting services relying on virtual coordinates and this is done by biasing the mapping process. The consequences of such attacks range from traffic eavesdropping, where attackers manipulate the virtual coordinates in order to force their location to be part of the communication path, to denial of service attacks, that lead to an unstable and inefficient overlay network. Specifically, identified attacks against virtual coordinate systems are: *inflation/deflation* - where the coordinate of a node is made to appear bigger/smaller and *oscillation* - where an attacker destabilizes the coordinate system. Previous research [32, 37] proposed techniques to mitigate these attacks considering a single attack strategy model in which attackers perform the same type of attack for the entire duration of the attack. Recent research [10, 11] identified new, more subtle and yet highly effective attacks called *frog-boiling* and *network-partition* that are able to bypass such defenses. During frog-boiling, attackers lie about their coordinates only by small amounts, but over time continuously move away from their correct positions. Network-partition is a variant of the frog-boiling attack where groups of attackers move their coordinates in opposite directions. No solutions to these attacks have been proposed to the best of our knowledge.

In this paper, we consider the detection of all existing attacks against decentralized virtual coordinate systems by leveraging supervised machine learning methods: decision trees and support vector machines. Our approach is able to detect and mitigate all known attacks used in both single attack strategies where individual attacks (frog-boiling, network-partition, oscillation, inflation and deflation) are launched by an attacker and more complex attack strategies, where successive attack phases are intermixed without assuming any fixed order in the attack sequence. Our contributions are as follows:

- We propose a practical method to counter the frog-boiling and network-partition attacks, or any complex attack strategy in which several individual attacks are launched by a powerful adversary. For example, the latter can combine several single attacks following a Markov chain model.
- We develop a feature set, based on a node's local information, for embedding it into a multidimensional manifold in order to reveal attacks. This process has resulted in seven feature variables that prove to be the most relevant for the prediction and classification task.
- We provide a quantitative analysis of supervised machine learning methods, *i.e.,* decision trees and support vector machines, for detecting all known attacks. We evaluate our techniques using the Vivaldi [15] virtual coordinate system through simulations using the King data set and real deployments on PlanetLab. Among the

two different machine learning techniques, decision trees and support vector machines, decision trees are able to mitigate all known attacks, outperforming support vector machines by achieving a much lower false positive rate. Our approach works both in a global manner, where all nodes actively exchange local information and a collective decision is taken, as well as in an individual manner, where each node locally decides whether an attack is occurring or not. The results for simulations using the King data set and for real deployments on PlanetLab both demonstrate good performance in terms of true positives ($\sim$ 95%) for identifying the different attacks.

The remainder of this paper is structured as follows. We overview virtual coordinate systems in Section 2. We describe existing known attacks and some limitations of existing protection mechanisms in Section 3. We describe our defense method in Section 4 and present experimental validation in Section 5. We discuss related work in Section 6. Finally, we conclude the paper in Section 7.

2 System Model

In this section, we give an overview of virtual coordinate systems and a representative decentralized system, Vivaldi, that we use in our simulations and experiments.

2.1 Virtual Coordinate Systems

Virtual Coordinate Systems (VCS) have been proposed as a way to accurately predict latency between arbitrary nodes without the need of performing extensive measurements. In a VCS, each node maintains a coordinate where the distance between two node's coordinates is the estimated round-trip time (RTT). The main service goals of virtual coordinate systems are the *accuracy* and *stability* of the resulting virtual coordinates. Accuracy captures how well the coordinates estimate actual RTTs between nodes. Stability captures the ability of the system to converge to the real coordinate values.

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 [17, 26, 27] or randomly selected [28, 35]. 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 [13], Vivaldi [15], and PCoord [23, 24].

In this paper, we focus on decentralized virtual coordinate systems as several such systems have become popular due to their low cost of deployment and increased scalability. In particular, we use Vivaldi [15] as a representative decentralized virtual coordinate system. We chose Vivaldi because it is a mature and widely-deployed system that has been shown to produce coordinates that result in low error estimations and is able to do so with reasonable performance and overhead.

2.2 Vivaldi Overview

Vivaldi is based on a spring-mass system where all nodes are connected via springs, where the current length of the spring is the estimated RTT and the actual RTT is considered to be the spring length when at rest. Thus as with real springs, if a spring is compressed it applies a force that pushes the nodes apart and if the spring is extended the spring pulls them together. Over time, the tension across all springs is minimized, and the position of each node produces the resulting coordinate.

Specifically, the Vivaldi protocol works as follows. Each node i is first assigned a coordinate x_i that is at the origin of the coordinate space and also finds several neighbors with which it exchanges updates. Every node i maintains a local error value e_i that is initialized to 1 and decreases as the RTT estimations improve. Node i will occasionally request an update from node j, which consists of node j's coordinate and local error. Node i also uses this opportunity to measure the RTT between itself and j. Once node i has this information it follows the update process as shown in Algorithm 1. An observation confidence w is calculated first (line 1) along with the error e_s in comparing the coordinates with the actual RTT (line 2). The local error value is then updated (line 4) by calculating an exponentially-weighted moving average with weight α and system parameter c_e (line 3). The movement dampening factor is then calculated with another system parameter c_c (line 5) and finally the coordinate is updated (line 6).

Algorithm 1: Vivaldi Coordinate Update

```
  Input: Remote node observation tuple ($\langle x_j, e_j, RTT_{ij} \rangle$)
  Result: Updated local node coordinate and error ($x_i$, $e_i$)
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 $x_i = x_i + \delta \times (RTT_{ij} - \|x_i - x_j\|) \times u(x_i - x_j)$
```

3 Attack Model and Strategies

While Vivaldi produces coordinates that can accurately predict RTTs, it is also vulnerable to insider attacks. An attacker can lie about its coordinate and local error, and can also increase the RTT by delaying probes sent to determine the RTT between itself and other nodes. As has been shown [6, 21], Vivaldi is vulnerable against such attacks that can lead to producing coordinates that have high error. Such attacks can be conducted by a set of attacker nodes, either individually or coordinating together. An attacker can mount an attack by using only one type of attack, or by mixing several attacks.

Below we first describe single attack scenarios where a malicious node applies the same attack for the entire duration of the experiment and all nodes apply the same attack. We then extend these scenarios to more complex ones, by assuming that not only one single attack is applied by all the malicious nodes, but sequences of different attacks can be launched.

3.1 Single Attack Strategies

Basic Attacks. Several basic attacks specific to coordinate systems have been identified. They are: inflation and deflation attacks that impact the accuracy of coordinate systems, and oscillation attacks [37] that impact both the accuracy and stability of coordinate systems. In an inflation attack, malicious nodes report a very large coordinate to pull nodes away from correct coordinates. In a deflation attack, to prevent benign nodes from updating and moving towards their correct coordinates, malicious nodes report coordinates near the origin. Finally, in an oscillation attack, malicious nodes report randomly chosen coordinates and increase the RTT by delaying probes for some randomly chosen amount of time. In each of these attacks, nodes report a small, but randomly chosen, local error value, signaling that they have high confidence in their coordinate position.

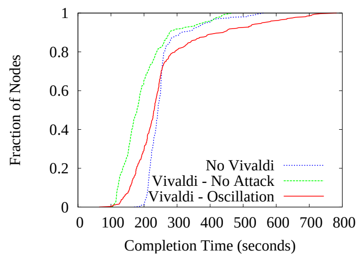
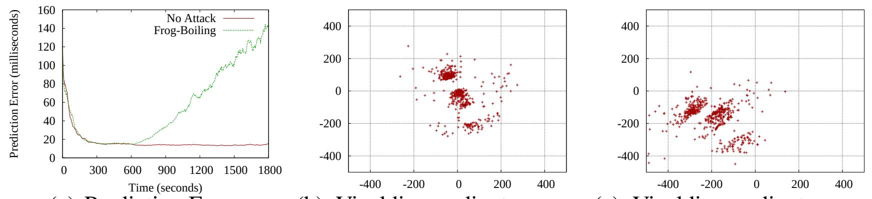


Fig. 1. Oscillation attacks against BitTorrent, 315 nodes (10% malicious) on PlanetLab.

To show how a small number of malicious nodes conducting oscillation attacks can affect application performance, we evaluated the file-sharing BitTorrent system [12] in a real-life PlanetLab [3] deployment of 315 nodes, out of which 10% act maliciously. We compare three scenarios in Fig. 1: **No Vivaldi**, the scenario where the BitTorrent tracker does not use virtual coordinates, but simply chooses nodes at random; **Vivaldi - No Attack**, the scenario where the tracker is coordinate-aware, i.e. when a client requests other peers to download from, the tracker will respond with a selection of nodes that are near the coordinate of the requesting node; **Vivaldi - Oscillation**, the scenario where the coordinates used by BiTorrent are impacted by an oscillation attack against Vivaldi. In our implementation, malicious nodes report randomly chosen coordinates and increase the RTT by delaying probes for up to 1 second. As can be seen in Fig. 1, when the tracker is aware of coordinates, the download times decreases by 50% for some nodes. However, when under attack, much of the gains brought on by the coordinates are lost, and for over 25% of nodes, the download times actually increase over the scenario when no virtual coordinates are used to optimize peer selection.

Advanced Attacks. Several proposals have been made to secure virtual coordinate systems against the above described basic attacks [20, 32, 37] and have been shown to effectively mitigate them. However, recent research [10, 11] has identified two more subtle and yet highly effective attacks that are able to bypass existing defenses. They are the frog-boiling and network-partition attacks. In a frog-boiling attack malicious nodes lie about their coordinates or latency by a very small amount to remain undetected by defense mechanisms. The key of the attack is that the malicious nodes gradually increase the amount they are lying about and continue to move further away from their correct coordinates, successfully manipulating benign node's coordinates and thus producing inaccurate RTT estimations. In a network-partition attack two or more groups of malicious nodes conduct a frog-boiling attack, but move their coordinates in opposite directions, effectively splitting the nodes into two or more groups.



(a) Prediction Error (b) Vivaldi coordinates mapping before attack (c) Vivaldi coordinates mapping after attack

Fig. 2. Frog-boiling attack against Vivaldi, 500 nodes (10% malicious) on PlanetLab.

We illustrate the effects of a frog-boiling attack conducted by a small group of attackers on the accuracy of Vivaldi on a real-life PlanetLab deployment of 500 nodes, out of which 10% act maliciously. We measure accuracy by evaluating the prediction error defined as:

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

where RTT_{Act} is the measured RTT and RTT_{Est} is the estimated RTT. Fig. 2(a) displays the median prediction error between all pairs of nodes. In this experiment, malicious nodes start the attack after 600 seconds, moving their coordinates only 250 microseconds every time they report their coordinate, and thus gradually increasing the prediction error over time. We also plot the coordinates of nodes before and after the attack has an effect, in Fig. 2(b) and Fig. 2(c) respectively. The attack thus results in nodes moving away from their correct coordinates and also away from the origin.

3.2 Complex Attack Strategies

Prior work has considered only single attack strategies, where a malicious node applies the same attack (inflation, deflation, oscillation, frog-boiling, network-partition) for the entire duration of the attack and all nodes apply the same attack. However, single attack strategies can be easily detected using techniques that leverage change-point detection methods. We extend these scenarios to more complex ones, by assuming that not only one single attack is applied by all the malicious nodes, but sequences of different attacks can be constructed. Sequences of different attacks do raise the stakes significantly, since the observed patterns are less easy to detect. We also consider cases when attackers do not all apply the same attack.

Single random attack scenario. One way to extend in a straightforward way the single attack strategy is to consider the case where nodes do not perform all the same attack. In this case, each node randomly selects one of the five single attacks, and applies no attack for some time, then switches to the randomly selected attack. This designates that one malicious node may conduct the inflation attack, while another malicious node conducts the frog-boiling attack. We refer to this attack strategy as Single-Random.

Two attack scenario. Another extension of the single attack strategy is a scenario where an attacker alternates between any of the five single attacks, interleaving them with a period of no attack. Specifically, such a strategy is composed of four equal time slots, the first time slot is a non attacking slot, the second consists of one of the five single attacks, followed by another non attacking slot, and finally the fourth time slot is a second single attack. The idea behind this model is to see how the existing detection methods, as well as the methods we propose in this paper, perform in comparison to single attack scenarios. We experiment with several of such scenarios and select the following as representative: Deflation - Frog-Boiling, Oscillation - Inflation, Network-Partition - Oscillation.

Sequence attack scenario. We model more complex attack scenarios, where the attacker applies different sequences of attacks, by using a Markov chain model. The states of such a chain represent all the different single attacks including the *No Attack* state in which an attacker does not apply an attack. The Markov chain is presented in Figure 3. This Markov chain is irreducible, as the state space is one single communicating class, meaning that every state is accessible from every state. We consider an irreducible chain, as we assume that the attacker can change the current attack strategy to any other attack, and even stop attacking for a while. Therefore, an attacker can execute every attack at any time, independently of what he has executed previously. Furthermore, the chain is aperiodic, as a return to a specific state can happen at irregular times. An attack that was already executed previously might be utilized again from time to time. Summarizing, we can say that the Markov chain is ergodic, as it is aperiodic, irreducible and positive recurrent. Such an ergodic chain allows to visit individual states indefinitely often and thus leads to more complex scenarios.

The transition probabilities presented in Figure 3 reflect several design goals for generating sequences of attacks. From the *No Attack* state, each attack is equally probable, except the probability that no transition (and therefore no attack) is only 10%, therefore the transition to any attack state has the probability 18%. We chose these transition probabilities to avoid the risk of the Markov chain remaining in the *No Attack* state. From an attack state the transitions to every other attack state are equally probable with 15%. This results in the transition probability to the *No Attack* state to always be 25% such that we ensure that there are some no attack intervals and that an attacker does not remain in an attacking state.

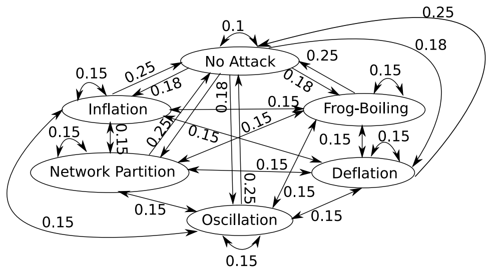


Fig. 3. Markov Chain with the different attacks and the transition probabilities

Based on the Markov chain presented in Figure 3 we created and assessed twenty different sequence-scenarios. All sequences start in the *No Attack* state. Below we describe the most relevant scenarios in terms of representing the different groups of sequences, one group that has a very small amount of non-attacking intervals, another group with intermediate values of non-attacking intervals, and the last group that has the highest amount of non-attacking intervals. We base our selection on the amount of non-attacking intervals as characteristic due to the importance of these intervals for the detection method leveraged in this work. Below we utilize the term iteration, an iteration is equivalent to 0.5% of the duration of an experiment. We focus on the following scenarios:

– *Sequence A*: No attack 15 iterations; inflation 15 iterations; network-partition 55 iterations; deflation 35 iterations; inflation 45 iterations; inflation 35 iterations. *Total amount of non-attacking intervals: 15*
– *Sequence B*: No attack 10 iterations; inflation 55 iterations; oscillation 50 iterations; frog-boiling 55 iterations; network-partition 30 iterations. *Total amount of non-attacking intervals: 10*
– *Sequence C*: No attack 30 iterations; network-partition 35 iterations; frog-boiling 35 iterations; No attack 15 iterations; frog-boiling 40 iterations; inflation 45 iterations. *Total amount of non-attacking intervals: 45*
– *Sequence D*: No attack 40 iterations; inflation 30 iterations; oscillation 40 iterations; network-partition 40 iterations; frog-boiling 35 iterations; No attack 15 iterations. *Total amount of non-attacking intervals: 55*
– *Sequence E*: No attack 50 iterations; inflation 10 iterations; No attack 50 iterations; oscillation 55 iterations; oscillation 10 iterations; inflation 25 iterations. *Total amount of non-attacking intervals: 100*
– *Sequence F*: No attack 55 iterations; network-partition 40 iterations; No attack 15 iterations; frog-boiling 45 iterations; inflation 15 iterations; No attack 25 iterations; No attack 5 iterations. *Total amount of non-attacking intervals: 100*

We note that in these sequences of attacks, we still consider malicious nodes that work together by applying the same attacks in the same time interval.

4 Mitigation Framework

This section describes our new mitigation framework based on machine learning techniques, and presents the feature set that we leveraged for use by the machine learning technique.

4.1 Background

Machine learning techniques, such as classification, have the aim to separate a given data set into different classes. In our case, the classes that exist are *normal* and *attack*, meaning that we have two different types of data in our data set. On one side, we have data that represents normal updates of the nodes, and on the other side, we have data that represents malicious update requests.

We choose to apply supervised classification methods as we know how the system works under normal circumstances and also how the attacks degrade performance when they are taking place. These classification methods are fed with training data to learn the difference between normal and malicious data. Supervised classification methods can operate directly in the feature space/predictor variables and identify separable regions that can be associated to a given class/dependent categorical variable. Such methods are implemented by decision trees that come in several variants. Simple versions such as Classification and Regression (Cart) [8] can predict both categorical and numerical outcomes, while other schemes (C4.5 for instance) relying on information theory [29] are uniquely adapted to categorical outputs. Another type of classification method, support vector machines, map the input space into another dimensional space and then rely on kernel functions for performing classification in the target space [9].

4.2 Feature Set

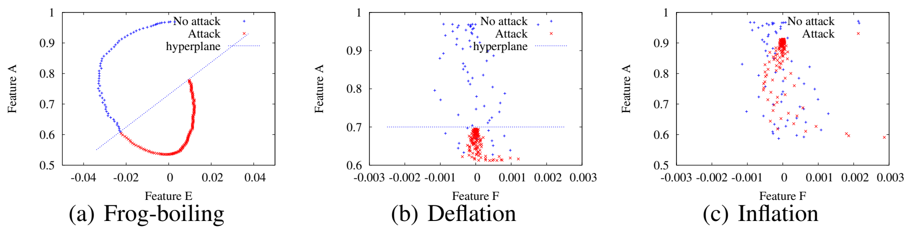
We have evaluated three different methods (SimpleCart, C4.5, and support vector machines) for their efficiency in protecting virtual coordinate systems. We did this for several reasons: first, we wanted to compare the individual approaches and identify the best one. Second, we considered that providing these results allows a more comprehensive analysis of the detection process, as well as to highlight some of the peculiar properties related to the different methods.

We have identified seven feature variables to be used in the prediction task. This process was challenging since several approaches that worked directly on the raw data were not successful. The raw data consisted, in our case, of statistical properties of the underlying local error values. We have analyzed the time series values of both the median and the average local error, but a straightforward analysis of simple time series values did not perform well. This was due to a four lag autocorrelation in the observed time series. In order to decorrelate the time series values, we applied an embedding of the observed one dimensional data into a seven dimensional manifold. Values in the original time series are given by the median local error described in Section 2.2. The embedding into a multidimensional manifold aims at revealing subspaces that can be associated to attack states and respectively non-attack ones. Thus, at each sample moment in time, we need to analyze a seven dimensional random vector.

1. *Feature A* is the median local error of the nodes e_{median}. This feature represents the global evolution of the local error. Intuitively, a low median local error means that most of the nodes have converged to their coordinates.
2. *Feature B* represents the difference of the median local error at one lag $\delta_1 = e_{median_t}$-$e_{median_{t-1}}$. This feature captures the sense of the variation in the local error. Positive

values indicate an increase in the error, while negative values show continuous decrease in the error. This feature can be seen as a discretized first derivate of the observed process. Although, discrete time events are used to index the time series, by analogy to the continuous case, we assume that this discretized first derivate captures the sense (increasing/decreasing) of the underlying time series.

3. *Feature C* is $\delta_2 = e_{median_t}$- $e_{median_{t-2}}$. This feature relates current values to previous values at a two lag distance.
4. *Feature D* is $\delta_3 = e_{median_t}$- $e_{median_{t-3}}$, is similar to *feature C*, but works at a three lag distance.
5. *Feature E* is $\delta_4 = e_{median_t}$- $e_{median_{t-4}}$. It captures longer dependence (lag four).
6. *Feature F* captures the discretized form of the second order derivate δ_{1_t}- $\delta_{1_{t-1}}$. Basically, this feature can indicate the shape (concave/convex) of the initial time series. We assume a discretized equivalent of the continuous definition.
7. *Feature G* is the absolute value of the discretized form of the second order derivate $|\ \delta_{1_t}$- $\delta_{1_{t-1}}\ |$. This absolute value can provide insights in inflection points (i.e., points, where a switch from convex to concave, or concave to convex is happening).

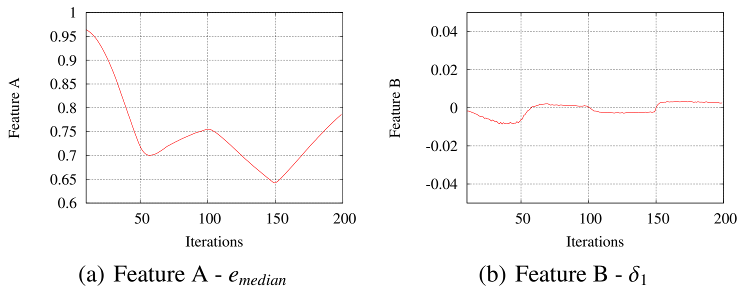


(a) Frog-boiling (b) Deflation (c) Inflation

Fig. 4. Bi-dimensional and pairwise feature representation

We can not visualize a seven dimensional manifold, but bi-dimensional pairwise scatter plots can illustrate the rationale for our approach. Figure 4(a) shows the two dimensional scatter plot for a frog-boiling attack. Feature A is used for the x-axis and feature E for y-axis. The two classes (attack and non attack) can be linearly separated in this two dimensional subspace. Figure 4(b) shows another 2 dimensional scatter plot, where feature A and feature F are used. This scenario corresponds to a deflation attack. In this scenario, the classes can be also linearly separated, and thus we argue that these features are appropriate for defending against a deflation attack. However, in Figure 4(c), the same set of features used during an inflation attack shows very limited detection potential. However, the global set of all seven features can be leveraged to detect the different (frog-boiling, deflation, inflation, oscillation and network-partition) attacks.

The attack detection problem is stated thus as deciding whether a seven dimensional tuple is representing an attack or not. From a mitigation point of view, once an attack is identified several measures can be taken. In a first phase, updating the virtual coordinates can be resumed after the attack stops, or limited to updates received from known and trusted nodes. The latter assumes an underlying reputation or trust model. In a second phase, the attacking hosts should be identified and contained.



(a) Feature A - e_{median} (b) Feature B - δ_1

Fig. 5. Classification features

To provide some intuition behind our methodology we present in Figure 5 the evolution of two features for a dataset that contains a two attack strategy, Inflation - Oscillation. This attack scenario consists of four time slots, so the first is a non-attacking slot. The second is in this case an inflation attack. The third time slot is again non attacking, and the fourth and last time slot is the oscillation attack. The objective of classification is, as already mentioned, to separate the different classes of the data set. Two classes exist, the non-attacking and attack class. Within Figure 5, we want to illustrate how the classifier can identify the different classes. Figure 5(a) shows how feature A, the median of the error, evolves. In this simple case, the increasing or decreasing trends are easy to identify and one can define when the attacking time slots take place. Feature A decreases in a non-attacking time slot, and increases during an attack. However, feature B captures a smoothed version of the overall evolution. In these plots, we can identify intervals that correspond to positive y-values for feature B. These positive values belong to attacking time slots.

5 Experimental Results

In this section, we evaluate the single and complex attack strategies described in Section 3 using Vivaldi within three different environments. First, we evaluate the effectiveness of the machine learning techniques on the dataset resulting from simulation using the p2psim simulator [2] and the King data set topology [18]. Second, we compare our machine learning methods to a previously proposed solution using outlier detection [37] that can defend against inflation, deflation, and oscillation. Third, we evaluate our machine learning techniques on the data set resulting from deploying Vivaldi on 500 nodes on the Internet PlanetLab testbed [3]. We evaluate our detection method in two setups: global and local. In the global case, every node's information is centrally collected and analyzed together, while in the local case each individual node decides if an attack is taking place or not based only on its own information.

We use the classification model as described in Section 4. To calculate the feature set for the global case we take the median local error of each node in the system, i.e. for 1740 nodes in the simulation and for 500 nodes in the Internet PlanetLab testbed. The acquired model is applied to three different classifiers, namely the two decision trees, SimpleCart [8] and C4.5 [29], and the support vector machines, LibSVM [1]. All experiments for the simulator as well as for PlanetLab are evaluated using the Java source code of weka [4]. We have tried all different kernel functions and their corresponding parameters for the LibSVM and because no significant differences were relevant, we decided to use the default values that come with this weka composant: C-SVC for the kernel type, radial-basis kernel function, with the default values (degree in kernel function was set to 3, gamma parameter to 0.5 and nu to 0.5).

To evaluate the results, we calculate the percentage of attack events that the classifier correctly classifies, which we refer to as the true positive rate (TPR). We also calculate the percentage of non-attack events that the classifier incorrectly classifies as attack events, which we refer to as the false positive rate (FPR). We computed the TPR and FPR using the well established 10-fold cross-validation scheme, where the system is trained with randomly extracted $\frac{9}{10}$ of the data, and tested with $\frac{1}{10}$ of the data. This process is repeated 10 times for each classification.

5.1 Simulation Results

We conduct simulations using the King data set topology [18], as it is representative of an Internet-wide deployment of a peer-to-peer system and has been used previously to validate several other VCSes. The King data set consists of RTT measurements between 1740 nodes, of which the average RTT is 180ms. For each simulation, all nodes join in a flash-crowd sequence at the beginning of the simulation. The simulations last for 200 time units, where each time unit is 500 seconds. Each node independently chooses a neighbor set of 64 nodes from which it receives coordinate updates.

Single Attack Strategies. We start by analyzing single attack scenarios, as defined in Section 3, where the following single attacks are classified: inflation, deflation, oscillation, frog-boiling, network-partition, and single-random. Table 1 shows the classification results. The data set consists of the first 30% of the time where no attack occurs, and the remaining 70% the attack does take place. This distribution of time intervals was chosen because some amount of samples of normal data, without attacks, is needed for training.

We note that for the decision trees, SimpleCart and C4.5, the TPR is, for all the different attacks, around 99%, and the FPR for the two classifiers is around 3%. This means that these decision trees can classify correctly almost all entries. Furthermore, while the number of attackers applying the given attack is increasing, the TPR remains more or less the same, whereas the FPR increases most of the time. Out of this we see that even though most attacks are still correctly classified, normal updates are classified incorrectly more often. We also observe that in these cases support vector machines perform badly, especially with regard to the FPR. In order to see to what degree decision trees can detect a frog-boiling attack, we applied a ten-times slower frog-boiling attack

Table 1. p2psim - Single Attack Strategies - Classification Results

Attack Strategy		SimpleCart		C4.5		LibSVM	
		TPR	FPR	TPR	FPR	TPR	FPR
Inflation	10% attackers	0.99	0.01	0.99	0.02	0.67	0.67
	20% attackers	0.99	0.01	0.99	0.02	0.67	0.67
	30% attackers	0.97	0.05	0.99	0.02	0.67	0.67
Deflation	10% attackers	0.99	0.013	0.99	0.02	0.67	0.66
	20% attackers	0.98	0.021	0.98	0.02	0.67	0.67
	30% attackers	0.98	0.016	0.97	0.03	0.67	0.67
Oscillation	10% attackers	0.099	0.008	0.99	0.01	0.67	0.66
	20% attackers	0.98	0.02	0.99	0.03	0.67	0.67
	30% attackers	0.98	0.020	0.98	0.030	0.67	0.67
Frog-Boiling	10% attackers	0.99	0.011	0.99	0.013	0.68	0.64
	20% attackers	0.99	0.016	0.98	0.03	0.67	0.67
	30% attackers	0.98	0.025	0.98	0.03	0.67	0.67
Network-Partition	10% attackers	0.99	0.01	0.98	0.014	0.79	0.44
	20% attackers	0.99	0.01	0.99	0.01	0.67	0.67
	30% attackers	0.99	0.006	0.98	0.03	0.67	0.67
Single-Random	10% attackers	0.99	0.02	0.98	0.03	0.67	0.67
	20% attackers	0.99	0.003	0.98	0.02	0.67	0.67
	30% attackers	0.99	0.002	0.99	0.02	0.67	0.67

Table 2. p2psim - Complex Scenarios - Classification Results

(a) Two Attack Scenario

Attack Strategy		SimpleCart		C4.5		LibSVM	
		TPR	FPR	TPR	FPR	TPR	FPR
Deflation - Boiling	10% attackers	0.95	0.05	0.94	0.05	0.58	0.41
	20% attackers	0.96	0.05	0.95	0.05	0.51	0.47
	30% attackers	0.98	0.02	0.97	0.03	0.52	0.49
Oscillation - Inflation	10% attackers	0.97	0.04	0.97	0.04	0.51	0.48
	20% attackers	0.97	0.03	0.96	0.04	0.50	0.49
	30% attackers	0.96	0.04	0.97	0.03	0.51	0.49
Network-Partition - Oscillation	10% attackers	0.95	0.05	0.97	0.03	0.67	0.34
	20% attackers	0.91	0.09	0.93	0.07	0.54	0.46
	30% attackers	0.90	0.10	0.92	0.08	0.55	0.45

(b) Sequence Attack Scenario

	SimpleCart		C4.5		LibSVM	
	TPR	FPR	TPR	FPR	TPR	FPR
A	0.93	0.43	0.94	0.42	0.93	0.86
B	0.96	0.48	0.97	0.33	0.95	0.76
C	0.97	0.08	0.97	0.05	0.79	0.72
D	0.97	0.05	0.98	0.02	0.73	0.73
E	0.98	0.02	0.99	0.02	0.53	0.46
F	0.97	0.04	0.98	0.02	0.7	0.3

as well as hundred-times and thousand times slower and evaluated. We obtained also for this case a very good performance as result, for 10%, 20%, and 30% of malicious peers we achieve always a true positive rate around 98% and a false positive rate around 2%.

Complex Attack Scenarios. We now investigate more complex sequences of attacks, specifically the two attack and sequence attack scenarios as defined in Section 3.2. Table 2(a) describes the classification results regarding the two attack scenario. It can be seen that the TPR for both decision trees (i.e., SimpleCart and C4.5) is less than for the single attack scenarios and the FPR is in comparison a bit higher. Overall, the decision trees perform well, although the results are not as good as the single attack scenario. In comparison, the support vector machine library seems to ameliorate, especially in the context of the FPR for the "Network-Partition - Oscillation" attack sequence.

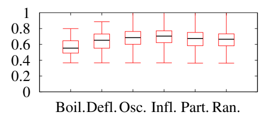
Furthermore, we produced different sequence-examples with the assessed Markov chain. Table 2(b) illustrates that all techniques have a very good TPR, whereas the FPRs differ significantly. We find that the difference lies in the amount of non-attacking intervals that each sequence has. Sequences A and B are in the group with only a small amount of non-attacking intervals - 10 and 15 intervals. The high FPR thus results due to the classifier not having enough training data for learning normal behavior. The two other groups show better results, for example, as sequences C and D have 45 and 55 normal intervals, respectively. Sequences E and F have in this case a quite high value of non-attacking intervals, both have 100 of them, so exactly half of the data set is non-attacking. We can deduce then that having only 5% non-attacking training data is definitely not enough, whereas 25% already shows good results. This outcome can be explained by the need for an heterogeneous training set for the decision trees; thus if we have less “No attack” time intervals, it is difficult for the classifier to learn what normal behavior is.

Comparison with Outlier Detection. In previous sections we showed that our classification techniques work well when applied globally. Nevertheless, previous works proposed mitigation techniques with respect to single nodes, even if only effective for inflation, deflation, and oscillation attacks. In particular, in the work from [37], each node independently decides if an update should be considered malicious or not by using spatial-temporal outlier detection. We compare our method, applied in a local manner where each node will classify attacks based only on its local information, with the work from [37], referred to as Outlier Detection in the remainder of the section. As this evaluation depends on the amount of updates those individual nodes receive, we observed some variety in the classification results. We illustrate the local classification results when there are 10% malicious nodes and for fifty randomly chosen benign nodes since this allows us to have a statistical overview over the whole data set. Based on these fifty nodes we create box-and-whisker diagrams, as those show the median values, the 25*th* and 75*th* percentiles, and the minimal and maximal value of each data set. These diagrams are shown in Figure 6 and in Figure 7. We show results only for the C4.5 technique as it has a similar performance with SimpleCart, while being more relevant in recent research, and it outperforms LibSVM.

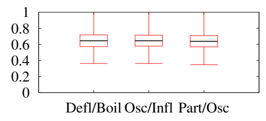
With respect to Figure 6 we note that for all the different cases of attack strategies considered, the classification technique performs better than Outlier Detection. In Figure 6(a), we see that Outlier Detection performs best for the inflation attack, and we see that frog-boiling has worse results. This is due to the fact that Outlier Detection can not handle frog-boiling as explained and shown in [10, 11]. Regarding Figure 7 we note that for all the different attack strategies, our classification technique has much better median FPRs than the Outlier Detection.

5.2 PlanetLab Results

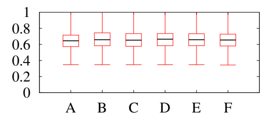
To validate our findings over the real Internet, we implemented Vivaldi and deployed it on PlanetLab. For our experiments we used 500 nodes, chosen from all over the world, from which the average RTT is 164ms. Each experiment was run for 30 minutes, while all other settings were the same as in the simulations. To find the effectiveness of our techniques, we apply in our PlanetLab experiments the same attacks and sequences as in the simulations on p2psim.



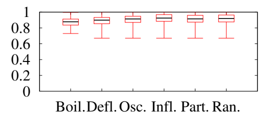
(a) OD - Single attacks



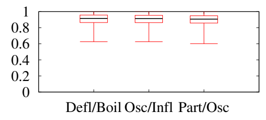
(b) OD - Two attacks



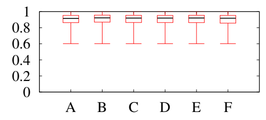
(c) OD - Sequences



(d) C4.5 - Single-attacks

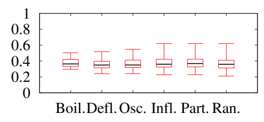


(e) C4.5 - Two Attacks

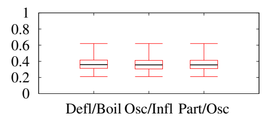


(f) C4.5 - Sequences

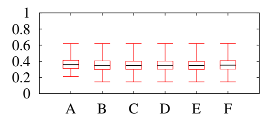
Fig. 6. p2psim - Outlier Detection Comparison (OD) -TPR



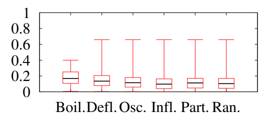
(a) OD - Single attacks



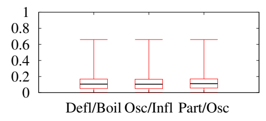
(b) OD - Two attacks



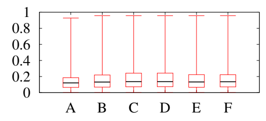
(c) OD - Sequences



(d) C4.5 - Single-attacks



(e) C4.5 - Two Attacks



(f) C4.5 - Sequences

Fig. 7. p2psim - Outlier Detection Comparison (OD) -FPR

Single Attack Strategies. In Table 3 the results for the single attack scenarios are illustrated, from which one can observe that for both decision trees the TPR and FPR are very good, which is similar to the simulation results. However, in the PlanetLab testbed we obtain much better results when applying the support vector machines.

Table 3. PlanetLab - Single Attack Strategies - Classification Results

Attack Strategy		SimpleCart		C4.5		LibSVM	
		TPR	FPR	TPR	FPR	TPR	FPR
Inflation	10% attackers	0.97	0.04	0.97	0.03	0.90	0.20
	20% attackers	0.95	0.08	0.95	0.08	0.91	0.17
	30% attackers	0.97	0.05	0.99	0.01	0.93	0.14
Deflation	10% attackers	0.99	0.02	0.98	0.2	0.90	0.21
	20% attackers	0.96	0.05	0.95	0.07	0.92	0.16
	30% attackers	0.97	0.05	0.98	0.03	0.93	0.13
Oscillation	10% attackers	0.99	0.02	0.99	0.01	0.95	0.10
	20% attackers	0.99	0.02	0.99	0.02	0.95	0.11
	30% attackers	0.99	0.02	0.99	0.02	0.95	0.09
Frog-Boiling	10% attackers	0.96	0.05	0.97	0.04	0.80	0.21
	20% attackers	0.97	0.04	0.98	0.03	0.85	0.15
	30% attackers	0.97	0.05	0.98	0.04	0.86	0.15
Network-Partition	10% attackers	0.93	0.10	0.93	0.07	0.83	0.17
	20% attackers	0.96	0.04	0.97	0.03	0.79	0.21
	30% attackers	0.96	0.05	0.94	0.08	0.85	0.14
Single-Random	10% attackers	0.99	0.03	0.98	0.03	0.92	0.16
	20% attackers	0.99	0.01	0.99	0.01	0.96	0.07
	30% attackers	0.99	0.014	0.99	0.013	0.94	0.11

Table 4. PlanetLab - Complex Scenarios - Classification Results

(a) Two Attack Scenarios

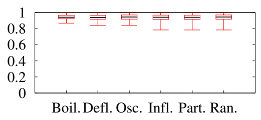
Attack Strategy		SimpleCart		C4.5		LibSVM	
		TPR	FPR	TPR	FPR	TPR	FPR
Deflation - Boiling	10% attackers	0.93	0.07	0.94	0.06	0.83	0.16
	20% attackers	0.88	0.11	0.89	0.10	0.86	0.13
	30% attackers	0.93	0.07	0.91	0.08	0.87	0.13
Oscillation - Inflation	10% attackers	0.95	0.05	0.95	0.05	0.92	0.085
	20% attackers	0.97	0.03	0.96	0.04	0.90	0.095
	30% attackers	0.97	0.03	0.97	0.03	0.89	0.11
Network-Partition - Oscillation	10% attackers	0.92	0.078	0.915	0.085	0.80	0.20
	20% attackers	0.89	0.11	0.90	0.09	0.84	0.16
	30% attackers	0.91	0.09	0.93	0.07	0.85	0.15

(b) Sequence Attack Scenarios

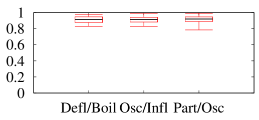
	SimpleCart		C4.5		LibSVM	
	TPR	FPR	TPR	FPR	TPR	FPR
A	0.93	0.83	0.93	0.65	0.93	0.93
B	0.95	0.95	0.94	0.52	0.95	0.95
C	0.86	0.26	0.84	0.31	0.78	0.78
D	0.87	0.21	0.89	0.22	0.73	0.71
E	0.95	0.06	0.96	0.05	0.95	0.06
F	0.87	0.14	0.88	0.12	0.80	0.19

Complex Attack Strategies. In addition, we also evaluate the more complex sequences as shown in Table 4. Table 4(a) provides results for the two-attack sequences. We note that, similar to the single attacks, the results for support vector machines are much improved for PlanetLab over the simulator. However, the opposite is true for the two decision trees, which did not perform as well on PlanetLab as they did for the simulations, especially for the 20% and 30% of malicious nodes. Overall, the results are still satisfying though, as the TPR is around 90% and the FPR does not exceed 11%. Table 4(b) illustrates the classification results for the sequence attack strategies.

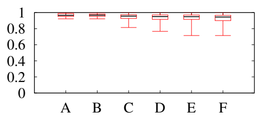
Local Classification. Furthermore, we also analyze PlanetLab results when each individual node decides locally if an attack is taking place or not based only on its individual information. We show the results in Figure 8. We illustrate the C4.5 classification technique, as it outperforms LibSVM, has similar performance to SimpleCart, and has been widely adopted. Similar to the simulations, we evaluate the results when there are 10% malicious nodes and for a set of fifty randomly chosen nodes to have again a statistical overview of the data. To illustrate the evaluation we again use box-and-whisker-diagrams.



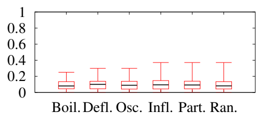
(a) Single attacks -TPR



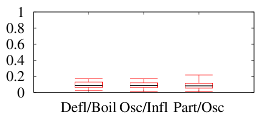
(b) Two attacks -TPR



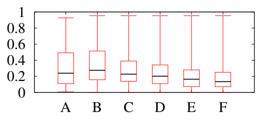
(c) Sequences - TPR



(d) Single attacks - FPR



(e) Two attacks - FPR



(f) Sequences - FPR

Fig. 8. PlanetLab Local Results

Figure 8 illustrates that C4.5 has a very high TPR in all the different attack strategies, which mirrors the results for the global classification. We also see that sequences A and B have high FPRs, which is similar to the global classification. Overall, except for sequences A and B, the results have good FPRs. This shows that the defined classification technique also work on a local basis when applied on a real Internet testbed.

6 Related Work

Anomaly detection has been extensively leveraged in developing intrusion detection systems [7, 19, 22], where for instance Bolzoni et al. [7] showed how to automatically and systematically classify detected attacks. The main idea was to compute similarities of the payloads of attack data, and later classify it automatically, semi-automatically, and even manually. One proposed method used support vector machines [36] and a rule learner algorithm for classification. While support vector machines have proved to be efficient (when tuned properly), in our case, we were surprised to discover that their potential usage was quite limited, despite extensive tuning with the most common kernel functions parameter calibration. This anecdotally confirms Sommer et al.'s [33] findings that Machine Learning techniques have often not been successful in real-world IDS applications due to that a detected anomaly does not immediately imply an attack. One major problem with any detection framework is given by the small drifts that might slowly bias the detection process. Repetitive training [14, 25] might be a general solution for decreasing the ratio of false positives, but in our work we show that such a process is not necessary for securing virtual coordinates systems.

Virtual coordinate systems have been protected against attacks in the past in several different ways. Kaafar et al. [20] use a trusted node set and anomaly detection using a Kalman filter to detect and discard malicious updates. Zage et al. [37] also use anomaly detection, but focus on a decentralized VCS without any trusted components. Outlier detection is performed by setting two different thresholds, a spatial and respectively a temporal one. Furthermore, Veracity [32] is a decentralized VCS that introduces the notion of a verification set. Each node maintains a verification set where several other nodes attest to whether a particular update increases their estimation error above a certain threshold, and if so, ignores it. We note, as described in Section 2, that all of these proposed systems have been shown to be insecure against the frog-boiling attack [10, 11]. Frog-boiling attacks have been mitigated before in a different context by ANTIDOTE [31]. ANTIDOTE is a principal component analysis-based poisoning attack detector that constructs a low dimensional subspace which reflects most of the dispersion in the data. The required computations are relatively expensive and assumes an existing multidimensional input space. Such assumptions do not hold in our case, where we had first to map the one dimensional data to a higher dimensional space (which is the opposite of ANTIDOTE's subspace construction) and then rely on an efficient and online decision mechanism.

7 Conclusion

In this paper, we have addressed the detection of different types of attacks against virtual coordinate systems. A detection method is presented for the known attacks, such as inflation, deflation and oscillation, as well as the recently identified frog-boiling and network-partition attacks. Besides these existing attacks, we have elaborated more complex attack strategies, the single-random attack scenario, two attack scenario, and sequence attack scenario. We have proposed, as a detection method, to apply supervised machine learning techniques that leverage decision trees, namely SimpleCart and C4.5, and support vector machines to detect all different attack strategies. For this reason a feature set is proposed and while representing this set in a multidimensional manifold, attacks can be revealed as these feature variables are used for the prediction and decision task.

We have validated our detection method through simulation using the King data set for the p2psim simulator as well as through real deployment on the PlanetLab testbed. The detection method is evaluated in a global manner, where the local information of all nodes are together analyzed, as well as in a local manner, where each node has only the local information to analyze and evaluate if an attack is happening or not. We have shown that in our setting, decision trees outperform support vector machines by achieving a much lower false positive rate. Regarding the two different types of decision trees, the results are similar, thus there is no clear better choice. The outcome for the sequence attack scenarios illustrates that a minimal set of normal data is needed for correctly classifying normal behavior, pointing to at most 25% of the data is needed to do so. Furthermore, we compared the proposed detection technique, the decision tree, to existing detection and mitigation techniques, outlier detection which is based on a threshold. This comparison has confirmed that the decision tree as a detection method outperforms the existing outlier detection not only for the frog-boiling, network-partition, or complex attack strategies but also for the inflation, deflation, and oscillation attacks. In future work, we plan on further refining the defense and attack strategies by using a game theoretical model, this will help in finding the most appropriate of the two different decision trees for the different attacks, as they have similar performance. To our knowledge, this is the first work that is capable of mitigating all known attacks against virtual coordinate systems.

References

1. Libsvm – a library for support vector machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.
2. p2psim: A simulator for peer-to-peer protocols. http://pdos.csail.mit.edu/p2psim/.
3. Planetlab: An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org.
4. Weka—machine learning software in java. http://sourceforge.net/projects/weka/.
5. V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P systems co-operate for improved performance? *ACM SIGCOMM Computer Communications Review (CCR)*, 37(3):29–40, July 2007.
6. M. A.Kaafar, L. Mathy, T. Turletti, and W. Dabbous. Real attacks on virtual networks: Vivaldi out of tune. In *Proc. of LSAD*, 2006.
7. D. Bolzoni, S. Etalle, and P. H. Hartel. Panacea: Automating attack classification for anomaly-based network intrusion detection systems. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 1–20, 2009.
8. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, California, 1984.
9. C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.
10. E. Chan-tin, D. Feldman, and Y. Kim. The frog-boiling attack: Limitations of anomaly detection for secure network coordinate systems. In *SecureComm*, 2009.
11. E. Chan-Tin and N. Hopper. Accurate and provably secure latency estimation with treeple. In *NDSS*, 2011.
12. B. Cohen. Incentives build robustness in BitTorrent. In *Proc. of P2P Economics*, 2003.
13. M. Costa, M. Castro, R. Rowstron, and P. Key. PIC: practical Internet coordinates for distance estimation. In *Proc. of ICDCS*, 2004.
14. G. F. Cretu-Ciocarlie, A. Stavrou, M. E. Locasto, and S. J. Stolfo. Adaptive anomaly detection via self-calibration and dynamic updating. In *RAID*, 2009.
15. F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *Proc. of ACM SIGCOMM*, 2004.
16. B. Donnet, B. Gueye, and M. A. Kaafar. A survey on network coordinates systems, design and security. *IEEE Communications Surveys and Tutorials*, 2009.
17. P. Francis, S. Jamin, C. Jin, Y. Jin, D.y Raz, Y. Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Trans. Netw.*, 9:525, 2001.

18. K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proc. of ACM SIGCOMM-IMW*, 2002.
19. I. U. Haq, S. Ali, H. Khan, and S. A. Khayam. What is the impact of p2p traffic on anomaly detection? In *Proceedings of the 13th international conference on Recent advances in intrusion detection*, RAID'10, pages 1–17, 2010.
20. M. A. Kaafar, L.Mathy, C. Barakatand Kave Salamatian, T. Turletti, and W. Dabbous. Securing internet coordinate embedding systems. In *Proc. of SIGCOMM*, 2007.
21. M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous. Virtual networks under attack: Disrupting internet coordinate systems. In *Proc. of CoNext*, 2006.
22. A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM*, 2004.
23. L. Lehman and S. Lerman. Pcoord: Network position estimation using peer-to-peer measurements. In *Proc. of NCA*, 2004.
24. L. Lehman and S. Lerman. A decentralized network coordinate system for robust internet distance. In *Proc. of ITNG*, 2006.
25. F. Maggi, W. Robertson, C. Kruegel, and G. Vigna. Protecting a moving target: Addressing web application concept drift. In *RAID*, 2009.
26. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *Proc. of INFOCOM*, 2002.
27. T.S.E. Ng and H. Zhang. A network positioning system for the internet. In *Proc. of USENIX*, 2004.
28. M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris. Lighthouses for scalable distributed location. In *Proc. of IPTPS*, 2003.
29. J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
30. I. Rimac, V. Hilt, M. Tomsu, V. Gurbani, and E. Marocco. A Survey on Research on the Application-Layer Traffic Optimization (ALTO) Problem. RFC 6029 (Informational), October 2010.
31. B. I.P. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S. Lau, S. Rao, N. Taft, and J. D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *IMC*, 2009.
32. M. Sherr, M. Blaze, and B. Thau Loo. Veracity: Practical secure network coordinates via vote-based agreements. In *Proc. of USENIX ATC*, 2009.
33. R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. *Security and Privacy, IEEE Symposium on*, 0:305–316, 2010.
34. M. Steiner and E. W. Biersack. Where is my peer? evaluation of the vivaldi network coordinate system in azureus. In *NETWORKING*, 2009.
35. L. Tang and M. Crovella. Virtual landmarks for the internet. In *Proc. of SIGCOMM*, 2003.
36. V Vapnik and A Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24(6):774–780, 1963.
37. D. Zage and C. Nita-Rotaru. On the accuracy of decentralized network coordinate systems in adversarial networks. In *Proc. of CCS*, 2007.