Tracking Relative Errors in Internet Coordinate Systems by a Kalman Filter

M.A. Kaafar[1], L. Mathy[2], K. Salamatian[2], C. Barakat[3], T. Turletti[3] and W. Dabbous[3]
[1]University of Liege,
[2]Lancaster University,
[3]INRIA Sophia-Antipolis,
[1]Belgium
[2]UK
[3]France

1. Introduction

Internet Coordinate Systems, shortly ICS, (e.g. [9] [8]) have been proposed to allow for distance (Round-Trip Time, shortly RTT) estimation between nodes, in order to reduce the measurement overhead of many applications and overlay networks. Indeed, by embedding the Internet delay space into a metric space – an operation that only requires each node in the system to measure delays to a small set of other nodes (its neighbors), nodes are attributed coordinates that can then be used to estimate the RTT between any two nodes, without further measurements, simply by applying the distance function associated with the chosen metric space to the nodes' coordinates.
Recent works have shown how coordinate-embedding services could be vulnerable to malicious attacks, providing a potentially attractive fertile ground for the disruption or collapse of the many applications and overlays that would use these services [2]. There are actually two obvious ways to disrupt the operation of a coordinate based system. First when requested to give its coordinate for a distance estimation at the application-level, a malicious node could simply and blatantly lie. Second, a malicious node, or even a colluding group, may aim at disrupting the embedding process itself. This latter strategy is very insidious and effective as it can result in important distortions of the coordinate space which then spoils the coordinate computations of many nodes (malicious and honest alike) [2]. This chapter focuses on developing and studying generic Kalman filter-based methods to secure the coordinate embedding process. More precisely, the embedding process, regardless of the actual coordinate-based positioning system, works on the premise that nodes adjust their coordinate based on some comparison between measured and estimated distances to some other nodes. Malicious nodes can interfere with this embedding process by, amongst other things, lying about their real coordinate and/or tampering with measurement probes, to create a discrepancy between measured and estimated latencies, so that unsuspecting nodes would wrongly adjust their own coordinate in a bid to reduce the difference [1]. Because the load on the network naturally varies in time, so does latency between pair of nodes, and as a

Source: Kalman Filter, Book edited by: Vedran Kordić,
ISBN 978-953-307-094-0, pp. 390, May 2010, INTECH, Croatia, downloaded from SCIYO.COM

result, the embedding process must be run periodically by all nodes to track changes in network conditions. This "continuous" adjustment of nodes' coordinates can not only result in a drift of the coordinate space [10] but also gives plenty of scope and opportunities for malicious activity. We therefore seek to equip (honest) nodes with a means to detect, with low overhead, malicious activities they may encounter during embedding.

Noting that, in the absence of malicious nodes, a node's coordinate depends on the combination of network conditions and the specificities of the embedding process itself (e.g. which coordinate protocol is in use, the chosen dimensionality of the geometric space, etc), we therefore introduce the concept of *Surveyor nodes* (or *Surveyors* in short). Surveyors form a group of trusted (honest) nodes, scattered across the network, which use each other *exclusively* to position themselves in the coordinate space. Of course, Surveyors do assist other nodes in their positioning (as prescribed by the embedding protocol), but we stress that Surveyors never rely on non-Surveyor nodes to compute their own coordinate. This strategy thus allows Surveyors to experience and learn the natural evolution of the coordinate space, as observed by the evolution of their own coordinate, in the absence of malicious activities. In essence, Surveyor nodes are thus vintage points guaranteed to be immune from malicious activities. The idea is that Surveyors can then share a "representation" of normal behavior in the system with other nodes to enable them to detect and filter out abnormal behavior.

We postulate and verify that, in the absence of malicious activity, a node's coordinate can be viewed as a stochastic process with linear dependencies whose evolution can be tracked by a Kalman filter [4]. Each Surveyor then computes and calibrates the parameters of a linear state space model and shares the parameters of this model with other nodes. These nodes can then use these parameters, to run locally and in a "standalone" fashion a Kalman filter tracking the coordinate adjustments. These nodes can then use the Kalman filter output (the innovation process), to compare their observed coordinate adjustments with the one predicted by the Kalman filter, and flag as "suspicious" embedding steps where the difference would be too high.

In section 2, we present a general model of coordinate embedding, in the absence of malicious nodes, that naturally leads to the Kalman filter framework. In section 3, we validate the model, with both simulations and PlanetLab experiments, in the case of both Vivaldi [9] and NPS [8]. This section also studies the viability of the idea of using Surveyor nodes in secure coordinate embedding. We then describe and evaluate, in sections 4 and 5, how Surveyors can effectively be used for malicious node detection in the specific embedding process of Vivaldi and NPS.

2. Coordinate embedding model

The goal of embedding systems, regardless of the embedding method and geometric space used, is to assign a coordinate to every node in the system so that, at any time, the distance between any two points in the geometric space should provide a good estimate of the network distance, measured as an RTT (Round Trip Time), between the corresponding nodes. Obviously, because at any instant in time, the RTT that can be measured between two nodes depends on the state of the network (e.g. traffic load, state of queues in routers, etc) as well as the state of the operating system in nodes (e.g. scheduling state generating measurement noise, etc), the exact value of the RTT varies continuously. However, it has been shown that RTT values in the Internet exhibit some stability in a statistical sense [14], with the statistical properties of RTTs exhibiting no significant change at timescales of several minutes. It is that property that embedding systems exploit to provide good distance estimates while only needing to have nodes adjust (recalculate) their coordinate on a periodic basis. Consequently, the coordinate of a node can be viewed as a discrete stochastic process, and we will use X_i^n to represent the coordinate of node *i* at "discrete time" *n*.

Without loss of generality, consider that a node (called the embedding node) computes its coordinate through a series of embedding steps, where each embedding step represents a coordinate adjustment based on a one-to-one interaction with another node, called a peer node (e.g. peer nodes are called neighbors in Vivaldi, and landmarks or reference points in NPS). Note that when the embedding protocol requires that a node uses several peer nodes simultaneously for repositioning, for the purpose of our modelization, we simply consider that each peer node corresponds to a distinct embedding step, each taking place at "successive" discrete times.

At every embedding step, the "fitness" (or "correctness") of the embedding node coordinate is assessed by computing the deviation between the measured RTT towards the corresponding peer node and the one estimated in the coordinate system. More precisely, suppose that at its nth embedding step, embedding node *i* has current coordinate X_i^n and uses peer node *j* with current coordinate X_j^n. Suppose that the RTT between these nodes, measured during this embedding step, is RTT_{ij}^n. The fitness of the embedding node coordinate can then be computed as the *measured relative error* $D_n = \frac{||| X_i^n - X_j^n || - RTT_{ij}^n|}{RTT_{ij}^n}$. The goal of *any* embedding system, regardless of the embedding method proposed and/or the geometric space structure, is to minimize a "cost" indicator (e.g. mean square error) that captures the measured relative error that could be observed between any node and any other node in the system, at any time.

As the measured relative errors are fundamental performance indicators to all embedding systems, it seems natural to develop a model that captures their dynamic characteristics, although we note that relative errors often have complex behavior (and may thus not be a natural choice from a modeling perspective).

Measured relative errors are subject to fluctuations of the RTT for the reasons mentioned above, namely transient network congestion and operating system scheduling issues. To isolate the impact of these RTT fluctuations on anomaly detection, we introduce Δ_n, the *nominal relative error* that our node under consideration would have obtained at its nth embedding step if the RTTs in the network had not fluctuated. An anomaly becomes simply a large deviation of measured relative error D_n from its nominal value defined by Δ_n.

Because many sources contribute to the deviation of D_n from its nominal value (RTT measurement error, RTT fluctuations, errors in node coordinate), it is reasonable to suppose that they relate to each other as follows,

$$D_n = \Delta_n + U_n \tag{1}$$

where U_n is a Gaussian random variable with mean zero and variance v_U. We now focus on the dynamics of the system in its nominal regime where RTTs do not fluctuate. In the absence of complete and accurate knowledge of the system, nodes keep on adapting the nominal relative error on a pairwise basis with their peer nodes, aiming to optimize the cost indicator. This adaptation is subject to an error caused by the other nodes in the system adapting their coordinate (and corresponding relative error) in a completely distributed way. We thus define the *system error* W_n which represents the impact of other nodes on the positioning of a node at embedding step n. Since the system error at a node results from many contributing sources, it is also reasonable to assume that it is a white gaussian process (with mean $\bar{w}$ and variance v_W)[1].

Because of the nature of large-scale embedding processes, the nominal relative error Δ_n can be deemed to follow a stochastic process that converges to some stationary regime characterized by a positive average. As a first approximation, the process Δ_n could be modeled as a first order Auto Regressive (AR) model:

$$\Delta_{n+1} = \beta \Delta_n + W_n. \tag{2}$$

where β is a constant factor strictly less than one otherwise the relative error does not converge to a stationary regime independently of the initial condition. This equation captures the dynamic evolution of the nominal relative error of a node through successive embedding steps.

Equations 2 and 1 define a linear state space model for the relative error of a node. Our goal is to devise a way to obtain relative error predictions from this model. Because of the linear properties of the model, a Kalman filter can be used to track the evolution of the nominal relative error and obtain a *predicted relative error* $\hat{\Delta}_{n|n-1}$ (see section 2.1). However, it is important to notice at this level that tracking relative errors using the Kaman filter would be relevant, only if we consider the embedding systems at their permanent regime. A permanent regime is typically reached when the average relative errors of nodes involved in the system stabilize enough, so that coordinates are considered to be usable by nodes.

The idea behind the strategy we mention above, is that if the stochastic space model, and especially its associated Kalman filter, are calibrated within a clean embedding system, then a simple hypothesis test can be used to assess whether the deviation between the measured relative error and the predicted relative error, observed at a given embedding step, is normal or is the fruit of anomalous or malicious activity from the peer node. From this perspective, even if the state space model considered is crude, its quality should be evaluated based on the final outcome in terms of probability of detection and false positive rate. We will see in the evaluation section (section 5) that this model achieves very good performance.

2.1 Kalman filter equations

The Kalman filter is used here to estimate Δ_n given the set of previously measured relative errors $\mathcal{D}_0^n = \{D_0, \ldots, D_n\}$. Under the hypothesis of a gaussian noise process in the underlying state space model, the Kalman filter gives the Least Mean Squared estimates of Δ_n, $\hat{\Delta}_n$. Moreover, it gives the quality of these estimates through an evaluation of the mean squared error i.e., $E[(\hat{\Delta}_n - \Delta_n)^2]$. This last value could be used to detect anomalies through large deviations of the measured relative error from its mean.

We will assume here that all the parameters of the space model given in Eq. (1) and Eq. (2) are known and given. In the next section we will describe how to derive these parameters.

Let us denote by $\hat{\Delta}_{i|i-1}$ the estimation of $¢_i$ knowing the observations of network delay up to time $i-1$, and $\hat{\Delta}_{i|i}$ the estimate after the measurement D_i is done. Similarly, let $P_{i|i-1}$ be the

[1] The value $\bar{w}$ accounts for the drift that has been observed in positioning systems [10].

estimated *a posteriori* error variance at time i knowing the observations up to time $i-1$, and let $P_{i|i}$ be the estimation of the *a posteriori* error variance after D_i is known. The Kalman Filter is composed of two steps that are iterated. The first step is called the prediction step and the second one the update step.
In the prediction step, the value of $\hat{\Delta}_{i|i-1}$ is calculated based on $\hat{\Delta}_{i-1|i-1}$ as:

$$\hat{\Delta}_{i|i-1} = \beta\hat{\Delta}_{i-1|i-1} + \bar{w}.$$

The *a posteriori* error variance of this estimate is:

$$P_{i|i-1} = \beta^2 P_{i-1|i-1} + v_W.$$

In the update step, $\hat{\Delta}_{i|i-1}$ is updated to integrate the observed measurement D_i:

$$\hat{\Delta}_{i|i} = \hat{\Delta}_{i|i-1} + K_i(D_i - \hat{\Delta}_{i|i-1})$$

where K_i denotes the updated gain and is obtained as:

$$K_i = \frac{P_{i|i-1}}{P_{i|i-1} + v_U}.$$

The *a posteriori* error variance of this estimate is :

$$P_{i|i} = \frac{v_U}{P_{i|i-1} + v_U} P_{i|i-1}.$$

The value $\eta_i = D_i - \hat{\Delta}_{i|i-1}$ is called the innovation process and is the main process to observe for anomalous behavior detection (see section 4.1). The innovation process is a white (meaning that it is an independent process) gaussian process with a mean 0 and a variance equal to $v_{\eta,i} = v_U + P_{i|i-1}$. Abnormality simply amounts to a significant deviation from the nominal values of the innovation process characterized by the Kalman filter.
To run the Kalman estimation, we need as initial values the system state value w_0 and the *a priori* state variance $P_{0|0} = p_0$. These two values are estimated during the parameters calibration step.

2.2 Calibration of the Kalman filter

Before running the estimation using the Kalman filter, the values of the filter parameters $\theta = (\beta, v_W, v_U, w_0, p_0)$ have to be computed. For this purpose we need to calibrate these parameters over coordinate measurements collected during a stationary and cheater-free period. The calibration can be done using a maximum likelihood criteria (choosing parameter values such that the likelihood of observing the measurements is maximized) by applying the Expectation Maximization (EM) method. We follow the approach presented in [15] for the EM derivation.
In the following, we give a brief description of the maximum likelihood estimation criterion and the EM method, we are using in this section to calibrate our filter.
The maximum likelihood estimation criterion Maximum likelihood estimation (MLE) is a statistical method used to make inferences about parameters of the underlying probability distribution from a given data set [16]. Basically, this method allows us to infer the parameters of a distribution given a sample of data $X = X_1, \ldots, X_n$. Commonly, one assumes

the data are independent, identically distributed (i.i.d) drawn from a particular distribution with unknown parameters and uses the MLE technique to create estimators for these unknown parameters.
Let us consider a family of distributions P_θ indexed by a parameter (which could be a vector of parameters) θ that belongs to a set Θ. Let $f(X \mid \theta)$ be either a probability function (in case of discrete distribution) or a probability density function (continuous case) of the distribution P_θ. Given our i.i.d. sample $X_1, \ldots, X_n$ with unknown distribution P_θ from this family, i.e. parameter θ is unknown. A likelihood function is defined by:

$$L(\theta|X) = f(X_1|\theta) \times \ \ldots \ \times f(X_n|\theta).$$

If our distributions are discrete then the probability function $f(x \mid \theta) = P_\theta(X = x)$ is the probability to observe a point x. $L(\theta) = f(X_1 \mid \theta) \times \ldots \times f(X_n \mid \theta) = P_\theta(X_1) \times \ldots P_\theta(X_n) = P_\theta(X_1, \ldots, X_n)$ is the probability to observe the sample $X_1, \ldots, X_n$ when the parameters of the distribution are equal to θ.
Suppose that there exists a parameter $\hat{\theta}$ that maximizes the likelihood function $L(\theta)$ on the set of possible parameters, i.e.

$$L(\hat{\theta}) = \max_{\theta \in \Theta} L(\theta)$$

Then $\hat{\theta}$ is called the Maximum Likelihood Estimator (MLE) of θ.
When finding the MLE, it is sometimes easier to maximize the log-likelihood function since

$$L(\theta) \rightarrow maximize \Leftrightarrow log(L(\theta)) \rightarrow maximize$$

maximizing $L(\theta)$ is equivalent maximizing log $L(\theta)$. Log-Likelihood function can be written as log $L(\theta) = \sum_{i=1}^{n}(\log f(X_i|\theta))$.
To summarize, one needs to recall that the MLE criterion chooses the parameter $\hat{\theta}$ that maximizes the probability of seeing the observed data given that their distribution follows these parameters. The log of the likelihood is often used instead of true likelihood because it leads to easier formulas, but still attains its maximum at the same point as the likelihood.
The Expectation maximization (EM) algorithm is a valuable approach for maximum likelihood parameter estimation. In the next paragraph, we use this method to calibrate our Kalman filter parameters, i.e . find the values for the filter parameters that maximize the probability of a sequence of observations.
Calibration by EMmethod Let's assume that $\mathcal{D}_0^N$ is the set of all measured prediction errors, $\mathcal{D}_0^N = \{D_0, \ldots, D_N\}$ and let $\mathbf{\Delta}_0^N = \{\Delta_0, \ldots, \Delta_N\}$ be the set of nominal relative errors.
As all the noise processes are assumed to be gaussian, $\mathcal{D}_0^N$ and $\mathbf{\Delta}_0^N$ will jointly follow a gaussian distribution. The log-likelihood of $\mathcal{D}_0^N$ and $\mathbf{\Delta}_0^N$, based on equations 2 and 1, can therefore be written as follows:

$$\begin{aligned}\log \text{Prob}\{\mathcal{D}_0^N, \mathbf{\Delta}_0^N\} &= -\sum_{i=0}^{N} \frac{(D_i - \Delta_i)^2}{2v_U} \\ &- \frac{N+1}{2}\log v_U - \sum_{i=1}^{N} \frac{(\Delta_i - \beta\Delta_{i-1} - \bar{w})^2}{2v_W} \\ &- \frac{N}{2}\log v_W - \frac{(\Delta_0 - w_0)^2}{2p_0} - \frac{1}{2}\log p_0 \\ &- (N+1)\log 2\pi.\end{aligned}$$

In a Maximum likelihood setting, we wish to find the values for the parameters that will maximize the above log-likelihood assuming that the sequence $\mathcal{D}_0^N$ has been observed. However as the sequence of system state $\mathbf{\Delta}_0^N$ has not been observed, this maximization is not tractable directly and we have to apply the Expectation Maximization method [17]. This method transforms the maximization of the above likelihood function with unobserved system state sequence $\mathbf{\Delta}_0^N$ to an iteration of successive steps where the system state sequence is assumed to be known and the parameters can be obtained through maximization of the likelihood function.

Each iteration of the EM method consists therefore of two steps. In the first step, we compute the expectation (over all values of the sequence of states $\mathbf{\Delta}_0^N$) of the loglikelihood, given the observed values of D_n and assume that the parameter values are equal to $\theta^{(k)}$. In a second step, the parameters $\theta^{(k+1)}$ are chosen so as to maximize the previously obtained likelihood expectation. Next we explain these two operations with some further details.

Let the superscript (k) indicate the value of any parameter at the kth step of the EM algorithm. As explained before, in the EM method, we need to estimate the value of the unobserved system states to be able to calculate the overall likelihood to maximize. The variables $\hat{\delta}_i^{(k)}$ are in fact those estimates at iteration k and $\hat{\pi}_i^{(k)}$ and $\hat{\pi}_{i,i-1}^{(k)}$ are the estimation error variances of this sequence of states:

$$\hat{\delta}_i^{(k)} = E[\Delta_i|\mathcal{D}_0^N, \theta^{(k)}],\ \hat{\pi}_i^{(k)} = E[\Delta_i^2|\mathcal{D}_0^N, \theta^{(k)}],$$

$$\hat{\pi}_{i,i-1}^{(k)} = E[\Delta_i \Delta_{i-1}|\mathcal{D}_0^N, \theta^{(k)}].$$

Expectation step The expected value of log-likelihood knowing the set of measured values $\mathcal{D}_0^N$ and the parameter $\theta^{(k)}$ is given by:

$$\begin{aligned}\bar{L}(\theta, \theta^{(k)}) &= E[\log \text{Prob}\{\mathcal{D}_0^N, \mathbf{\Delta}_0^N\}|\mathcal{D}_0^N, \theta^{(k)}] \\ &= -\sum_{i=0}^{N} \frac{D_i^2 - 2D_i\hat{\delta}_i^{(k)} + \hat{\pi}_i^{(k)}}{2v_U} \\ &- \frac{N+1}{2}\log v_U - \sum_{i=1}^{N} \frac{\hat{\pi}_i^{(k)} + \beta^2\hat{\pi}_{i-1}^{(k)} + \bar{w}^2}{2v_W} \\ &+ \sum_{i=1}^{N} \frac{\beta\hat{\pi}_{i,i-1}^{(k)} + \hat{\delta}_i^{(k)}\bar{w} - \beta\hat{\delta}_{i-1}^{(k)}\bar{w}}{v_W} \\ &- \frac{N}{2}\log v_W - \frac{\pi_0 - 2\hat{\delta}_0^{(k)}w_0 + w_0^2}{2p_0} \\ &- \frac{1}{2}\log p_0 - (N+1)\log 2\pi.\end{aligned}$$

By replacing θ by its value at the kth step of the EM algorithm, we obtain $\hat{\delta}_i^{(k)}$, $\hat{\pi}_i^{(k)}$ and $\hat{\pi}_{i,i-1}^{(k)}$, which gives the expected log-likelihood at the (k +1)th step. Next, we describe how to compute these values.

Calculating the parameters $\hat{\boldsymbol{\delta}}_{\boldsymbol{i}}, \hat{\boldsymbol{\pi}}_{\boldsymbol{i}}, \hat{\boldsymbol{\pi}}_{\boldsymbol{i,i-1}}$ As explained in section 2.2, the sequence of system states $\mathbf{\Delta}_0^N$ is not observable. However, we need to give an estimate of this sequence to be able to obtain the likelihood. The sequence $\hat{\delta}_i$, $i = 1 \dots ,N$, is the sequence of system state estimates and $\hat{\pi}_i$, and $\hat{\pi}_{i,i-1}$ is the error variance of these estimates assuming that the sequence $\mathcal{D}_0^N$ has been observed. We resort to the solution in [15] for the calculation of these estimates using the overall measurements set.

The value $\hat{\delta}_i$, $\hat{\pi}_i$, and $\hat{\pi}_{i,i-1}$ are estimated using a Kalman filter, assuming that the system parameters are set as in $\theta_{(k)}$. However, there is a subtle difference with Kalman filter case described in section 2.1; here the estimates $\hat{\delta}_i$, $\hat{\pi}_i$ and $\hat{\pi}_{i,i-1}$ do not depend only on observations up to time i, but on future observations up to time $N \geq i$. The solution to deal with this is to implement a forward-backward approach similar to Baum- Welch filter used for finite EM algorithm [20].

For each value of the parameter set $\theta^{(k)}$, we first do a forward step following the relations given in section 2.1. The application of this forward step results in the values $\hat{\Delta}_{i|i}$, $i = 1, \dots ,N$, $P_{i|i}$, $i = 1, \dots ,N$ and $P_{i|i-1}$, $i = 1, \dots ,N$.

To add the future measurements in the Kalman filter, a backward recursion step is also added. This step consists of the following equations:

$$\begin{cases} \hat{\delta}_{i-1} = \hat{\Delta}_{i-1|i-1} + J_{i-1}(\hat{\delta}_i - \beta\hat{\Delta}_{i-1|i-1}) \\ \hat{\pi}_{i-1} = P_{i-1|i-1} + J_{i-1}^2(\hat{\pi}_i - P_{i|i-1}) \\ J_{i-1} = \beta\frac{P_{i-1|i-1}}{P_{i|i-1}} \end{cases}$$

These equations give recursively the values $\hat{\delta}_i$ and $\hat{\pi}_i$. It still remains to obtain $\hat{\pi}_{i,i-1}$.
This last value could be obtained using the relation:

$$\hat{\pi}_{i,i-1} = v_{i,i-1}^N + \hat{\delta}_i\hat{\delta}_{i-1}$$

where $v_{i,i-1}^N$ can be obtained through the backward recursion

$$v_{i,i-1}^N = P_{i-1|i-1}J_{i-1} + J_i(v_{i+1,i}^N - \beta P_{i|i})J_{i-1},$$

that is initialized by setting

$$v_{N,N-1}^N = (1 - K_N)\beta P_{N-1|N-1}.$$

Maximization step In this step, the parameter vector at step (k + 1) is chosen to maximize the expected log-likelihood. This is done by solving the equation

$$\frac{\partial \bar{L}(\theta, \theta^{(k)})}{\partial\theta} = 0.$$

This results in the following set of equations:

$$\begin{cases} w_0^{(k+1)} &= \hat{\delta}_0^{(k)} \\ p_0^{(k+1)} &= \hat{\pi}_0^{(k)} - (\hat{\delta}_0^{(k)})^2 \\ v_U^{(k+1)} &= \frac{1}{N+1} \sum_{i=0}^{N} D_i^2 - 2D_i\hat{\delta}_i^{(k)} + \hat{\pi}_i^{(k)} \\ \bar{w}^{(k+1)} &= \sum_{i=1}^{N} \hat{\delta}_i^{(k)} - \beta^{(k+1)} \sum_{i=1}^{N} \hat{\delta}_{i-1}^{(k)} \\ \beta^{(k+1)} &= \frac{\sum_{i=1}^{N} \hat{\pi}_{i,i-1}^{(k)} - \bar{w}^{(k+1)} \sum_{i=1}^{N} \hat{\delta}_{i-1}^{(k)}}{\sum_{i=1}^{N} \hat{\pi}_{i-1}^{(k)}} \\ v_W^{(k+1)} &= \frac{1}{N} \sum_{i=1}^{N} \hat{\pi}_i^{(k)} + (\beta^{(k+1)})^2 \hat{\pi}_{i-1}^{(k)} + (\bar{w}^{(k+1)})^2 \\ -2\beta^{(k+1)}\hat{\pi}_{i,i-1}^{(k)} - 2\hat{\delta}_i^{(k)}\bar{w}^{(k+1)} + 2\beta^{(k+1)}\hat{\delta}_{i-1}^{(k)}\bar{w}^{(k+1)}. \end{cases}$$

By solving this set of equations we can obtain the vector $\theta^{(k+1)}$, then we iterate with the expectation calculation as described above.
We note that the complexity of the approach lies in the linear state space modeling phase by EM algorithm that incurs a number of iterations over N dimensional vectors, which is well within the capability of modern computers. We will see later that this phase has to be run on a subset of nodes (the Surveyors). On the other hand, predicting relative errors using the Kalman filter(section 2.1), which occurs on every node, only implies a few simple scalar operations and is negligible in terms of required computing power.
Finally, because we expect each of the innovation observation η_n to be inside a confidence interval of $\pm 2\sqrt{v_{\eta,n}}$ (where $v_{\eta,n}$ is the variance of the innovation process at time n) with a probability higher than 95%, when a Kalman filter yields 10 consecutive innovation observations outside such confidence interval, the filter is re-calibrated by re-applying the calibration procedure described in this section. Re-calibration is likely to occur following a significant change in the corresponding node's coordinate, caused by changes in network conditions.

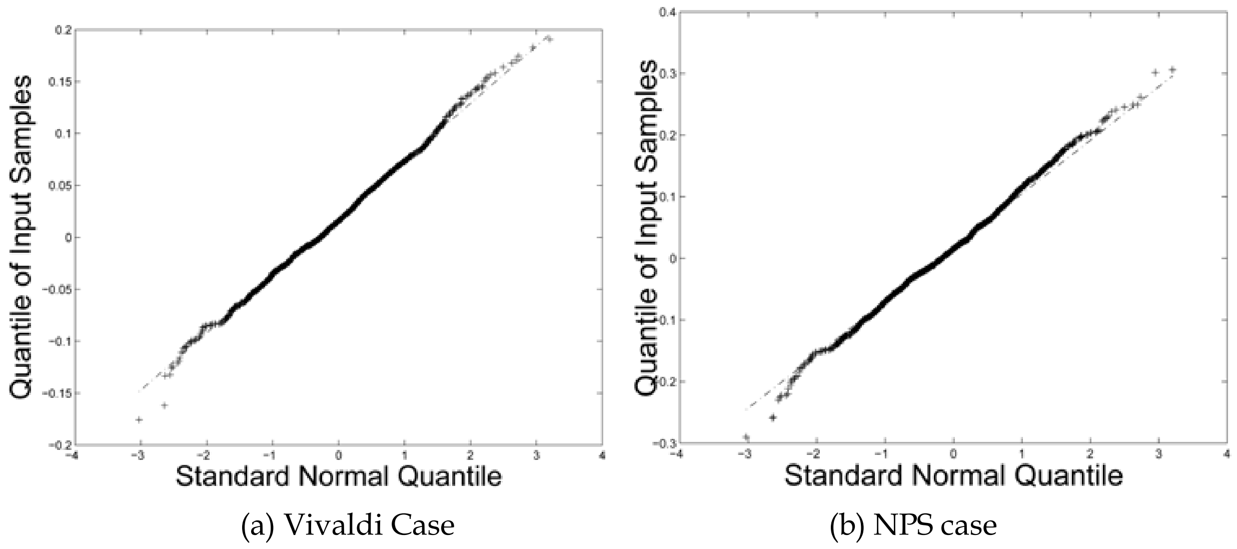
3. Validation

To validate our model, we conducted simulations and PlanetLab experiments for both Vivaldi [9] and NPS [8]. We consider Vivaldi as a prominent representative of purely peer-to-peer-based (i.e. without infrastructure support) positioning systems, while NPS is typical of infrastructure-based systems, where a hierarchy of landmarks and reference points govern the positioning of nodes.
As the goal of this section is to assess the fitness of the proposed model to represent the normal behavior of the embedding processes, all results presented were acquired in a clean environment with no malicious node. While the goal of the simulation studies is to assess our results for large scale coordinate-based systems, the PlanetLab experiments aim to show their applicability in real-world conditions. The PlanetLab experiments were conducted over a set of 280 PlanetLab nodes spread world-wide. We discuss a representative set of experimental results conducted over several days in December 2006. The simulations were driven by a matrix of inter-host Internet RTTs (the "King" dataset) to model latencies based on real world measurements. This dataset contains the pair-wise RTTs between 1740 Internet DNS servers collected using the King method [13] and was used to generate a topology with 1740 overlay nodes.
In the case of Vivaldi, each node had 64 neighbors (i.e. was attached to 64 springs), 32 of which being chosen to be closer than 50 ms. The constant fraction C_c for the adaptive

timestep is set to 0.25, as proposed in [9]. A 2-dimensional coordinate space augmented with a height vector was used.
For NPS, we considered an 8-dimensional Euclidean space for the embedding. We used an NPS positioning hierarchy with 4 layers. The top layer had a set of 20 well separated permanent landmarks. Each subsequent layer then had 20% of nodes randomly chosen as reference points. The security mechanism already proposed in NPS, shown to be too primitive in [2], was turned on and its sensitivity constant *C* was set to 4, as advised by authors in [8].
When needed, Surveyor nodes were chosen at random[2]. Again, because we seek to validate the kalman filter model, during the permanent regime of the embedding systems, all our experiments were are performed when nodes coordinates are stabilized enough, i.e. the relative errors vary at most by 0.02.

3.1 Assumption validation

In section 2, the assumption that the system error W_n follows a gaussian distribution was made. This is fundamental to the applicability of the Kalman filter framework. For the purpose of validation, every node calibrated its own Kalman filter based on the observation of its own embedding, and we checked this assumption by applying the Lilliefors test [18], a robust version of the well known kolmogoroff-Smirnov goodnessof- fit test, to whitened filter inputs. We observed that the Lilliefors test leads to only 14 gaussian fitting rejections in simulations (over 1720 samples) and 5 rejections in PlanetLab (over 260 samples). This test allows us to conclude that the hypothesis we took for the Kalmanmodel is valid. In addition, we plot in figure 1 the Quantile-Quantile (QQ) plots of 2 innovation processes (for both Vivaldi and NPS) taken on PlanetLab nodes running their own Kalman filter. These plots, typical of observations on all nodes, show that each of these distributions indeed closely follows a Gaussian distribution.



(a) Vivaldi Case (b) NPS case

Fig. 1. Quantile-Quantile plot of 2 innovation processes.

[2] Note that in NPS, all permanent landmarks also act as Surveyors.

3.2 Effective behavior representation

From section 2, it is clear that the Kalman filter model attempts to represent the behavior of the embedding process by capturing the dynamics of the system through its convergence behavior (by tracking of relative errors over time). In this section, we therefore assess the representational power of this approach by having each node calibrate its own Kalman filter from the measurements it observed during the embedding of its own coordinate, in a cheat-free regime. Then, once the model has converged at every node (i.e. the EM method has converged and the variations of all the θ components become smaller than 0.02), a new embedding process is started (i.e. the nodes forget their coordinates and rejoin the system). During this second embedding process, the prediction error, that is the absolute value between the error predicted by the node's Kalman filter and the measured actual error, is computed.

Figure 2 shows a typical evolution of actual (measured) relative errors and predicted errors for a node on PlanetLab (for Vivaldi, but similar behavior was observed for NPS). The two curves of the top graph of the figure are so similar that they are almost indistinguishable. The bottom graph of the figure represents the prediction error which is the difference between these two curves (note the different scale used for this graph). We see that the prediction errors are small which shows that a node's calibrated Kalman filter can capture effectively the node's behavior "in the wild".

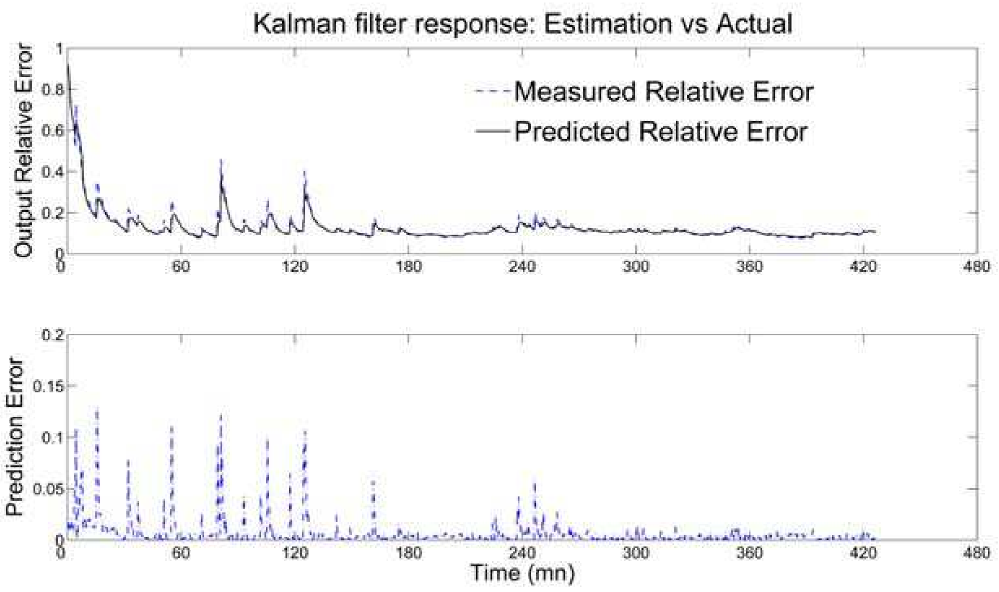


Fig. 2. Prediction errors (PlanetLab node).

Figure 3 shows the cumulative distribution function (CDF) of all the prediction errors observed across all nodes in the system. This confirms that the vast majority of predictions are indeed excellent. This demonstrates the power and generality of the model in capturing the dynamics of the system and its adaptability to current system conditions.

However, there are a few "outlier" predictions with large errors. To better understand their nature, we show in table 1 the repartition of prediction errors in error intervals. The table shows the number of nodes with prediction errors in this interval, the number of occurrences of the smallest prediction error observed in the interval, and the number of occurrences of the largest prediction errors observed in the interval (e.g. Number of node(s)/number of observed min error/number of observed max error).We see that only a few nodes contribute (sometimes very many) large prediction errors. Looking further, we identified 3 nodes, all located in India, who contributed consistently to the "tail" of the CDF in figure 3. It is interesting to note that these nodes exhibited large (> 0.75) average measured relative errors during embedding, and were clearly having trouble with the embedding process itself, due to adverse network conditions.

Error Interval	(Vivaldi)	(NPS)
0.0-0.05	257/830/922	232/854/943
0.05-0.1	32/201/995	44/180/941
0.1-0.15	5/3/992	18/5/229
0.15-0.2	1/997/997	2/12/884
0.4-0.45	4/3/56	3/5/12
0.45-0.5	1/12/12	2/1/17
0.5-0.55	1/985/985	2/32/40
0.6-0.65	-	1/851/851

Table 1. Prediction Error Histogram

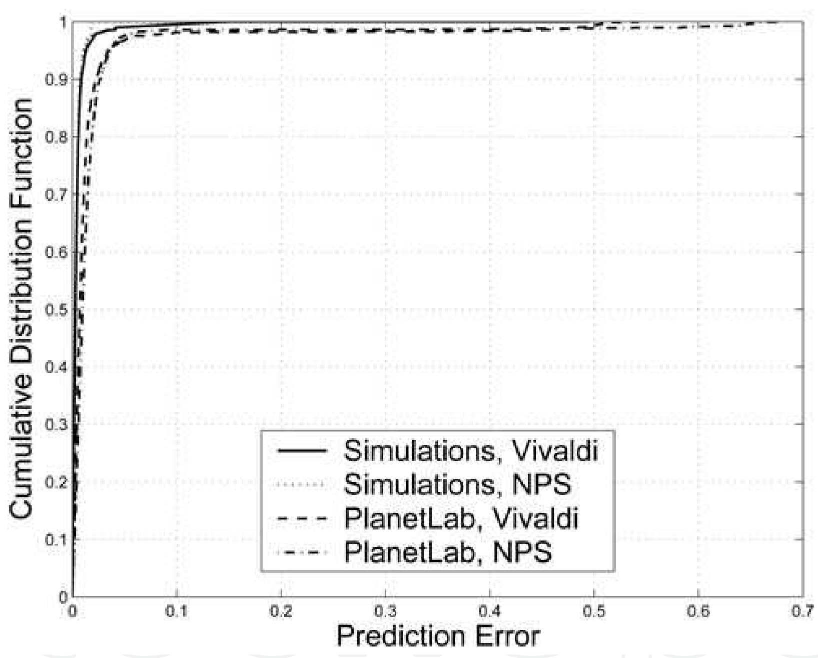


Fig. 3. CDF of prediction errors.

3.3 Representativeness of surveyors

If a subset of nodes in the system (called Surveyor nodes) are trusted and use each other exclusively to compute their own coordinates, they will be immune to the effects of malicious behavior during embedding. The premise of our work is that the "clean" (normal) system behavior thus learnt can then be shared with other nodes and used by these nodes to detect malicious behavior they may be subjected to by other nodes in the system. This obviously assumes that the behavior of the system as observed by Surveyors can approximate or represent well enough the normal behavior of the system as observed by other nodes in the absence of malicious behavior. In the following validation of this assumption, Surveyors are chosen at random in the node population.

Note that a random choice will give an upper bound on the number of Surveyors needed. Indeed, intuitively, Surveyors should be roughly uniformly distributed in the system to be representative of most other nodes. However, choosing nodes at random in the system does not yield a uniform distribution of Surveyors (i.e. "Surveyor clusters" appear due to the cluster structure of nodes themselves) and therefore not every new Surveyor increases representativeness. Consequently, more Surveyors must be chosen in order to achieve a good coverage of the system, than if they were placed more strategically. Nevertheless, the random choice method does provide general results, without the need to address the question of optimal Surveyor deployment strategy.

One of the first questions to answer is how many Surveyors are needed to be representative of the rest of the population. Noting that our model is based on measured relative errors and that each node in the system observes a series (i.e. distribution) of such errors, we characterize the system-wide relative error behavior as the CDF of the 95th percentiles of the relative errors observed at each (normal) node (i.e. the distribution is made up of the 95th percentile value observed at every node). We then compare this CDF with those of the 95th percentiles of the relative errors observed across a varying population of Surveyors. The choice of the 95th percentile is so that outliers, as observed in section 3.2, do not skew the results, while preserving a high degree of generality. Figure 4, obtained by simulations of Vivaldi, indicates that a population of Surveyors of about 8% of the overall population is closely representative of this overall population (because the CDF for these populations in the figure are similar).

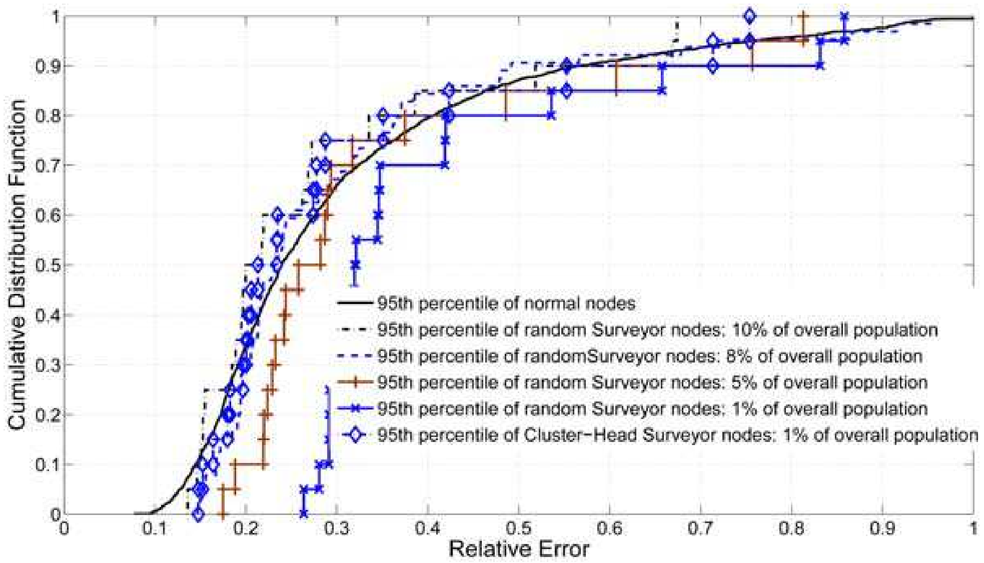


Fig. 4. Impact of Surveyor population size on repretentativeness.

To generalize this result, we then repeated the experiment, using both simulations and PlanetLab measurements, on a Vivaldi system with 8% of Surveyors. We again chose to represent the system by the distribution of the 95th percentile of the measured relative errors (figure 5).

These experiments confirm that less than 10% of randomly chosen Surveyor nodes is enough to gain a good representation of the system behavior. Similar results were observed for NPS. We note that 8% to 10% of the overall node population is a very stringent requirement for most practical purposes and can represent a huge number of nodes. However, as already pointed out above, random Surveyor deployment is not optimal and this value is an upper bound on the number of Surveyors needed.

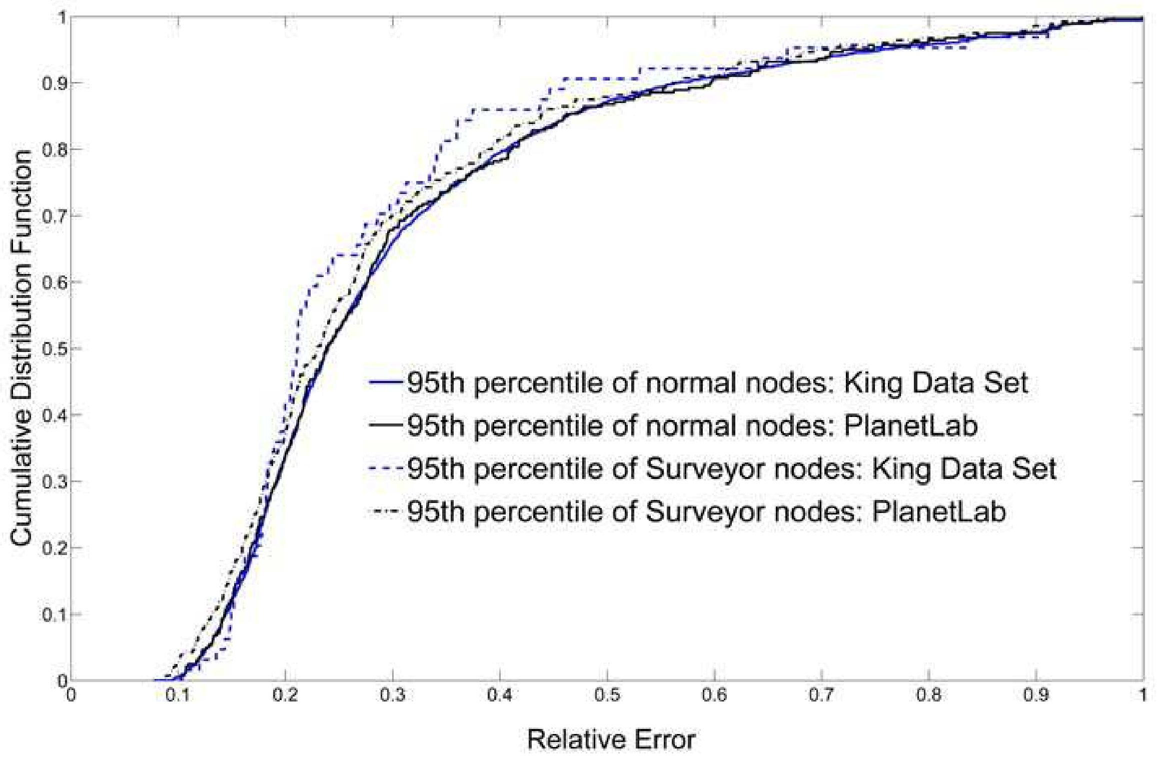


Fig. 5. Representativeness with 8% Surveyor nodes.

To gain further insight into how conservative this upper bound may be, we tried a simple k-means clustering algorithm for Surveyor deployment. Figure 4 shows that when taking cluster heads as Surveyors, good representativeness can be achieved with roughly 1% of Surveyors. Although this does not give much indication as to what the lower bound on the number of Surveyors needed is in the case of optimal Surveyor deployment, it nevertheless shows that simple deployment methods can reduce requirements considerably and that the upper bound yielded by random deployment is indeed very conservative.

Having shown that a population of Surveyors can represent the overall system, the next question is how well the behavior of the system as captured by the Kalman filter calibrated by a Surveyor, can represent the behavior of a single (normal) node. To answer this question, we carried out an experiment where a population of nodes took part in a Vivaldi embedding on PlanetLab. Each node used the Kalman filter of every Surveyor and generated multiple prediction errors (one per Surveyor) at every embedding step.

Figure 6 shows the maximum prediction error yielded by each Surveyor, for each normal node in the system, observed during this experiment. What we observe is that although each normal node can find at least one Surveyor node whose Kalman filter yields very low prediction errors, not every Surveyor is a good representative for any given normal node. The Surveyor chosen as a representative by a normal node is therefore important to achieve good prediction performance (and thus good malicious behavior detection).

Figure 7 plots the prediction accuracy (measured as an average prediction error) against the distance (measured as an RTT) between a node and the corresponding Surveyor, as observed during the PlanetLab experiment. It is clear that better locality between a node and its Surveyor yields more accurate predictions. This property seems intuitive, as a Surveyor closer in terms of RTT will also be closer in the geometric space, and will thus be more likely to experience dynamics of the coordinate system similar to that of the local area where the node resides. This is confirmed in figure 8 which shows the maximum prediction error, observed for Vivaldi on PlanetLab, when nodes use the closest Surveyor as their representative. Similar results were observed for PlanetLab experiments with NPS.

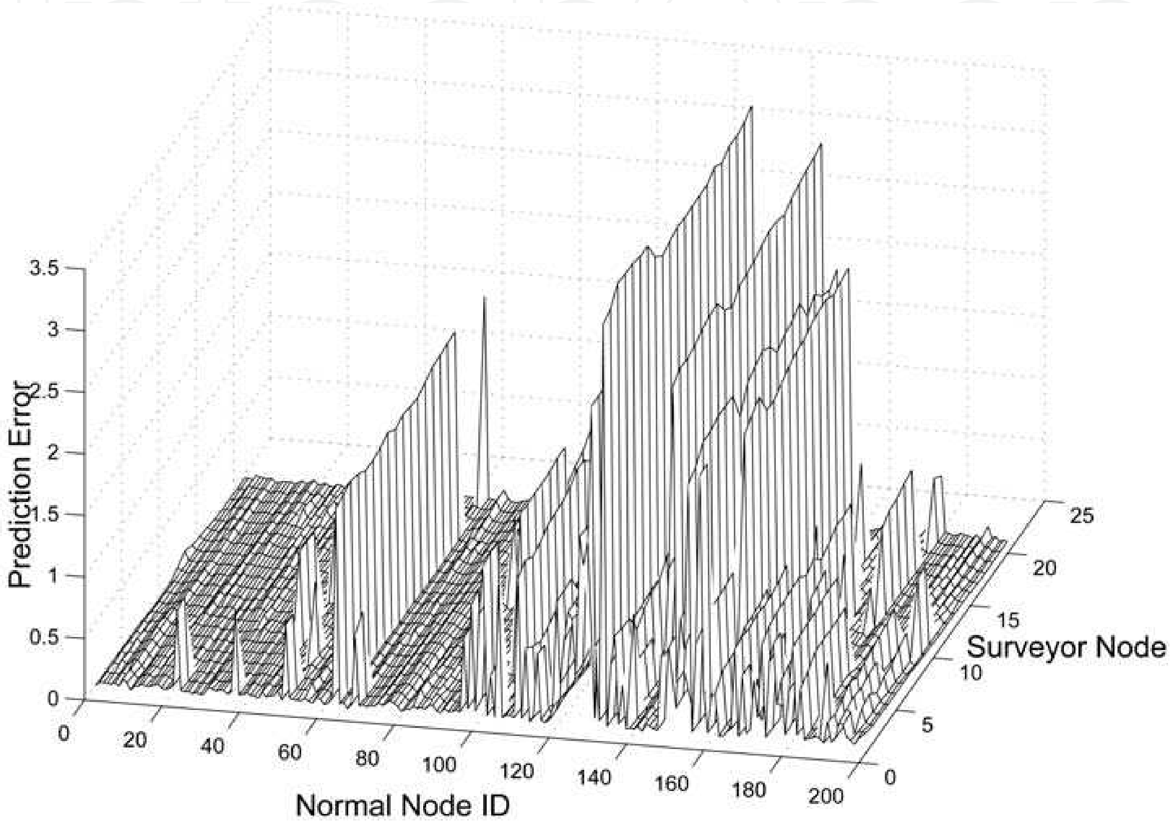


Fig. 6. Maximum prediction errors with Surveyor filter parameters (PlanetLab).

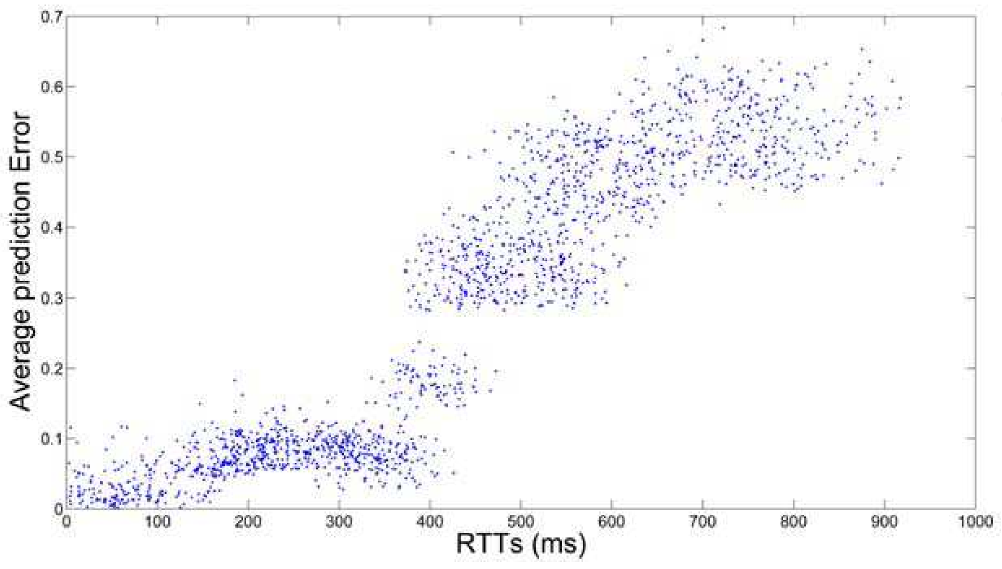


Fig. 7. Correlation between 'Node-Surveyor' RTTs and estimation accuracy (PlanetLab).

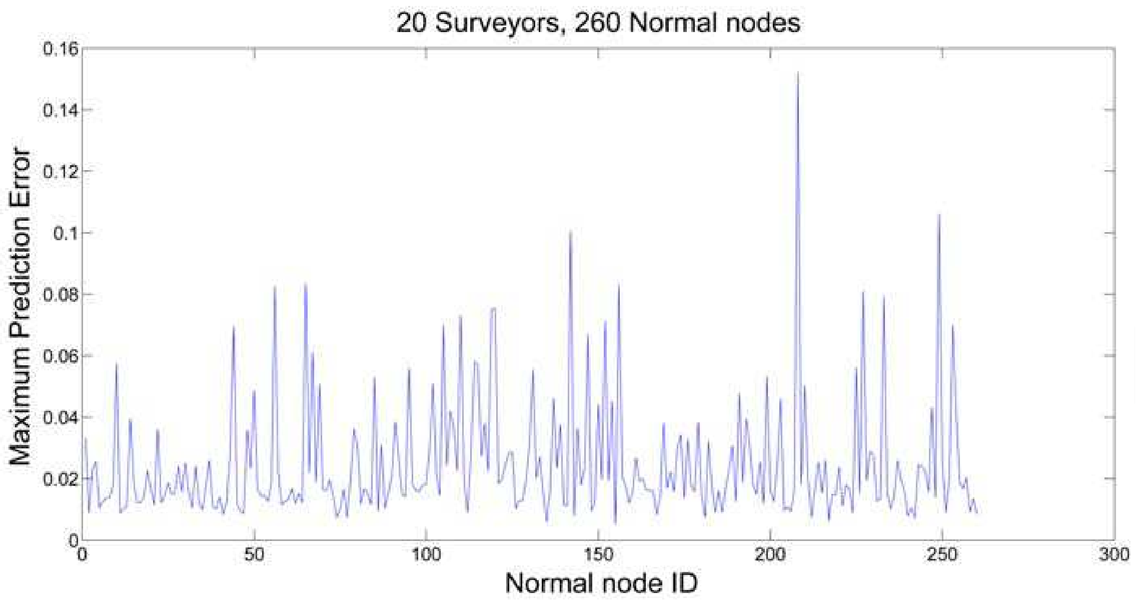


Fig. 8. Maximum prediction errors with closest Surveyor.

Finally, it is again important to note that all the results in this section were obtained with randomly deployed Surveyors. Strategically placing Surveyors to ensure a better coverage of the network and coordinate space, would simply improve the prediction accuracy, while reducing the number of Surveyors required.

4. Malicious behavior detection

The previous section has shown that normal node behavior can be modeled by a Kalman filter. More importantly, it has also been shown that this technique is powerful and robust enough that the normal behavior model captured on one node is readily and effectively applicable on other nearby nodes. This property leads to the idea of Surveyors.

Surveyors are a set of nodes in the coordinate space that exclusively use each other to compute their own coordinates. In other words, in Vivaldi, Surveyors only use other Surveyors as neighbors, while in NPS, they only use other Surveyors as reference points (note that in NPS, all landmarks also act as Surveyors, although not all reference points will be Surveyors). Of course, Surveyors can, and will be chosen as neighbors or reference points by other (non-Surveyor) nodes in the system, but the point is that a Surveyor adjusts its coordinate solely in response to embedding steps (i.e. measurements) with other Surveyors. If Surveyors run a clean version of the coordinate embedding software and they are carefully kept clean of malicious software, such as viruses or worms, that could implement malicious modifications to the embedding, then they can be considered as clean, honest nodes. Because Surveyors only interact with each other during their own embedding, they are therefore immune to malicious or anomalous behavior in the system, and they therefore observe the behavior of the system in clean, normal conditions. The idea is then to use the thus obtained normal behavior model as a basis for anomalous behavior detection at other nodes of the system. To do so, nodes use the parameters of the Kalman filter calibrated at a nearby Surveyor.

It is important to note that the proposed method is entirely distributed as each node has its own filter. Indeed, Surveyors calibrate and recalibrate their own filter as needed, depending on varying network conditions, and share the resulting filter parameters with other nodes, but they take no further active part in anomalous behavior detection at other nodes. When a node's filter needs re-calibrating (e.g. because it starts giving too many detection alarms), the node simply obtains fresh filter parameters from a Surveyor.

4.1 Anomalous behavior detection method

At each embedding step, a node computes a measured relative error D_n towards a peer node. Recall from section 2 that the Kalman filter at the node can provide $\hat{\Delta}_{n|n-1}$, the predicted relative error from the previously measured relative errors. The innovation process of the Kalman filter yields the deviation between the measured and predicted relative errors, $\eta_n = (D_n - \hat{\Delta}_{n|n-1})$, which, in a system without malicious node, follows a zero-mean gaussian distribution with variance $v_{\eta,n} = v_U + P_{n|n-1}$ (also yielded by the filter).
This allows us to detect malicious behavior as a simple hypothesis test. Let H_0 be the hypothesis that the peer node has a normal behavior (i.e. it is honest). The hypothesis testing simply consists of assessing whether the deviation between the measured and predicted relative errors is normal enough under expected system behavior. Given a "significance level" α, which determines the "aggressivity" or "strictness" of the test, the problem is to find the threshold value t_n such that

$$P(|D_n - \hat{\Delta}_{n|n-1}| \geq t_n \mid H_0) = \alpha. \tag{3}$$

But since, under hypothesis H_0, $(D_n - \hat{\Delta}_{n|n-1})$ follows a zero-mean normal distribution with variance $v_{\eta,n}$, we also have that

$$P(|D_n - \hat{\Delta}_{n|n-1}| \geq t_n \mid H_0) = 2Q(t_n/\sqrt{v_{\eta,n}}), \tag{4}$$

where $Q(x) = 1-\Phi(x)$, with $\Phi(x)$ being the CDF of a zero-mean, unit-variance normal distribution.
From equations 3 and 4, we therefore have

$$t_n = \sqrt{v_{\eta,n}}Q^{-1}(\alpha/2). \tag{5}$$

If the observed deviation exceeds the threshold given by equation 5, then the hypothesis is rejected, the peer node is flagged as suspicious, the embedding step is aborted and the measured relative error D_n is discarded (i.e. it is not used to update the state of the filter).
Note that a suspicious node, as detected by this test, is not necessarily associated with malicious intent, but could be caused by changing network conditions. Honest nodes classified as suspicious represent false positives and have little impact on the system as long as their occurrence is low. The trade-off between aggressivity and strictness of the test is represented by the so called ROC (Receiver Operation Characteristic) curves [19]. These curves plot the true positive rate versus the false positive rate, *i.e.* the probability of correctly detecting a malicious node versus the probability of labelling an honest node as malicious. In practice trying to increase the true positive rate (the probability of malicious node detection) comes at the cost of increasing the false positive rate.

4.2 Generic detection protocol

In general, on identifying a peer node as suspicious, a node will replace it, that is choose a new neighbor in Vivaldi or a new reference point in NPS.

The only exception to this rule is when the node was embedding against the peer node for the very first time. In this particular case, the node uses its local error e_l[3] as an indicator of the confidence it has in its own coordinate, to carry out a second hypothesis test identical to that presented in the previous section, but this time with a confidence level of $e_l\alpha$. If the test is accepted, then the peer node gets a reprieve and is not replaced, so that a second embedding against this peer node will be attempted at a later time.

The main idea behind this potential reprieve for first-time peer nodes is that a node whose coordinate has already converged towards its true value can afford a few aborted embedding steps with very little impact on the accuracy of its coordinate. On the other hand, a new peer node which is in the process of joining the network may trigger the abortion of an embedding step, simply because its coordinate has not converged yet (as opposed to because it displays a malicious behavior). In this case, the reprieve simply gives time to the new (joining) peer node to converge before being identified as malicious. Of course an embedding node which is not confident in its coordinate must strive to reduce the number of aborted embedding steps so as not to compromise its convergence in the system, and will therefore grant fewer reprieves (because its e_l is higher) than a node that has already converged.

Finally, we use a simple mechanism for the selection of the Surveyor from which a node obtains its calibrated Kalman filter. All Surveyor nodes register with an infrastructure server (e.g. the membership server in NPS can act as Surveyor registrar, while in Vivaldi such server must either be introduced or at least integrated inside an existing bootstrap infrastructure). On joining the coordinate system, a node interrogates this server to obtain the identity of several (randomly chosen) Surveyors. The node then measures its distance to these Surveyors and selects the closest one as representative. From there on, the node fully complies to the embedding protocol rules, except that it will use our detection method to accept or reject embedding steps.

However, when the node has rejected half of its current peer nodes during a same embedding round, it will seek to acquire a new filter as the high rejection rate may indicate that the filter parameters in use may have become stale (i.e. the filter needs “recalibrating”). The node then gets from its current Surveyor (or, as a fallback, any other Surveyors it knows, or the infrastructure server) the list of all the Surveyors it knows. After acquiring the current coordinates of these Surveyors, the node selects the closest one (in term of estimated distance) and obtains its Kalman filter parameters. Note that in the experiments we have carried out, which are described below, we observed very few “recalibrations”, so this very simple Surveyor selection mechanism was appropriate. However, more sophisticated approaches can be considered if need be.

5. Evaluation

We evaluate the effectiveness of the simple anomalous/malicious behavior detection method in securing both Vivaldi and NPS. For each of these embedding protocols, we chose

[3] e_l is the exponential moving weighted average of the measured relative errors of all previously completed embedding steps.

the most potent attack described in [2] and experimented with various populations of malicious nodes within the experimental set-up described in section 3. On PlanetLab, all these experiments were run concurrently so as to experience the same network conditions. In line with the results of section 3.3, the population of Surveyors was set to 8% of the overall population. Surveyors and malicious nodes were chosen at random.

5.1 Performance metrics

To characterize the performance of our detection test, we use the classical false/true positives/negatives indicators. Specifically, a *negative* is a normal embedding step which should therefore be accepted by the test and completed. A *positive* is a malicious embedding step (i.e. where either, or both, the distance estimation and distance measurement between the node and its peer node have been tampered with) which should therefore be rejected by the test and aborted. The number of negatives (resp. positives) in the population comprising all the embedding steps is $\mathcal{P}_N$ (resp. $\mathcal{P}_P$).

A *false negative* is a malicious embedding step that has been wrongly classified by the test as negative, and has therefore been wrongly completed. A *false positive* is a normal embedding step that has been wrongly rejected by the test and therefore wrongly aborted. *True positives (resp. true negatives)* are positives (resp. negatives) that have been correctly reported by the test and therefore have been rightly aborted (resp. completed). The number of false negatives (resp. false positives, true negatives and true positives) reported by the test is $\mathcal{T}_{FN}$ (resp. $\mathcal{T}_{FP}$, $\mathcal{T}_{TN}$ and $\mathcal{T}_{TP}$).

We use the notion of *false negative rate* (FNR) which is the proportion of all the malicious embedding steps that have been wrongly reported as normal by the test, and FNR = $\mathcal{T}_{FN}/\mathcal{P}_P$. The *false positive rate* (FPR) is the proportion of all the normal embedding steps that have been wrongly reported as positive by the test, so FPR = $\mathcal{T}_{FP}/\mathcal{P}_N$. Similarly, the *true positive rate* (TPR) is the proportion of malicious embedding steps that have been rightly reported as malicious by the test, and we have TPR = $\mathcal{T}_{TP}/\mathcal{P}_P$.

The *true positive test fraction* (TPTF) is the proportion of positive tests that correctly identified malicious embedding steps (TPTF = $\mathcal{T}_{TP}/(\mathcal{T}_{TP}+\mathcal{T}_{FP})$).

5.2 Securing Vivaldi

We experimented with our detection scheme on a Vivaldi system subjected to a colluding isolation attack as described in [2]. In this scenario, malicious nodes are trying to isolate a target node, by repulsing all other nodes away from it. The malicious nodes agree on a large "exclusion" zone around the target node and randomly set their own coordinates outside this zone to try and attract honest nodes out of the exclusion zone. Note that an attacker always uses the same coordinate when lying to a given honest node.

Detection Method Performance To evaluate the efficiency of the test, we first plot in figure 9, ROC (Receiver Operation Characteristics) curves observed for different significance levels (α) and several intensities of attacks. These plots show, for each significance level[4], the point corresponding to the false positive rate along the x-axis and to the true positive rate along the y-axis, with one curve per malicious group size (line x=y is plotted as a reference). Obviously, the closer to the upper left corner of the graph a curve is, the better, since such

[4] Significance level values α always increase as a ROC curve is "followed" from the origin. In our experiments, we used values of 1%, 3%, 5% and 10% for α.

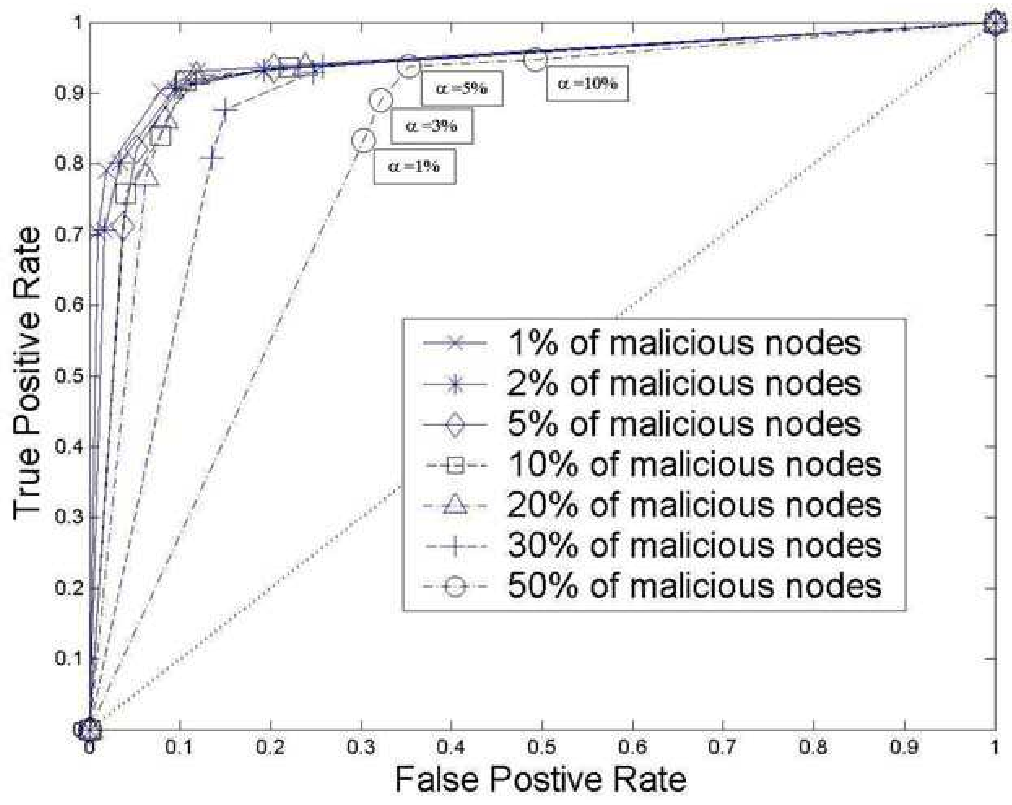


Fig. 9. ROC curves. Each tick on the plots corresponds to a different value of the test's significance level (α).

points correspond to high true positive rates (i.e. a high proportion of positives being reported as such by the test) for low false positive rates (i.e. a small proportion of negatives incorrectly reported as positives). We observe that from this perspective, the detection method can be considered to be excellent for 20% of malicious nodes or less, and still performs well even under heavy attack of up to about 30% of malicious nodes, while the power of the detection method naturally decreases as the malicious population becomes more significant. Another interesting properties of ROC curves is that they show the optimal range for the significance level. Indeed, as the slope of the ROC curve flattens, the increase in true positive rate is proportionally smaller than the corresponding increase in false positives. In other words, a higher significance level, although it always increases the true positive rate of the test, is not always productive as it eventually does more bad than good through increased false positive rates (i.e. the proportion of normal embedding steps that are aborted increases). This means that the significance level of the test should be set to a value that yields a point in the "elbow" of the ROC curve. Based on figure 9, we can deduce that a significance level of 5% seems to be a good compromise.

Figure 10 shows the true positive test fraction of the detection method for various test significance levels under various intensity of attacks. We see that the proportion of positive tests that are true positives is constantly high, regardless of the significance level chosen, for moderate to quite significant proportions of malicious nodes in the population (up to 20% of malicious nodes). However, thereafter the proportion of correct positive tests starts to decrease, although the rate of decrease is inversely proportional to the significance level used. This is because a higher significance level produces more positive tests, catching most malicious embedding steps, and so many more false positives are needed to make up a significant proportion of these. In light of this, a significance level of 5% offers a good compromise.

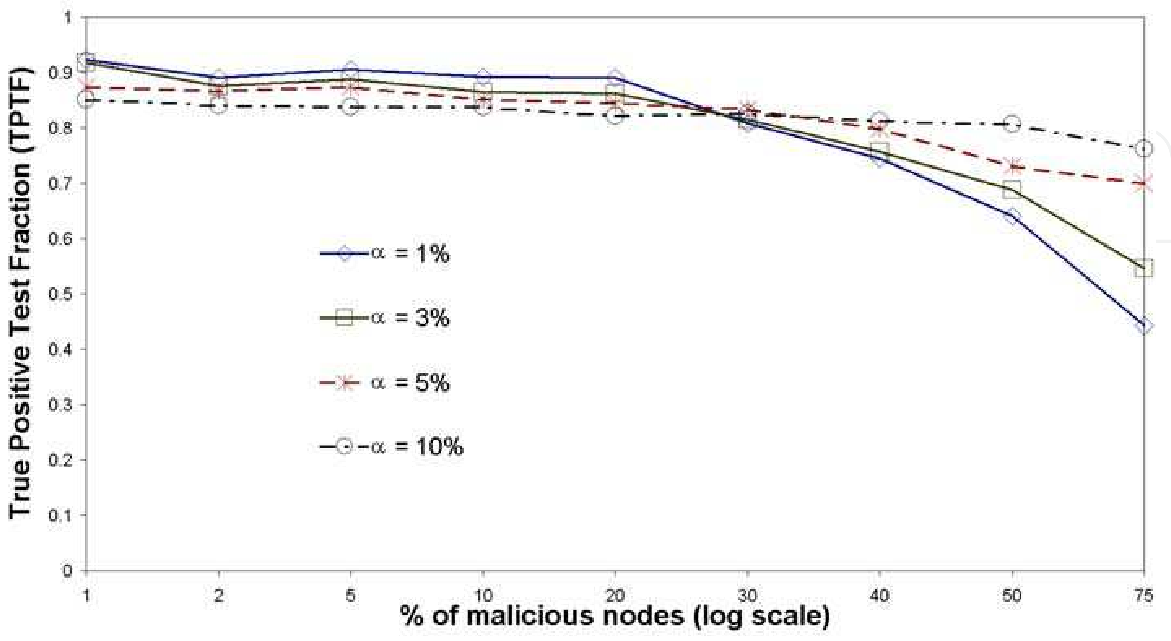


Fig. 10. True positive test fraction.

Figures 11 and 12 show the false positive and negative rates respectively. As expected, a higher significance level results in a more aggressive test that incorrectly classifies a larger portion of normal embedding steps (figure 11) as malicious, while a more lenient test (lower significance level) wrongly reports a higher proportion of malicious embedding steps as normal (figure 12).

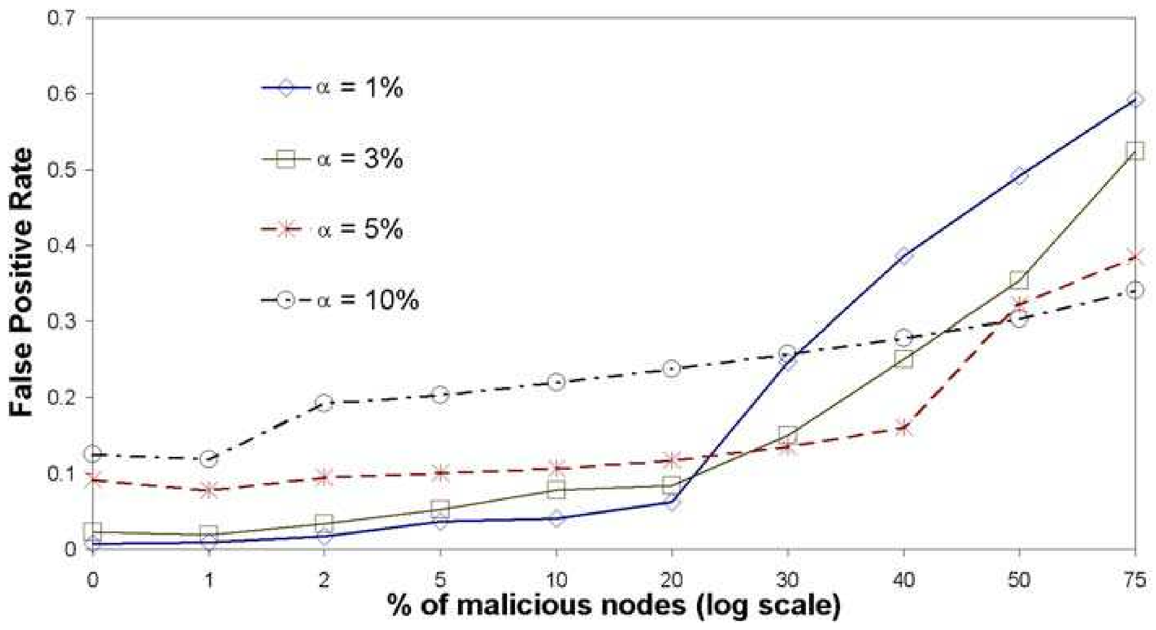


Fig. 11. False Positive Rate.

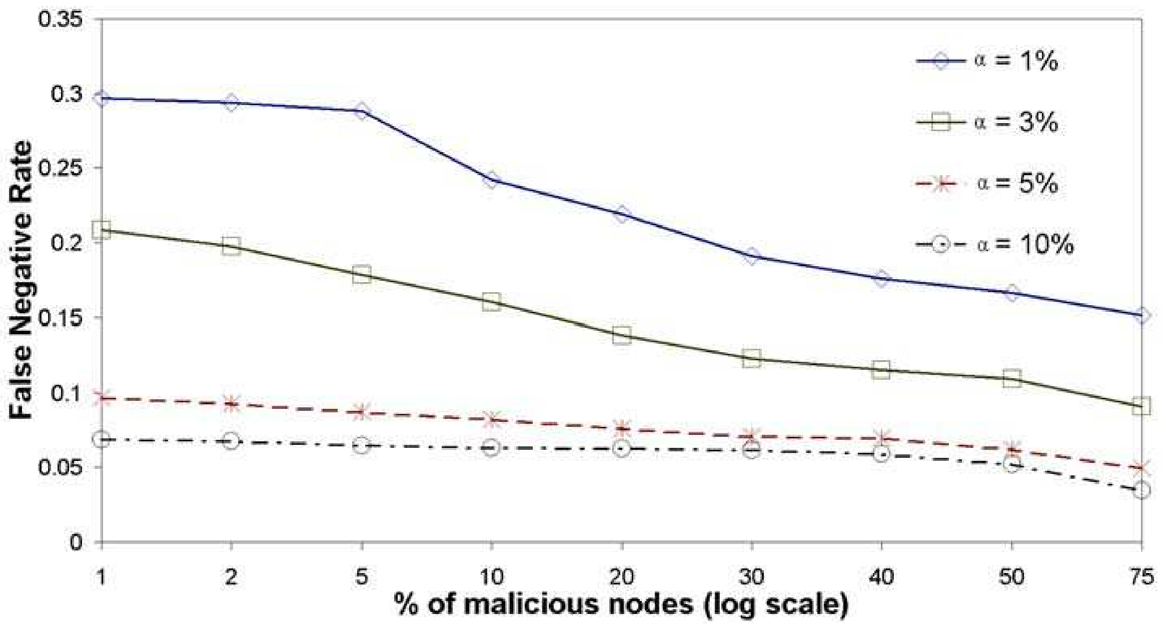


Fig. 12. False negative rate.

Incorrect test results do have a negative impact on the embedding system: false positives artificially reduce the size of available normal nodes that can be used for normal embedding; false negatives give malicious nodes opportunities to corrupt and distort the coordinate space, which can propagate through the system and result in a greater proportion of normal nodes being identified as malicious (false positives) because of mis-positioning. This is exemplified in figure 11, where the false positive rate increases faster, as the population of malicious nodes increases, for lower values of the significance level of the test. Also, despite the fact that the false negative rate curves (figure 12) clearly exhibit negative slopes, one should note that these rates decrease much slower than the increase in malicious population. That is to say that as the number of malicious nodes in the system increases, the number of false negatives does increase, and more damage is incurred in the coordinate space. Although the accuracy of coordinate systems increases with the number of participating nodes, false negatives can therefore have a greater impact on the system than false positives and should therefore be thwarted in priority. As the false negative rates exhibited by tests with significance levels of 5% and 10% are roughly similar, while the more aggressive test yields proportionally a higher false positive rate, the significance level of 5% is a good compromise.

Embedding System Performance From section 5.2, it should be clear that a significance level of 5% gives the overall best test performance. We therefore set the significance level to this value and assess the resistance of a Vivaldi system under various intensity of attacks.

The cumulative distribution function of the measured relative errors, across all normal nodes, after convergence (in the sense of error convergence as defined in section 2) is shown in figure 13. We see that the detection mechanism renders the system practically immune to the attack, when the proportion of malicious nodes is 30%, or less, of the overall node population. Although the system does indeed show degraded performance for higher intensities of malicious attacks, the steeper slope of the CDF with detection, compared to the corresponding curve without (e.g. curves for 50% of malicious nodes), shows that the detection mechanism is not completely overwhelmed and still offers good protection by significantly reducing the impact of the attack.

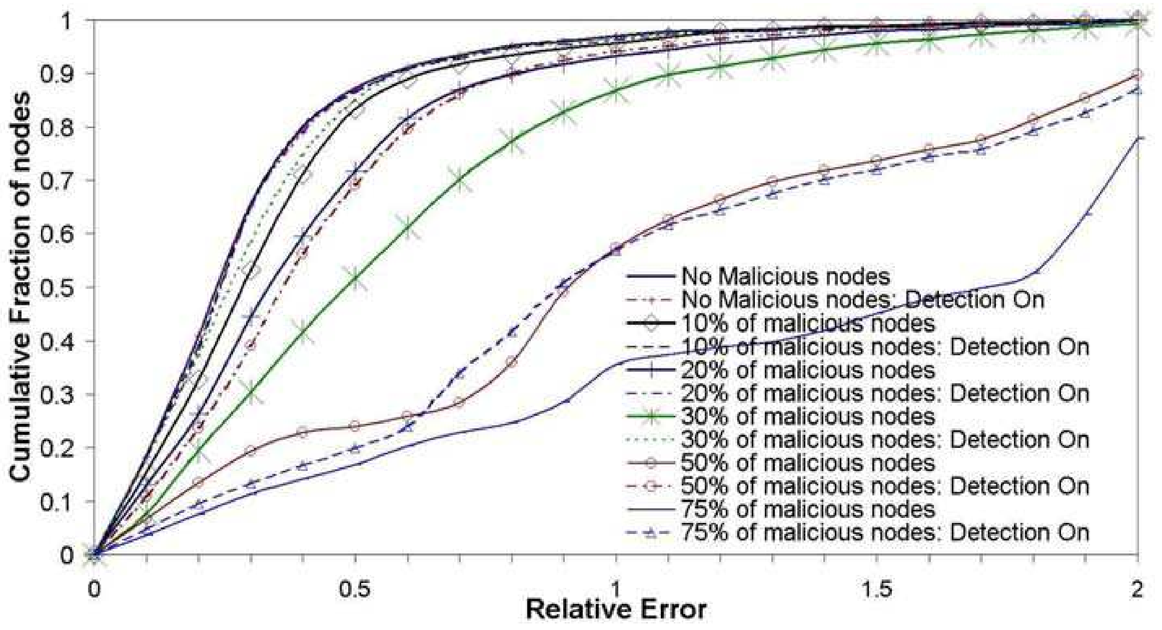


Fig. 13. Distribution of measured relative errors.

5.3 Securing NPS

To test our proposed detection method in the context of the NPS coordinate system, we chose to study the effects of colluding isolation attack as described in [2]. The malicious nodes cooperate with each other and behave in a correct and honest way until enough of them become reference points at each layer. As soon as a minimum number of malicious reference points has been reached (in our experiments this number is set to 5) in a layer, these attackers identify a common set of victims (50% of the normal nodes they know from the layer directly below). When involved in the positioning of their victims, the malicious nodes agree to pretend they are all clustered into a remote (far away) part of the coordinate space and try and push the victims into a remote location at the ”opposite” of where the attackers pretend to be, in order to isolate the victims from the other nodes in the coordinate space. In order to evade detection, including the basic detection method proposed in NPS and which is *always* turned on in our experiments, the malicious nodes use the sophisticated anti-detection method proposed in [2] during their attacks.

Detection Method Performance Figure 14 shows the ROC curves for the detection test in NPS. These curves show characteristics similar to those observed in the Vivaldi system (see section 5.2), albeit slightly better. In particular, these curves show that the detection method withstands heavier attacks better in NPS than in Vivaldi.

There are several reasons for this. First, the basic detection method in NPS works in concert with our own, providing greater opportunities to identify malicious behavior. Also, by its very nature, the embedding method in NPS is less prone to mis-positioning error propagation amongst normal nodes, as nodes in the lower layer do not take part in the embedding of other nodes. And finally, by design, the attack considered in this section makes fewer victims than that studied in section 5.2 (i.e. 50% of normal nodes as victims vs 100% in Vivaldi).

The same observation is also true for the false positive and false negative rates (not shown) with again, overall, a significance level of 5% seemingly offering the best compromise between “catching” malicious embedding steps while not being overly cautious and over-reacting to normal variations in network conditions.

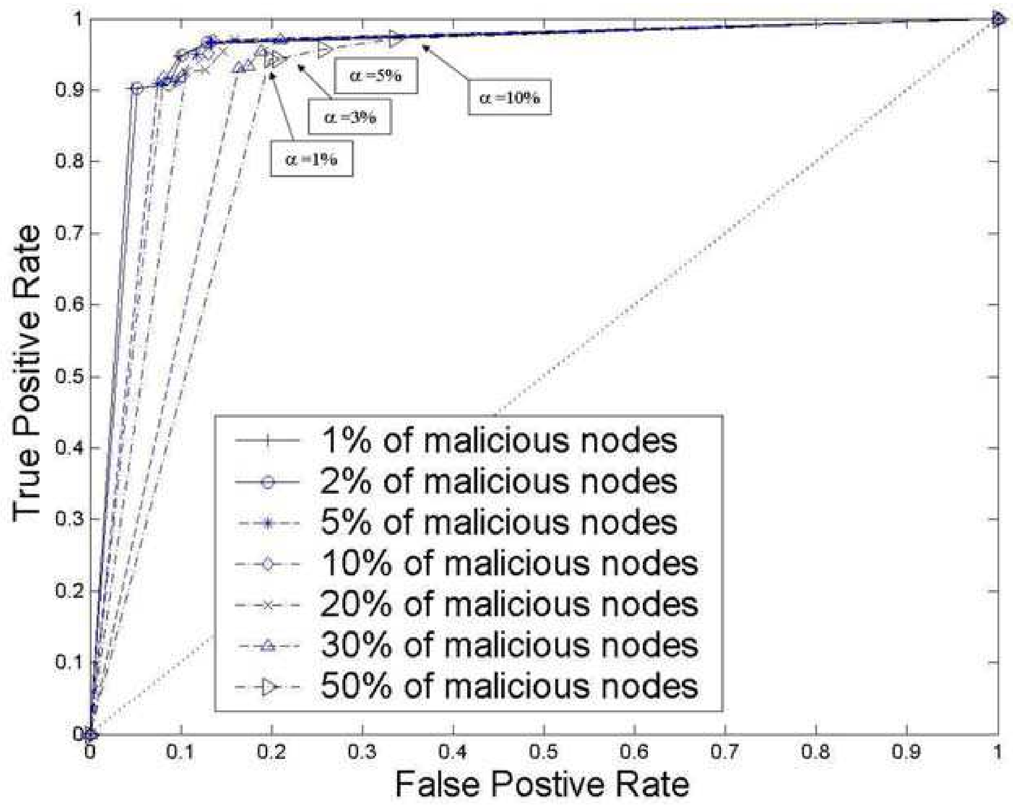


Fig. 14. ROC curves.

The similarities between the test performance under NPS and Vivaldi, despite the different nature of the attacks under consideration and even differences in coordinate "structure" (two-dimensional with height for Vivaldi versus eight-dimensional for NPS), illustrates the generality of the proposed detection method. This is because our detection test is based on the modeling of a dimension-less quantity (the relative error) which is at the very core of *any* coordinates embedding system.

Embedding System Performance We study the performance of the NPS embedding system when subject to increasing intensity of attacks, while being protected by our detection scheme. Note that in this section, "detection off" really means that our proposed detection mechanism is not used, but the basic NPS detection mechanism is still "on".

Figure 15 shows the cumulative distribution function of relative errors in the system. We note again similarities with the dynamic behavior of similar Vivaldi systems, except that the tail of the CDF for 50% malicious nodes with detection is heavier than the corresponding curve in the Vivaldi case. Keeping in mind that in NPS not all nodes are victims and that not all normal nodes will propagate mis-positioning errors, this indicates that the attack is still quite effective against its victims, albeit "dampened" by the detection mechanism. This effect is compounded by the fact that, with our simple detection protocol, malicious nodes that have found their way into the layer hierarchy of NPS and act as Reference Points, do stay in place throughout the experiment, despite numerous detections of their corrupt embedding steps. Nevertheless, the detection method proposed affords near immunity to the system up to rather severe attack conditions (e.g. about 30% of malicious nodes in the system).

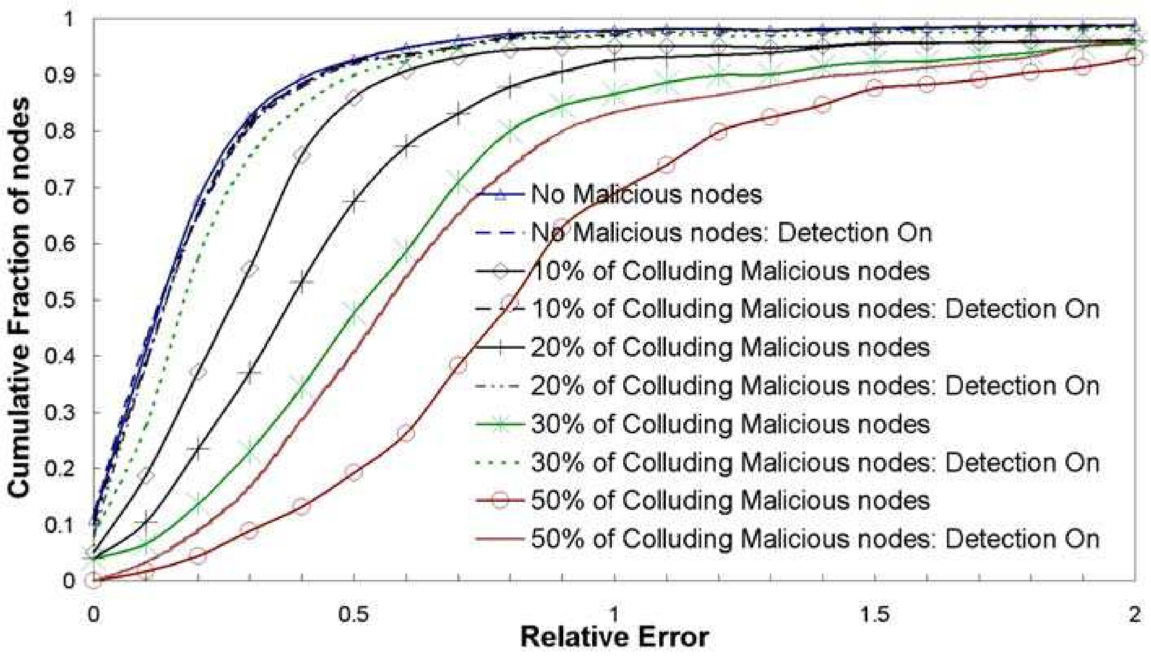


Fig. 15. Distribution of measured relative errors.

6. Conclusions

We have presented a method for malicious behavior detection to secure the embedding phase of Internet coordinate systems. Our method does not rely on the geometric properties of the coordinate space, and is therefore unaffected by potential triangular inequality violations which often occur in the Internet [11, 12]. Instead, our detection test is based on the modeling of the dynamic relative errors observed in a clean system. The relative error is a dimension-less quantity which is at the very core of *any* embedding method, leading us to believe that our proposed detection test can effectively identify malicious behavior in very many embedding protocols and coordinate space structures that are under a potential very large range of attacks. The experiments presented in this chapter do show that the performance of the detection test is effectively the same in two different scenarios involving different embedding protocols and different attacks. As far as we know, this is the first such general detection test, capable of surviving sophisticated attacks. Also, we consider exclusively attacks aimed at distorting the coordinate space, carried out by nodes inside the embedding system. Our method thus succeeds where more obvious methods based on authentication would fail.

7. References

[1] M. A. Kaafar, L.Mathy, T. Turletti andW. Dabbous, *Real attacks on virtual networks: Vivaldi out of tune*, In Proceedings of the SIGCOMM workshop on Large Scale Attack Defense LSAD 2006,p129-146, PISA, September 2006.
[2] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous, *Virtual Networks under Attack: Disrupting Internet Coordinate Systems*, In Proceedings of CoNext 2006, Lisboa, December, 2006.

[3] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti and W. Dabbous, *Securing Internet Coordinate Embedding Systems*, In proceedings of ACMSIGCOMM, Kyoto, Japan, August 2007.

[4] R.E. Kalman, and R.S. Bucy, *New Results in Linear Filtering and Prediction Theory*, In Transactions of the ASME - Journal of Basic Engineering Vol. 83: pp. 95-107, 1961.

[5] T. E. Ng, and H. Zhang, *Predicting internet network distance with coordinates-based approaches*, In Proceedings of the IEEE INFOCOM, New York, June 2002.

[6] M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris, *Lighthouses for Scalable Distributed Location*, In Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS), Berkeley, February 2003.

[7] M. Costa, M. Castro, A. Rowstron, and P. Key, *Practical Internet coordinates for distance estimation*, In Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), Tokyo, March 2004.

[8] T. E. Ng and H. Zhang, *A Network Positioning System for the Internet*, In Proceedings of the USENIX annual technical conference, Boston, June 2004.

[9] F. Dabek, R. Cox, F. Kaashoek and R. Morris, *Vivaldi: A decentralized network coordinate system*, In Proceedings of the ACM SIGCOMM, Portland, Oregon, August 2004.

[10] J. Ledlie, P. Gardner, and M. Seltzer, *Network Coordinates in the Wild*, In Proceedings of NSDI, Cambridge, April 2007. Available as Harvard University Computer Science Technical Report TR-20-06, October 2006.

[11] E. K. Lua, T. griffin, M. Pias, H. Zheng, and J. Crowcroft, *On the accuracy of Embeddings for Internet Coordinate Systems*, In Proceedings of InternetMeasurement Conference (IMC), Berkeley, October 2005.

[12] H. Zheng, E. K. Lua,M. Pias, and T. Griffin, *Internet Routing Policies and Round-Trip Times*, In Proceedings of the Passive Active Measurement (PAM), Boston, March 2005.

[13] K. P. Gummadi, S. Saroiu, and S. D. Gribble, *King: Estimating Latency between Arbitrary Internet End Hosts*, In Proceedings of SIGCOMMInternetMesasurementWorkshop (IMW), Pittsburgh November 2002.

[14] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, *On the Constancy of Internet Path Properties*, In Proceedings of ACMSIGCOMMInternetMeasurementWorkshop, San Francisco, November 2001.

[15] Z. Ghahramani, G. Hinton, *Parameter Estimation for Linear Dynamical Systems*, University of Toronto, Technical Report CRG-TR-96-2.

[16] S. M. Kay, Fundamentals of Statistical Signal Processing: Estimation Theory, Prentice Hall, Ch. 7, 1993.

[17] A. Dempster, N. Laird, and D. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, Series B, 39(1):1-38, 1977.

[18] H. Lilliefors, *On the Kolmogorov-Smirnov test for normality with mean and variance unknown*, Journal of the American Statistical Association, Vol. 62. pp. 399-402, June, 1967.

[19] A. Soule, K. Salamatian, and N. Taft, *Combining Filtering and Statistical Methods for Anomaly Detection*, In Proceedings of Internet Measurement Conference (IMC), Berkeley, October, 2005.

[20] J. Bilmes, *A gentle tutorial on the EM algorithm including gaussian mixtures and baumwelch*, Technical Report TR-97-021, International Computer Science Institute, Berkeley, CA, 1997.

Kalman Filter
Edited by Vedran Kordic

ISBN 978-953-307-094-0
Hard cover, 390 pages
Publisher InTech
Published online 01, May, 2010
Published in print edition May, 2010