Veracity: A Fully Decentralized Service for Securing Network Coordinate Systems

Micah Sherr   Boon Thau Loo   Matt Blaze
University of Pennsylvania

Abstract

Decentralized logical coordinate systems have been proposed as a means of estimating network distances. These systems have widespread usage in p2p networks, ranging from neighbor selection to replica placement. Unfortunately, these systems are vulnerable to even a small number of malicious nodes lying about their coordinates or measurements. In this paper, we introduce *Veracity*, a *fully* decentralized service for securing network coordinate systems. Unlike prior proposals, Veracity requires neither the presence of a large number of *a priori* trusted nodes nor the use of network triangle inequality testing. Veracity utilizes a vote-based approach, where all advertised coordinates are independently verified by a minimal set of nodes before being used. Via detailed simulations in p2psim, we demonstrate that Veracity mitigates a variety of known attacks against Vivaldi for moderate sizes of malicious nodes, incurring acceptable communication overhead, and in some cases, even reducing the convergence time of the coordinate system.

1 Introduction

Decentralized logical coordinate systems such as Vivaldi [4], PIC [3] and NPS [12] have been proposed as a means for estimating network distances. These systems share the same basic functionality: a node measures the latency between itself and a subset of other nodes to estimate a logical position, usually a point in an n-dimensional hyperplane or hypersphere. These systems have widespread usage in p2p networks, ranging from proximity-based neighbor selection [5] to replica placement [13]. Embedded coordinate systems such as Vivaldi are particularly amenable to a p2p environment, due to their low maintenance overhead and ability to handle network churn. The lack of explicit remote measurements also makes such systems useful as a measurement substrate for performance-aware anonymized networks [15].

Unfortunately, a major impediment to the widespread deployment of these systems is their vulnerability to attack. In particular, Vivaldi has been shown to be particularly susceptible to various forms of attack. To illustrate, recent studies [9] have shown that when 30% of nodes lie about their coordinates, Vivaldi's accuracy decreases by a factor of 5, essentially rendering the system unusable. When attackers collude, even 5% malicious nodes have a sizable impact on system performance.

To address these vulnerabilities, there have been recent proposals that are targeted at maintaining accurate coordinates in the presence of malicious nodes. PIC [3] disregards coordinates that fail the RTT triangle inequality, an unrealistic assumption given that such violations could potentially be common and persistent [11]. Saucez's [14] reputation-based approach and Kaafar's [8] proposal require the presence of a large number of *a priori* trusted nodes (e.g. at least 8% [8]) in order for the system to be adequately protected. This is an impractical requirement given the targeted scale of most embedded coordinate systems (hundreds of thousands). For networks of smaller scale, direct measurement of RTT is preferred without requiring a decentralized coordinate system.

In light of the above limitations, we explore a new point in the design space: *Veracity*, a *fully* decentralized service for securing network coordinate systems without requiring either preselected trusted nodes or the triangle inequality test. At a high-level, Veracity utilizes a vote-based approach, where all advertised coordinates have to be independently verified by a minimal random set of nodes before they can be used. An adversary who attempts to disrupt the network by publishing inconsistent coordinates or by publishing coordinates that do not correspond to its effective latency will not pass Veracity's checks, and consequently its coordinates are ignored by honest nodes.

Although our vote-based approach applies generally to most decentralized coordinate systems, we focus on Vivaldi since it has received the most study recently from the p2p community. Via detailed simulations in p2psim [6], we demonstrate that Veracity mitigates a variety of known attacks against Vivaldi [9] for moderate sizes of malicious nodes, incurring acceptable communication overhead, and in some cases, even reducing the convergence time of the coordinate system.

2 Background on Vivaldi

In this section, we present a brief introduction to the Vivaldi system in order to provide the background necessary for presenting Veracity.

Vivaldi uses a fully distributed spring relaxation algorithm, requiring no fixed network infrastructure and no distinguished nodes. The system envisions a spring between each pair of nodes, with the resting position of the spring equaling the network latency between the pair. At any point in time, the distance between the nodes in the coordinate space determines the current length of the spring connecting the nodes.

Nodes adjust their coordinates after collecting latency information from a set of neighbors. The squared error function, $E = (RTT_{ij} - ||x_i - x_j||)^2$ (where RTT_{ij} is the roundtrip time between the two nodes and $||x_i - x_j||$ is the distance between their coordinates), reflects the potential energy of the spring connecting the two nodes. Vivaldi attempts to minimize the potential energies over all springs. In each timestep of the algorithm, nodes allow themselves to be pulled or pushed by a connected spring. The system converges when the squared error function (i.e., the potential energies) is minimized.

3 Attacker Model

Prior work [9] has identified three types of attacks against coordinate systems: *disorder*, *isolation* and *repulsion*. In disorder attacks, malicious nodes publish false coordinates and/or delay responses to measurements in order to cause instability and inaccuracy in the coordinate system. Isolation and repulsion attacks are aimed at isolating or repulsing targeted victim nodes. We focus on evaluating Veracity against disorder attacks. However, since Veracity's general approach defends against malicious nodes that falsify their coordinates or induce/report artificially inflated latencies, we believe that the techniques outlined in this paper can mitigate all three attacks.

Given that malicious nodes can collude, we adopt the constrained-collusion Byzantine model proposed by Castro *et al.* [2] on securing distributed hash tables (DHTs). Under the constrained-collusion model, malicious nodes can insert, delete, or delay messages. We assume that attackers control some fraction ($f < 1$) of the network and that there are independent coalitions of size cN, where N is the number of nodes in the network and $1/N \leq c \leq f$.

4 Veracity Service

In this section, we describe the operations of Veracity based on modifications to Vivaldi. While we focus on Vivaldi for ease of exposition, the techniques generally apply to any coordinate system with an update model that involves each node adjusting its coordinates based on measured RTTs and reported coordinates of other nodes. For example, Veracity can be used to verify the choice of landmark nodes in PIC and NPS.

The Veracity service runs on every node in the coordinate system, publishing node updates to members of a deterministically assigned *verification set* (VSet). The publisher's coordinates must be verified by members of his VSet before it may be used by other nodes.

The verification process works as follows: Each VSet member measures the RTT between itself and the publisher, and determines the accuracy of the coordinates by computing the difference between the measured RTT and the estimated RTT obtained from the logical coordinates. A node that is interested in a publisher's coordinates first ensures that the majority of nodes within the publisher's VSet indicate that the error is below a pre-defined error threshold.

Veracity makes no distinction between false coordinates specified by an attacker and inaccurate coordinates computed by the coordinate system. In either case, members of the VSet will dissuade interested nodes from using the coordinates. As a result, Veracity can potentially improve the accuracy of the underlying coordinate system even when no malicious nodes exist.

In the remainder of this section, we will detail the operations of the Veracity service.

4.1 VSet Construction

When a Veracity node joins the network, it is assigned a *Veracity IDentifier (VID)*. The VID is computed by applying a collision-resistant cryptographic hash function (e.g., SHA-1) to the node's IP address. The use of a cryptographic hash function over IP addresses reduces the likelihood that malicious nodes in a coalition can join each other's VSets.

Given a node with identifier vid, the members of the VSet are determined using the recurrence

$$h_i = \begin{cases} \text{SHA-1}(h_{i-1}) & \text{if } i > 1 \\ \text{SHA-1}(vid) & \text{if } i = 1 \end{cases}$$

where i ranges from 1 to the *VSet size*, Γ. A larger Γ increases the trustworthiness of coordinates (since more nodes are required in the verification process), at the expense of an additional communication cost and convergence time. At a minimum, Γ should be one greater than the dimensionality of the Euclidean space.

Each Veracity node uses a distributed lookup service to select nodes for its VSet. Veracity is agnostic with respect to the particular lookup service. Our implementation utilizes DHTs [1] since they provide support for scalable lookups. Specifically, Veracity nodes participate in a DHT overlay, and given a candidate VSet identifier h_i, we use the $p_i \leftarrow lookup(h_i)$ API to scalably retrieve the IP address p_i of the VSet member whose VID is closest to h_i.

Since the performance of Veracity depends on the reliability of the underlying lookup service, it is important to ensure that the service is resilient to malicious nodes. While there are known attacks against DHTs [2], most of these have well-known orthogonal solutions. For example, the *routing failure test* [2] is used to detect attacks against DHT routing. When an attack is detected, *redundant routing* [2] can be employed to deliver the lookup correctly using multiple diverse routes.

The routing failure test uses a heuristic based on the density of node identifiers. As a result, the test may result in false positives and false negatives. Interestingly, the use of the VSet size (Γ) naturally mitigates the inaccuracies of this test. Given a false negative (FN) probability and their desired effective VSet size Γ', Veracity nodes can set $\Gamma \leftarrow \Gamma'(1 + FN)$ to compensate for the expected number of updates that will undetectably not reach the members of their VSets.

4.2 Update Dissemination to VSet

When a node (for clarity, the *"publisher"*) updates its coordinates, it transmits the new coordinates to its VSet members so that other nodes may verify and use it. The VSet members are retrieved on demand at each update: given its VSet identifiers $h_1, h_2, ..., h_\Gamma$, the node utilizes the distributed lookup service to locate all its VSet members $p_1, p_2, ..., p_\Gamma$.

On each coordinate update, the node transmits an *update tuple* (V, τ, C, ip) to all VSet members. This tuple contains information on the updated coordinates, as well as additional information required for the verification process. V is the node's VID, τ is a logical timestamp incremented whenever the node updates its coordinates, C is the new coordinates, and ip is the node's IP address.

Upon receiving the update tuple, each VSet member $p_i \in \{p_1, p_2, ..., p_\Gamma\}$ measures the RTT between itself and ip, and computes the *error ratio* $\delta_{(p_i,V)} = \Big|RTT(p_i, ip) - ||C - C_{p_i}||\Big| / RTT(p_i, ip)$, where C_{p_i} is the set of coordinates of p_i and $||C - C_{p_i}||$ denotes the distance between the two nodes' coordinates. Finally, p_i locally stores the *evidence tuple* $(V, \tau, C, ip, \delta_{(p_i,V)})$.[1]

4.3 VSet Verification of Updates

To verify a coordinate, a node (for clarity, the *"investigator"*) first contacts the coordinate's publisher and obtains the publisher's *claim tuple* (V, Γ, τ, C, ip). In the case of Vivaldi, the investigator is a node who wishes to verify a neighbor's (i.e., the publisher's) coordinates before using the coordinates to update its own. The investigator immediately discards the publisher's coordinates if the publisher's IP address is not ip, $V \neq$ SHA-1(ip), or it deems Γ insufficiently large to offer enough supporting evidence for the coordinates.

Otherwise, the investigator contacts the members of the publisher's VSet (constructed, as before, on demand by taking recursive hashes of V) to ensure the coordinates satisfactorily reflects measured RTTs. The investigator transmits the query (V, τ) to each member $p_i \in \{p_1, p_2, ..., p_\Gamma\}$ of the publisher's VSet. If p_i stores an evidence tuple containing both V and τ, it returns that tuple to the investigator. The investigator then checks that the VID, IP address, and coordinates in the publisher's claim tuple matches those in the evidence tuple. If there is a discrepancy, the evidence tuple is ignored.

After querying all members of the publisher's VSet, the investigator counts the number of non-discarded evidence tuples for which $\delta_{(p_i,V)} \leq \hat{\delta}$, where $\hat{\delta}$ is the investigator's chosen *ratio cutoff parameter*. Intuitively, this parameter gauges the investigator's tolerance of coordinate errors: a large $\hat{\delta}$ permits fast convergence times when all nodes are honest, but risks increased likelihood of accepting false coordinates.

If the count of passing evidence tuples meets or exceeds the investigator's *evidence cutoff parameter*, R, the coordinate is considered verified. Otherwise, the publisher's coordinate is discarded. The underlying coordinate system (e.g., Vivaldi) can treat discarded coordinates in the same manner as message loss. That is, the system does not update its coordinates and waits until the subsequent sampling period to pick a new neighbor.

Note that the additional storage and communication costs incurred by Veracity are minimal. Assuming all nodes use the same veracity assertion parameter (Γ), each node is expected to store Γ tuples. For a network of size N, the communication overhead imposed by the verification process is $O(\Gamma \log N)$, since Γ lookups are required.

4.4 Other Practical Issues

Having presented Veracity's verification process, we outline two additional issues essential for the practical realization of Veracity.

Bootstrapping: When a node joins a coordinate system, its initial coordinate estimations will likely be inaccurate. Hence, our VSet verification process may consistently reject coordinates from this new node even though it is honest. When many nodes initially join the coordinate system simultaneously, a verification scheme that aggressively rejects coordinates will significantly delay the convergence time. To address this issue, we impose an initial startup period where the cutoff ratio $\hat{\delta}$ is set to infinity. This essentially results in the verification process ignor-

[1]Nodes may periodically purge inactive tuples (i.e., those that have not recently been queried) to reduce storage costs.

ing the accuracy of measurements during the startup period. Updates are accepted as long as all VSet members see a consistent view of the updates with the same logical timestamps. While this approach risks the likelihood of accepting false coordinates during the startup process, these false coordinates will be eliminated once the verification process begins to impose the cutoff ratio. The length of the startup period depends on the stabilization rate of the underlying coordinate system, and is outside the scope of this paper.

Network churn: Under network churn, VSet members may receive different updated coordinates due to routing inconsistencies. The VSet membership can also change during the verification process. If a node requesting coordinate verification derives an inaccurate VSet due to network churn, some of the polled peers will not respond (either because they are offline or they did not receive the relevant updates). To compensate, Γ can be chosen to allow the expected number of responses (taking into consideration churn, or more generally, message loss) to exceed the evidence cutoff parameter.

5 Evaluation

In this section, we describe the performance of Veracity based on detailed p2psim simulations. As input latencies to our simulation, we make use of the "King" data set [10], a collection of pairwise Internet latencies between 1740 DNS servers determined using the King method [7].

We use p2psim's Vivaldi implementation as the underlying coordinate system, executed at timesteps of 0.005. Each Vivaldi node maintained a neighbor set of 64 randomly chosen peers, and all coordinates lay on a two-dimensional Euclidean space. All nodes joined the network within the first five "ticks" (a tick corresponds to 10 simulated seconds) and persisted throughout the simulation (10,800 ticks).

Veracity is implemented as an extension to the Vivaldi protocol. To handle the initial bootstrap process described in Section 4.4, all nodes execute Vivaldi during the first 600 ticks, and then switch to use Veracity's verification process for the simulation's duration. All nodes set $\Gamma = 6$, $\hat{\delta} = 0.3$, and used an evidence cutoff parameter of 4.

Malicious nodes begin their attack after 1200 ticks[2] We simulate the attacker model as described in Section 3. Malicious nodes lie about their coordinates by choosing a random point with a radial distance from the origin chosen uniformly at random from $[0, 1000]$ and an angular coordinate uniformly at random from $[0, 2\pi)$. Malicious nodes delay their responses to measurements so that the measured RTTs are chosen uniformly at random from $[R, 1000]$, where R is the actual RTT between the malicious node and the node conducting the measurement. To maximize the disorder caused by the attack, all malicious nodes report an error of 0.01, hence relaying high confidence in their faulty coordinates to other Vivaldi nodes.

In addition, malicious nodes can also participate in the verification process. When asked to verify a peer's coordinate, malicious VSet nodes never respond and therefore offer no evidence towards meeting the evidence cutoff ratio.

5.1 Evaluation Metrics

To measure the accuracy of Vivaldi with and without using Veracity, we make use of the following three metrics:

Median Error in RTT: Each honest node n_i calculates the median of the differences $\big|RTT(n_i, n_j) - ||C_{n_i} - C_{n_j}||\big|$ between itself and all nodes n_j, $n_j \neq n_i$. The median error in RTT is the median over all such medians.

Normalized Error Ratio: Each honest node computes the median of the error ratios (computed as described in Section 4.2) between itself and all other nodes. Letting the *system error ratio* denote the median over all such medians, a coordinate system's normalized error ratio is the system error ratio divided by the system error ratio of Vivaldi when no attack takes place. For example, if Vivaldi experiences a normalized error ratio of three when some of its nodes are malicious, then it experiences three times the median error ratio as when all nodes behave honestly. This metric is essentially a normalization of the error ratio relative to Vivaldi's accuracy *when no attack takes place*.

Convergence time: The earliest time at which the median error ratio is within 3% of the final error ratio (i.e., the ratio obtained at the end of the simulation) for 10 consecutive ticks. Comparing against the ratio at the end of the simulation ensures the convergence time does not capture transient "plateaus" during the simulation. If any of the error ratios for the last 10 ticks of the simulation differ from the final error ratio by more than 3%, we say that the coordinate system does not converge.

5.2 Absence of Malicious Nodes

We first show that in the absence of malicious nodes, Veracity does not lessen the performance of the underlying coordinate system. As shown in Figure 1, Veracity and Vivaldi yield similar accuracy when no attackers are present. At the end of the simulation, Veracity's normal-

[2]This corresponds to Kaafar et al.'s notion of an "injection disorder attack" [9].

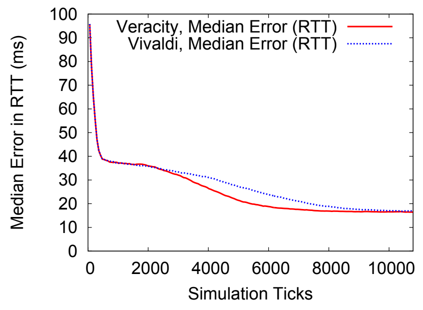

Figure 1: Median error RTTs of Vivaldi and Veracity in the absence of an attack.

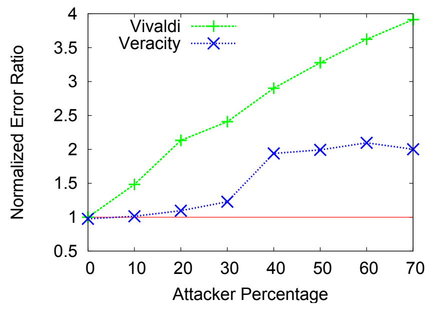

Figure 2: Normalized error ratios of Vivaldi and Veracity with inconsistent coordinates.

ized error ratio was 0.98, indicating a modest improvement in accuracy over Vivaldi. Additionally, Veracity reduced Vivaldi's convergence time by 12%. We conjecture that the increased performance is due to Veracity's ability to de-emphasize erroneous coordinates (as may occur in the case of triangle inequality violations).

5.3 Uncoordinated Attacks

We now explore Veracity's effectiveness when some participating nodes are malicious. In this section, we assume that the malicious nodes do not cooperate, and consider coalitions of attackers in the following section.

In a naïve attack, malicious nodes report nonidentical coordinates to members of their VSets and artificially inflate the RTTs of network queries. Figure 2 shows the efficacy of this strategy against Veracity and Vivaldi. Since honest nodes discard any evidence tuples from the VSet that do not match the coordinate being verified, the inconsistent coordinates do not pass Veracity's coordinate verification process. Hence, under this type of attack, Veracity offers a significant improvement in accuracy over Vivaldi. Even when 70% of the nodes act maliciously, Veracity yields a normalized error ratio of just 2.0. Moreover, by reducing the influence of nodes that produce random RTT measurements, Veracity experienced a median convergence time (computed over all attacker percentages) 20% less than that of Vivaldi.

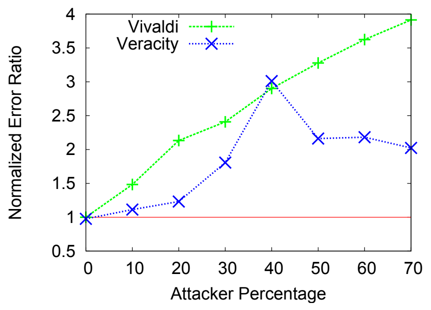

Figure 3: Normalized error ratios of Vivaldi and Veracity under uncoordinated attacks.

Of course, Veracity-aware nodes can choose not to report inconsistent coordinates to members of its VSet. Instead, nodes can report a static but randomly chosen coordinate and inflate RTTs as before. Figure 3 shows the effectiveness of Veracity at mitigating such an attack. For values of f (the fraction of malicious nodes) between 0 and 0.3, Veracity significantly improves upon the accuracy of the underlying coordinate system. For example, when 20% of the nodes act maliciously, Veracity lowers Vivaldi's normalized error ratio by 42%. The median convergence times taken over the eight tested attacker percentages differed by only 1%.

Vivaldi slightly outperforms Veracity when $f = 0.4$. When malicious nodes comprise a significant fraction of the network, fewer coordinates (both honest or dishonest) are verified since dishonest nodes never respond to coordinate verification requests. The effect of verifying malicious nodes' "borderline" coordinates (i.e., those that yield error ratios just below $\hat{\delta}$) will therefore be more catastrophic to Veracity's overall accuracy. As the fraction of malicious nodes increases (but remains sufficiently small to allow some coordinates to be verified), so does the availability of such borderline coordinates.

Interestingly, Veracity's accuracy improves when $f > 0.4$. With such a high percentage of malicious nodes, very few coordinates will pass the coordinate verification process. Consequently, most nodes will hover around the origin (their initial coordinate), resulting in a lesser normalized error ratio (since the RTTs advertised by malicious nodes exceeds that of true Internet RTTs).

5.4 Coalition Attacks

Under the constrained-collusion Byzantine model, malicious nodes within the same coalition coordinate their attack. We adapt our previous attack by making the follow-

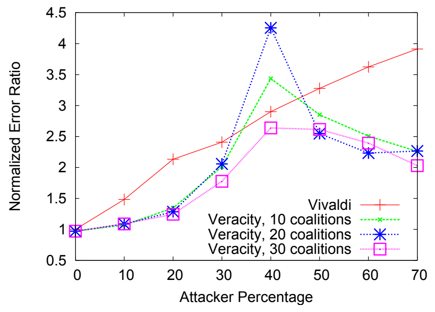

Figure 4: Normalized error ratios of Vivaldi and Veracity under coalition attacks.

ing modification: if a node is malicious and it is asked to verify a coordinate from a malicious node in its coalition, then it returns an error ratio that is less than $\hat{\delta}$ (i.e., supporting the coordinate); in all other cases, the malicious node does not reply.

Figure 4 shows the performance of Veracity when malicious nodes are organized into coalitions. Unsurprisingly, Veracity performs better with smaller coalitions. When Γ and f are fixed, the expected number of malicious nodes in a VSet that belong to the same coalition decreases as the number of such coalitions increases. At the extreme, each malicious node acts as its own coalition, resulting in the uncoordinated attack described above.

Comparing Figures 3 and 4, we learn that the the coordinated attack strategy does not significantly reduce the effectiveness of Veracity if the number of coalitions is at least 10 and $f < 0.4$. The expected number of malicious nodes in a VSet belonging to the same coalition is insufficient to mount a successful attack. For instance, when malicious nodes are partitioned into 10 coalitions, Veracity reduces Vivaldi's normalized error ratio by 37% when attackers control 20% of the network (compared with 42% when attackers do not collude) and 16% (compared with 25%) when malicious nodes comprise 30%.

6 Conclusions and Future Work

In this paper, we propose *Veracity*, a fully decentralized service for securing logical coordinate systems. Unlike prior approaches, Veracity requires neither the use of *a priori* trusted nodes nor network triangle inequality testing. Our preliminary simulation results are promising: Veracity is effective at mitigating various disorder attacks aimed at the underlying coordinate system, and in some instances, results in a decrease in convergence time. Even against coalitions of cooperating opponents, Veracity significantly reduces the effects of attacks. For instance, Veracity yields a 43% reduction in the normalized error ratio when 20% of the network is malicious and acts independently, and a 37% reduction when malicious peers are organized into 10 coalitions of cooperating nodes.

Our most immediate future work entails the deployment of Veracity on PlanetLab together with various coordinate systems (Vivaldi, PIC, NPS, etc.) to better study the performance of the system under actual network conditions (e.g., churn, routing changes, etc.). Additionally, the availability of a fully distributed and secure coordinate system has interesting implications to our ongoing work [15] on anonymous routing.

References

[1] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking Up Data in P2P Systems. *Communications of the ACM, Vol. 46, No. 2*, Feb. 2003.

[2] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. Wallach. Secure Routing for Structured Peer-to-peer Overlay Networks. In *OSDI 2002*, 2002.

[3] M. Costa, M. Castro, R. Rowstron, and P. Key. PIC: Practical internet coordinates for distance estimation. In *International Conference on Distributed Computing Systems*, 2004.

[4] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. *SIGCOMM*, 34(4):15–26, 2004.

[5] F. Dabek, J. Li, E. Sit, F. Kaashoek, R. Morris, and C. Blake. Designing a DHT for low latency and high throughput. In *USENIX Symposium on Networked Systems Design and Implementation*, 2004.

[6] T. M. Gil, F. Kaashoek, J. Li, R. Morris, and J. Stribling. p2psim: a simulator for peer-to-peer protocols. `http://pdos.csail.mit.edu/p2psim/`.

[7] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: estimating latency between arbitrary internet end hosts. In *ACM SIGCOMM Workshop on Internet Measurment (IMW)*, 2002.

[8] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti, and W. Dabbous. Securing internet coordinate embedding systems. In *ACM SIGCOMM*, Aug 2007.

[9] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous. Real attacks on virtual networks: Vivaldi out of tune. In *LSAD '06: Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense*, pages 139–146, 2006.

[10] "king" data set. `http://pdos.csail.mit.edu/p2psim/kingdata/`.

[11] E. K. Lua, T. G. Griffin, M. Pias, H. Zheng, and J. Crowcroft. On the accuracy of embeddings for internet coordinate systems. In *Internet Measurment Conference*, 2005.

[12] T. S. E. Ng and H. Zhang. A network positioning system for the internet. In *Proceedings of the 2004 USENIX Annual Technical Conference*, Jun 2004.

[13] P. Pietzuch, J. Ledlie, M. Mitzenmacher, , and M. Seltzer. Network-aware overlays with network coordinates. In *1st Workshop on Dynamic Distributed Systems*, 2006.

[14] D. Saucez, B. Donnet, and O. Bonaventure. A reputation-based approach for securing vivaldi embedding system. In *Dependable and Adaptable Networks and Services*, 2007.

[15] M. Sherr, B. T. Loo, and M. Blaze. Towards application-aware anonymous routing. In *Second USENIX Workshop on Hot Topics in Security (HotSec)*, August 2007.