MIT-CSAIL-TR-2009-015 April 16, 2009

Computing Network Coordinates in the Presence of Byzantine Faults

You Zhou

Computing Network Coordinates in the Presence of Byzantine Faults

by

You Zhou

S.B., Electrical Engineering and Computer Science (2007); S.B., Mathematics (2007)
Massachusetts Institute of Technology

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2008

Author . . . .
Department of Electrical Engineering and Computer Science
May 23, 2008

Certified by . . . .
Barbara Liskov
Ford Professor of Engineering
Thesis Supervisor

Accepted by . . . .
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Computing Network Coordinates in the Presence of Byzantine Faults

by

You Zhou

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2008, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Network coordinate systems allow for efficient construction of large-scale distributed systems on the Internet. Coordinates provide locality information in a compact way, without requiring each node to contact every potential neighbor; distances between two nodes' coordinates represent estimates of the network latency between them.

Past work on network coordinates has assumed that all nodes in the system behave correctly. The techniques in these systems do not behave well when nodes are Byzantine. These Byzantine failures, wherein a faulty node can behave arbitrarily, can make the coordinate-based distance estimates meaningless. For example, a Byzantine node can delay responding to some other node, thus distorting that node's computation of its own location.

We present a network coordinate system based on landmarks, reference nodes that are used for measurements, some of which may be Byzantine faulty. It scales linearly in the number of clients computing their coordinates and does not require excessive network traffic to allow clients to do so. Our results show that our system is able to compute accurate coordinates even when some landmarks are exhibiting Byzantine faults.

Thesis Supervisor: Barbara Liskov
Title: Ford Professor of Engineering

Acknowledgments

Thanks foremost to my advisor, Barbara Liskov, for her support, feedback, and well-considered optimism, without which I never would have written this all in time. Her sense of direction and guidance on this thesis were invaluable.

Thanks to James Cowling for instigating this project, for innumerable helpful discussions, and for providing code for implementations of some of the related work. This thesis could not be possible without him.

Thanks to my academic advisor, Madhu Sudan, for providing bountiful advice and genuinely caring; to David Karger for pointing out an error in an earlier version of this work; to Dan Ports for sending papers our way; and to Waseem Daher, Jeff Arnold, the other SIPB thesers, and David Dryjanski for staying afloat together.

Thanks to all the PMG labmates, my guides down the forking paths of research, who helped me learn the ways of systems and made the kidney a welcoming place. And all the other researchers publishing in this area of late unknowingly provided some relevance-motivation.

MIT Medical helped prevent an eleventh-hour injury from spelling midnight for finishing this thesis on time. It gave me yet another perspective on Eri Izawa's nice "thesis comes at the end" sentiment.

It's no use to write but lose one's sight. Thanks to my friends for their encouragement, especially to Yi-Hsin Lin for keeping my life in tune, and to my parents for their perspicacious perspective.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nodes in a system distributed across the Internet have no inherent measure of their proximity to each other, but by communicating with nearby nodes, they may be able to reduce communication costs. Assuming bandwidth does not represent a bottleneck, nodes can actively ping other nodes to determine the latencies or round-trip times to other nodes; however, doing so for thousands of nodes can be costly and may outweigh any derived benefit. Network coordinates are a compact representation of this proximity information that can avoid such costs.

Network coordinates are an embedding of nodes into a metric space, with the distance between nodes in the space representing the round-trip time between them. Typically, a node will make some small number of measurements to other nodes in order to determine its own coordinates. When all the nodes in the system behave correctly, the resulting coordinates can predict network latencies accurately [14, 31].

To be useful, coordinates for a node must have a low *error*, defined as the difference between the actual and predicted round-trip times. However, things can go badly if nodes in the system are trying to compromise its effectiveness. Attackers can try to worsen the error for a correctly behaving node by distorting measurements, lying about their own coordinates, or causing network delays. For example, in Vivaldi [6], a decentralized coordinate system, even with only 10% of nodes in the system maliciously colluding to disrupt the coordinates, the median average error can more than double [13].

1.2 Our contributions

The main problem our design addresses is that of faulty *landmarks*, the nodes to which clients make measurements to compute their own coordinates. We present a new system that allows correctly behaving clients to compute accurate coordinates even if some of their measurements are faulty—the measured round-trip times may be inaccurate or the landmarks may never respond.

Our system protects against a powerful Byzantine adversary that can control a fraction of the nodes in the system and coordinate them to behave arbitrarily. Additionally, our coordinate system is practical: coordinates are compact, they can be efficiently computed by a node from its measurements to a set of landmarks, they are easily translated into predictions for network latencies, and very little bandwidth overhead is used for measurement.

Results from our simulations show that even with measurements from Byzantine faulty landmarks, our coordinate system provides clients with coordinates that are nearly as accurate for predicting network latencies as they would have been had there not been faults.

1.3 Thesis outline

The remainder of the thesis is organized as follows:

Chapter 2 outlines the basic problems this thesis addresses and the design decisions we made.

Chapter 3 provides definitions and groundwork for our system.

Chapter 4 explains the protocol our system uses to compute coordinates.

Chapter 5 describes our implementation, a simulation based on actual network measurements, and our results.

Chapter 6 surveys prior research in network coordinate systems, including previous approaches to mitigating error.

Finally, Chapter 7 concludes with our observations and directions for future work.

Chapter 2

Approach

2.1 Design considerations

2.1.1 Abstract model

The abstract model of a network coordinate system consists of nodes in the network and the pairwise round-trip times, or distances, between them. It ignores heuristic information, such as IP prefixes, that might provide hints about which nodes are proximate to each other. Synthetic coordinates, embedding the nodes into a coordinate space, are then computed from some measurements of the round-trip times, so that the distance between nodes is a prediction of the corresponding round-trip latency; coordinates are chosen to minimize some measure of error. The state of the network at any given time reflects the round-trip times measured at that time. Most previous work in the area of network coordinates [6, 32, 25, 21, 22, 29, 38, 34, 30] follows this model.

In network coordinate systems, a *client* node may join and need to compute its coordinates, or it may need to maintain them to be accurate over time. In the abstract model, to carry out this task, a client uses *measurements* of the round-trip times to some subset of the nodes in the system, referred to as its *neighbors*, and computes (adjustments to) its own coordinates relative to the coordinates of its neighbors. If every node uses the same set of neighbors, this set is called the *landmarks* and the system is said to be a *landmark system*. Otherwise, it is considered a *decentralized system*.

Some systems, such as Vivaldi [6], also use auxiliary information about the estimated error in coordinates, so that accurate coordinates influence inaccurate ones more than vice versa.

The coordinates, measurements, and other information must be accurate in order for nodes to accurately compute coordinates and gain useful information about the proximity of other nodes but are easily distorted by an adversary trying to make the system less useful. We designed our system to protect against as powerful an adversary as possible, for maximal generality. For the reasons described in the next section, we use a landmark-based approach rather than a decentralized approach to computing network coordinates.

2.1.2 Byzantine adversary

The adversarial model we assume is based on *Byzantine faults*. A node experiencing a Byzantine fault can behave arbitrarily [4]—it may stop responding, send spurious messages, delay messages, and so on. A Byzantine faulty node can actively try to compromise the correctness of the system.

We assume a general, powerful Byzantine adversary that can coordinate all of the faulty machines in the network to do its bidding. Additionally, the Byzantine adversary may introduce network delays for a bounded period of time and manipulate the routing of packets to distort distance both positively and negatively.

2.1.3 Faults

How can a Byzantine adversary attack correctly behaving nodes in the system? It can do so principally by forging information given to other nodes or manipulating the measurements other nodes make to malicious nodes.

If a neighbor is faulty, it cannot be relied upon to respond with its own coordinates or auxiliary information correctly. It is easy for a malicious node to lie about its own

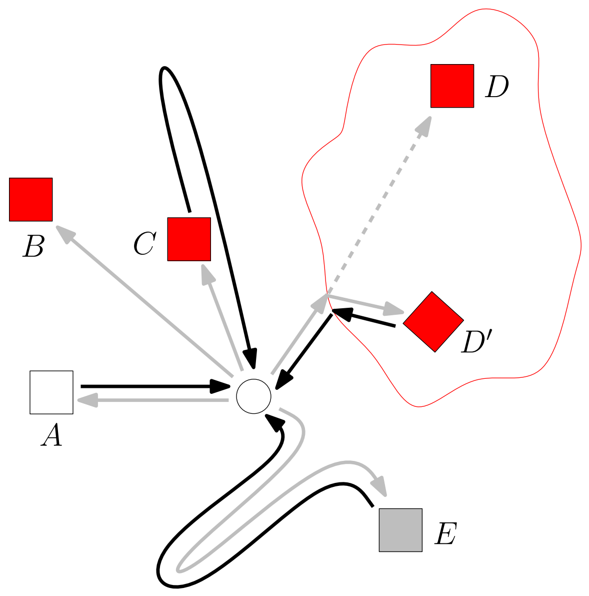

Figure 2-1: The central node makes measurements to nodes A through E. A behaves correctly. B does not respond. C delays messages, increasing the measured RTT. D has messages rerouted to D', *decreasing* the measured RTT. E responds correctly but is affected by network delay.

state in response to requests that ask for it. And unfortunately, a protocol to verify a neighbor's coordinates using the neighbor's neighbors can be prohibitively expensive, especially if that neighbor is also updating its own coordinates and cannot be expected to have the same coordinates from measurement to measurement.

Furthermore, a neighbor will always be able to distort measurements made to it. Consider the measured round-trip time to be the elapsed time between a node sending a message to its neighbor and its receiving a response. Figure 2-1 illustrates the possible adversarial behavior:

(B) The measured times can go to infinity if message responses are dropped, as responses never arrive.

(C) The measured times can increase arbitrarily if message responses are intentionally delayed.

(D) The measured times can increase or even *decrease* if messages are intercepted and rerouted to a closer colluding machine. This last, counter-intuitive scenario is further explained below.

Puppeteer

Consider a malicious node operated by an adversary that also controls routing for some subnetwork containing it. This node can share its credentials with other machines in this subnetwork, allowing them to "impersonate" it. The border routers of this subnetwork can then redirect traffic to the closest of these machines. This adversary, which we call a *puppeteer*, can thus act to *decrease* distances measured to this malicious node, even when the node's coordinates are accurate and it does not misrepresent its own coordinates. With sufficiently many puppets and enough control over how packets are routed, the distance may be decreased arbitrarily.

To our knowledge, this adversary has not been previously examined in the literature. For example, Kaafar et al. assumed an attacker that only increases measured times [13], and PIC assumed an adversary could decrease distances only down to the distance to the nearest malicious node [5].

Network delays

We assume the attacker is able to introduce persistent network delays on some links between a node and some neighbors (such as E in Figure 2-1), even if the neighbors are behaving correctly. Measurements to these neighbors thus will be inconsistent with the original network round-trip times. However, we believe it is reasonable to expect network damage to be repaired eventually, so that delays cannot persist forever. We address the problem of network delays in an extension to our work by adapting over time in response to new measurements, as described in 4.4.1.

2.1.4 Landmarks

It is clear that a Byzantine adversary able to misrepresent both the coordinates of and measurements to nodes it controls is much more dangerous than one that can only manipulate one of these pieces of data. For this reason, we chose our system to be a *landmark* system, in which client measurements are made only to a fixed set of landmark nodes with published, stable coordinates: landmarks cannot misrepresent their own coordinates and do not update them.

Any distributed system can be set up to use landmarks. Since landmarks are only in infrequent communication with all the clients in the system, they require only a small amount of additional infrastructure. Ratnasamy et al. [26] observe that a single landmark may even be replicated across multiple machines in one data center. Geometric requirements on the positions of the landmarks are examined in section 3.2.3.

2.2 Goals

Our system was designed to achieve certain properties, some of which are largely defined by the role of a network coordinate system. These properties, as we formulate them, are:

Accuracy: Our system should provide coordinates that predict network round-trip times well.

Fault tolerance: Our system should provide reasonably accurate coordinates to honest clients even if some landmarks are Byzantine faulty or network delays cause inaccurate measurements. Our coordinates should be accurate even when all faulty nodes in the system can collude.

Practicality: Clients must be able to compute their own coordinates in a reasonable amount of time. Landmarks should not have intensive computational demands placed on them.

Efficiency: Since the purpose of network coordinates is to reduce the amount of communications overhead necessary to determine the proximity of other nodes in the network, our system should not impose excessive communication burdens.

Scalability: For the measurement overhead to grow linearly with the number of clients in the system, each client should communicate with only a constant number of nodes—the landmarks.

Certifiability: Our coordinates can be made self-certifying, so that other nodes can verify a client's coordinates.

Chapter 3

Requirements

In this chapter, we present the basic assumptions and groundwork for our network coordinates system.

3.1 Definitions

We call *clients* the nodes that join the system and compute their own coordinates from measurements made to the *landmarks*.

The system parameter f, known to all nodes in the system, specifies how many faulty landmarks are tolerated. In addition to the L landmarks that are required to compute accurate coordinates even if none of them are faulty, to tolerate f faults we include an additional $2f$ landmarks. We describe these parameters in more detail below.

3.1.1 Base number of landmarks

Suppose that no landmarks are faulty. While in theory, in a d-dimensional space, only $d+1$ landmarks are required to accurately triangulate coordinates, due to the inherent embedding distortion, more are usually needed for a reasonable level of accuracy. We call L the base number of requisite landmarks.

Past work has studied the number of landmarks without settling on a definitive number for L. Ratnasamy et al. [26] found that 8 to 12 landmarks are sufficient in a low-dimensional space. Dabek et al. [6] found that the accuracy of GNP [21], a landmark-based coordinate system, did not significantly increase beyond 16 landmarks, although the authors of GNP used 19 in their experiments. Tang and Crovella [32] used 11 and 12 landmarks for some other datasets, according to what was available.

3.1.2 Metric space

The authors of Vivaldi [6] studied the effects of choosing different metric spaces for the coordinate embedding. One kind of space, a 2-dimensional Euclidean space together with a *height vector* representing the distance to the network core that most paths to a node would have to traverse, was found to be more effective than either a Euclidean space in higher dimensions or a spherical space. Later work by Ledlie et al. [15] confirmed that 2-D spaces with height vector were more effective than higher-dimensional Euclidean spaces for representing the topology of the Internet on several data sets.

Following Vivaldi's results, we chose to use a 2-D Euclidean space with height vectors for our implementation. A point in the space is given as $\mathbf{x} = (x, y, h)$, with the third coordinate representing the height ($h \geq 0$). The precise definition of the distance metric is as follows:

$$d\big((x_1, y_1, h_1), (x_2, y_2, h_2)\big) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + h_1 + h_2$$

It is clear that this definition satisfies the metric space axioms of symmetry and the triangle inequality [27]. Although the distance from a node to itself $d(\mathbf{x}, \mathbf{x})$ is not 0 in general, this detail has no effect in practice because a node never needs to measure the round-trip time to itself.

3.1.3 Byzantine fault tolerance

Since there is no way to determine when a landmark is Byzantine faulty, as it may behave arbitrarily, when we compute coordinates, we have no a priori way to decide which measurements are the results of faults and should be excluded from the computation. Our approach will be to have the non-faulty landmarks outweigh the faulty ones.

Intuitively, because in a d-dimensional space, d measurements are insufficient to locate a single point, the faulty landmarks may be able to agree in a region of the coordinate space where several honest landmarks also agree in hopes of causing a client to choose bad coordinates there. So f additional honest landmarks may be needed to agree on the correct coordinate, and we suppose that L are necessary for accurate coordinates in any case. Thus, we choose $2f+L$ for the number of landmarks used to tolerate f faults. While our choice of $2f+L$ landmarks appears to suffice, in theory, it may also be possible to have fewer landmarks beyond the f.

3.2 Landmark coordinates

3.2.1 Accuracy

The coordinates in our system are bootstrapped by the landmarks' published coordinates. We assume that the landmarks have accurate, unchanging coordinates. Coordinates that never change are a reasonable simplified model, since we do not expect network topologies to change frequently.

However, if topologies do change over long periods of time, fixed coordinates, such as those of the landmarks, will not remain accurate, and neither will the client coordinates that were computed with them as reference points. However, this problem is orthogonal to that of computing accurate client coordinates for honest clients, the focus of our work, so we do not consider it here.

3.2.2 Publication

A client joining the system must have some mechanism to discover the landmarks' published coordinates. Our system uses the idea of a *directory* service from Tor [8]. We assume that the landmarks publish their IP addresses and coordinates in the directory, and that it is easily accessible to clients. The directory contains a certificate authority's public key, which the client can use to verify the landmarks' public keys in order to prevent a man-in-the-middle attack. We assume the directory is not faulty and always has available the list of coordinates and IP addresses of all the landmarks, signed by the certificate authority. (See section 3.3 for the cryptographic assumptions we require.)

The directory provides a trusted view of the landmarks in the system to clients. It prevents landmarks that later become faulty from reneging on their originally chosen, correct coordinates.

A directory can represent a single point of failure and can become a bottleneck. The former problem can be solved using replication—for example, by using PBFT [4]. The latter is not problematic in our system because we have only very limited bandwidth demands on the directory. Each client needs only to retrieve the landmarks' coordinates and one public key, which even for hundreds of landmarks should not take up more than a few kilobytes. Alternative approaches to discovering the network, such as asking landmarks for a list of the other landmarks in the system, can offload some of the work from the directory, but the directory stills needs to provide the certificate authority's public key.

3.2.3 Distribution

The accuracy of a landmark-based coordinate system depends on the relative position of the landmarks in the coordinate space. Two landmarks that are too close to one another may not provide as much discriminating information as they would if they had greatly differing paths across the Internet to clients. Tang and Crovella [33] showed that well-chosen landmarks can significantly reduce the number of landmarks needed for the same level of error. Some other work considers [5, 16, 39] how landmark selection differently affects the accuracy of predicting short and long distances.

We assume that our landmarks can be chosen to be distributed across data centers that are separated geographically and are pre-selected out-of-band by the system operators to be well-distributed.

3.3 Cryptography

Our system requires a basic level of authenticity. A landmark signs the measurement messages it sends, and the public keys of the landmarks are all signed by a certificate authority whose public key is available from the directory service. All signatures are assumed to be existentially unforgeable [10].

We also include nonces in all measurement messages to prevent replay attacks or spoofed replies from being able to affect round-trip times. Some systems [5] assume that nonces are sufficient to guarantee the authenticity of message responses from landmarks, but we note the possibility of intermediate routers under the Byzantine attacker's control that could read the nonce.

Our scheme uses public key cryptography, because the communications are so infrequent that the cost of public key cryptography is not too great. To set up a shared key using Diffie-Hellman key exchange [7] would require more round-trips and additional messages, and the landmarks would have an additional resource burden of at least temporarily storing a secret key for every client.

We assume our cryptographic primitives are unbreakable, treating them as a black box. We do not specify any particular cryptographic scheme to be used, so long as it meets our requirements.

Chapter 4

Protocol for Computing Coordinates

This section describes the protocol for a client to compute its coordinates from scratch. Our system begins with established landmark coordinates. Clients joining the system compute their coordinates from measurements to the landmarks. However, because some landmarks are faulty, we would like to prevent measurements to them from distorting our computed coordinates; thus, we attempt to exclude responses from faulty landmarks.

Each client can be considered in isolation, since its communications are only with the landmarks. There are two phases in this protocol: First, the client makes measurements to all of the landmarks. Then, once it has these measurements, it runs a computation to find its coordinates, eventually discarding some of the measurements.

To decouple the measurements from the computation step, we introduce an abstraction, the *estimate*, representing a client's view over time of the round-trip latency to a landmark. The computation step runs atop the estimate abstraction. Since we assume that network latencies return to normal eventually, in the average case we will have an estimate for all landmarks.

The system we implemented is a simplification that does not use estimates gleaned from many measurements over time, but instead a single measurement is used as the estimate. This simplification has the same properties except that it does not handle network delays, as introduced in section 2.1.3; instead, it assumes that nodes affected by persistent delay are never heard from and thus faulty. It reflects the client's initial state upon joining, when it does not have prior histories for any landmarks and may be unable to compute its coordinates accurately if it is affected by these network faults.

4.1 Initialization

The client first obtains the list of landmarks and the certificate authority's key from the directory service, and it verifies the list's accuracy. Then it begins to measure the round-trip times to the landmarks, using the landmark IP addresses from the directory. The landmark coordinates from the directory are used later, in the computation.

4.2 Measurement Protocol

A client initiates a measurement to a landmark A by sending a *ping* message containing the client's nonce, $nonce_1$, to A. Upon receiving such a message, A responds with a *pong* message containing $nonce_1$. This message is signed by A to prevent spoofing by an adversarial network. When the client receives the *pong*, it verifies that the message came from A and takes the elapsed time since sending the *ping* message as the measured round-trip time. In our simplified system, this measurement is taken directly as the estimate that is used in the computation.

While an honest landmark will reply to the client immediately, faulty landmarks may affect the measurements in a number of ways, as described in section 2.1.3. Though landmarks cannot lie about their coordinates, they can distort measurements both positively and negatively.

Some faulty nodes may fail and never respond to measurements, so we also implement a timeout of approximately 800 milliseconds for each measurement. We assume that any latencies longer than this timeout do not correspond to accurate measurements and discard them.

```
Compute-Coordinates(initial-coords, estimates)
1  $x \leftarrow$ Gradient-Descent(initial-coords, estimates)
2  $\ell \leftarrow |\,estimates\,|$
3  while $\ell > f + L$
4        do estimates.remove(Estimate-with-Worst-Error(estimates))
5           $\ell \leftarrow \ell - 1$
6           $x \leftarrow$ Gradient-Descent($x$, estimates)
7  return $x$
```

Figure 4-1: Pseudocode for the computation. An estimate is removed and the coordinates are recomputed, iteratively, until $f + L$ remain.

4.3 Computation

After obtaining estimates, of which the client will have up to $2f + L$, and having learned the landmarks' published coordinates from the directory, the client computes its own coordinates. Our protocol for computing landmark-based coordinates removes up to f nodes from the computation incrementally.

There are two issues to consider:

1. Given a set of estimates to landmarks, how are coordinates chosen to minimize the error?

2. Which landmarks' estimates should be included in the computation of the client's coordinates?

These aspects of the computation are referred to as the *triangulation* for selecting coordinates and the *strategy* for selecting landmarks to use, respectively. We chose to use a gradient descent for the triangulation step; it is run initially to include all available estimates. Our strategy is to iteratively remove the estimate with the worst error from the triangulation and recompute until only $f + L$ landmark estimates remain. Figure 4-1 shows a pseudocode description of our algorithm.

4.3.1 Error function

Since the coordinate system is used to predict round-trip times between nodes, we define the error in each estimate as the inaccuracy between the coordinate distance, or the predicted round-trip time, and the estimate, or the measured round-trip time. Let p_i be the measured round-trip ping time from the client to landmark i, whose coordinates are $\mathbf{x_i} = (x_i, y_i, h_i)$ in the 2-D with height metric space. For the individual errors, we use the spring potential energy error function as defined in Vivaldi [6], $(d(\mathbf{x}, \mathbf{x_i}) - p_i)^2$.

The total error function is defined to be the average of the errors in the estimate to each landmark. Thus, the average error at a point $\mathbf{x} = (x, y, h)$ is the average of the spring potential energies,

$$\frac{1}{n} \sum_{i \in \text{landmarks}} \left(d(\mathbf{x}, \mathbf{x_i}) - p_i\right)^2,$$

where n is the number of landmarks in the computation. In a 2-dimensional with height metric space, this expression becomes

$$\frac{1}{n} \sum_{i \in \text{landmarks}} \left(\sqrt{(x - x_i)^2 + (y - y_i)^2} + h + h_i - p_i\right)^2.$$

4.3.2 Triangulation

In the triangulation step, a set of estimates to landmarks and a starting point in the coordinate space is given as the input, and the output is a point in the coordinate space that minimizes the error function for those landmarks. In this step, a client finds the coordinates $\mathbf{x} = (x, y, h)$ that minimize the energy function. The starting point can be initially selected randomly or fixed at the origin; for subsequent triangulation steps, the triangulation can begin from the point computed by the previous one.

The problem of computing the coordinates that minimize some error function determined by the estimates falls into the domain of unconstrained nonlinear programming, a well-studied numerical problem in the literature [1, 35]. The nonlinear optimization can be solved using a numerical method such as gradient descent, the Nelder-Meade simplex algorithm (which requires a starting simplex of $n + 1$ points in an n-dimensional space), or simulated annealing [35]. These techniques will find a local minimum, not necessarily a global minimum, of a potential function.

We chose to use a gradient descent to find the coordinates in the triangulation step. For gradient descent, we must find the gradient of the error function; since the energy function corresponds to spring potential energy, by Hooke's law, the "force" should correspond to the spring force.

While a gradient descent may be implemented to travel along the direction of the gradient to a one-dimensional minimum on that line [35], we sacrifice this level of exactness for efficiency and avoid solving this one-dimensional minimization problem. Our step size in the gradient descent is chosen to be proportional to the magnitude of the gradient of the error function. We find that the scaling of $1/n$ from the averaging works well, and our experiments do not show any convergence failures, indicating that our gradient descent is finding a local minimum as desired.

4.3.3 Strategy

Since we would like to compute accurate coordinates, even when up to f landmarks are faulty, we will compute coordinates from the estimates to only $f + L$ landmarks. Thus, we remove landmarks from the triangulation computation until only $f + L$ remain. Up to f landmarks will be removed—fewer than f if some landmarks are never heard from and thus the client has no estimate for them. Equivalently, we can say that exactly f will not be included in the final round of computation, and those landmarks that are never heard from were removed to begin with.

In the worst case, when all $2f + L$ landmarks respond, finding the subset of $f + L$ that gives the absolute lowest error requires examining all $\binom{2f+L}{f} = O\big((2f + L)^f\big)$ possible subsets, an exponentially large number (because L is large, at least 10 or so). Hence, it is impractical for even small values of f, such as 4, because the gradient descent in the triangulation is not a cheap computation. Note that there is also no guarantee that this optimal subset does not include faulty landmarks. (Since we do not evaluate coordinates' accuracy based on faulty landmarks, the optimal coordinates for the client should be computed just with the estimates from the $f+L$ honest landmarks, but since the client has no way of knowing which landmarks are faulty, the exponential-time strategy is the best it can hope to do.)

Therefore, a client cannot practically find the absolute minimum over all possible subsets of $f+L$ landmarks. A client instead tries to find an appropriate subset of landmarks that gives a low average error. To do so, we use intermediate triangulation steps on certain larger subsets.

We use the following approximation. Given the coordinates computed from an triangulation step, consider the average error to each landmark. That landmark that contributes the most error is the one to be removed from the triangulation to decrease the average error by the most, *if the resulting coordinates are the same.* However, the resulting coordinates should be different, since removing a landmark will change the set of landmarks and thus the error function. To justify removing this landmark with the worst error, the approximation we use is that the new coordinates are approximately the old coordinates.

We only remove one landmark at a time from the computation because the discrepancy between new error function and the old error function grows with the number of landmarks' estimates removed from the error function. Our strategy thus takes at most $f+1$ rounds of the triangulation to compute the final coordinates.

Other strategies

We also studied an alternative strategy, using $O\big((L+f)f\big)$ rounds of triangulation. Again we iteratively remove one landmark's estimate from the triangulation at a time, but we select that estimate differently. If we have n estimates left, then for each estimate, we tentatively remove it, leaving $n-1$ on which to run the triangulation, and see what the resulting error is. The one that is actually removed is the one whose removal gave the lowest resulting error.

Although this alternative strategy is provably better when $f=1$, as it does evaluate all possible subsets of size $f+L$ and choose the best one, we found that it does not perform as well for larger values of f. We conjecture that it may be because, given a choice between a region of coordinate space agreed on by mainly honest landmarks and another region in accordance with mainly faulty landmarks, it is easier for the coordinate to wander toward one or the other, and hence to become lost in the faulty space.

Our strategy is thus a more practical approach to computing the coordinates that minimize the average error.

4.4 Extensions

4.4.1 Estimate abstraction

As an extension to our system, to handle network faults, we dispense with the simplification that measurements are identified with estimates. Instead, each client maintains, for each landmark, its current *estimate* of the round-trip time to that landmark. The estimates are undefined before any measurements take place. Estimates are updated by measurements to the landmarks and are used by the coordinate computation.

There are two reasons to define this abstraction. First, there may be changes to the network topology over time, including persistent network delays. Second, jitter, the variance in network round-trip times due to queueing delays at routers, should not unduly affect how coordinates are computed. Thus, the current estimate should be able to adapt over time to reflect a new underlying round-trip time but should also include elements of a low-pass filter.

Our system aggregates measurements over time to form estimates using an exponential weighted moving average according to

$$estimate_{i+1} = (1-\alpha) \cdot estimate_i + \alpha \cdot measurement_i \tag{4.1}$$

for some small fraction α. This method is similar to the predictors used in other systems; for example, TCP's retransmission timer uses an exponential weighted moving average to estimate a link's round-trip time [24].

4.4.2 Certifiability and faulty clients

While our system addresses the problem of preventing faulty landmarks from degrading clients' computed coordinates, it is also worth asking how to prevent clients from choosing arbitrary coordinates. After all, in a locality-aware overlay network, malicious clients inserting themselves into the coordinate space may make the cost of communication more expensive for an application running atop the coordinates by tricking honest nodes, close to the fictitious coordinates in the coordinate space, into routing through them while they are in actuality far away, thereby defeating the original purpose of the coordinates.

Our coordinates can be made *self-certifying* with a small amount of additional communication that allows the landmarks to generate signed estimates for the clients to collect. The landmarks also maintain estimates to the clients. These self-certifying coordinates can be verified independently by any other node in the system.

To make measurements self-certifying, we modify the protocol as follows. Landmarks also include a landmark nonce, $nonce_2$, in their *pong* messages. Once a client has an estimate, it can send a *guess* mesage to the landmark with $nonce_2$ and its current estimate, which is derived from its previous estimate and the new measurement. The landmark verifies that the client's estimate is within some tolerance of its own. If it is, it replies with a *check* message in which it signs the client's estimate (it must use public key cryptography here in order for other nodes to be able to verify the coordinates). The client then collects the signed estimates. The total number of messages is doubled from our original protocol.

If the computation of coordinates from the set of estimates is deterministic, then any node can compute the same set of coordinates from these signed estimates, which suffice to certify the coordinates (though in practice the certificate should consist of both the set of estimates and the derived coordinates). Elements of randomness can be made deterministic by initializing from the same random seed, which can be derived from the client's ID. Thus, verifying a client's self-certifying coordinates can be done by checking the validity of the landmarks' signatures on the estimates and re-running the computation based on those estimates.

For certifiable coordinates, the additional resource burdens on the landmarks are in storage and bandwidth—the current estimate, a single number, is stored for every client, and approximately twice as much network bandwidth is used. Either can serve to limit the number of clients our system can support.

Unfortunately, faulty clients can be choosy about which estimates they use in their computation, especially in collusion with faulty landmarks. They can do so by claiming that measurements to a certain subset of the landmarks had been dropped, so that their coordinates were computed with only the remaining. Exactly how much a faulty client can manipulate its own coordinates within these boundaries is a subject for future work.

Chapter 5

Evaluation

To establish the feasibility of our approach to computing coordinates and determine how accurate we were in the presence of Byzantine faults, we implemented a simulation of our system. We found that our system's error in the presence of faulty landmarks was comparable to the error for coordinates computed when no landmarks were faulty. These results show that our system's approach is valid. In this chapter, we discuss our experimental setup, results, and interpretations thereof.

5.1 Simulation

Our simulation of our system is written in Java and consists of 1162 lines of code. It uses an event-driven simulation to represent the delivery of messages in the steps of this protocol and our own code for the gradient descent in the triangulation step.

Our simulation includes the modeling of the extra communications and computations needed to handle Byzantine clients, but we have not yet implemented a model for the faulty client that tries to manipulate its own coordinates. As mentioned in section 4.4.2, a faulty client may selectively ignore some of the measurements it receives in order to compute its coordinates. The simulation implements the simplified protocol in chapter 4 that does not deal with estimates over time.

We used our simulation framework to study several adversarial models and evaluate our system's effectiveness.

5.1.1 King data

To run our simulation on data consisting of actual Internet latencies, we used the King dataset from the P2PSim project [9], containing 1740 DNS servers and the actual measured round-trip times between them. The King method for collecting the pairwise round-trip times was to make a recursive query to server A through server B, and then make a query directly to server B, and compute the difference between the round-trip times, as described in [11].

We ran our system on a sample of 100 nodes selected from the King dataset. Since a small but nonzero fraction of the pairs of nodes in the King dataset do not have measurements between them, we selected our subset such that all $\binom{100}{2}$ pairwise measurements were present.

Vivaldi [6] was run on the 1740 nodes to generate the initial coordinates for landmarks to be used in simulations. This is not necessarily the best approach—it would perhaps be more valid to generate the coordinates by computing network coordinates on only the nodes in our sample or only the landmarks in an experiment—but it provides a reasonable approximation.

5.1.2 Adversary

The Byzantine adversary we chose for this simulation is quite powerful. We assume it has knowledge of all the inter-node round-trip times and that a malicious landmark is able to increase or decrease measured round-trip times; this actually serves to simplify our model because we do not require a lower bound constraint on the measurement that a faulty landmark may return.

Because we did not know a priori what kind of attack on our system would be most effective, we experimented with several different adversarial behaviors. In each adversarial model, the delay is the discrepancy between what the correctly measured round-trip time would be and the round-trip time that the client sees; delays can be positive or negative. Every client is subject to attack. The adversaries we studied are categorized below.

Constant delay: Malicious landmarks all cause measurements to be delayed by the same length of time.

Random delay: Malicious landmarks independently randomly choose a length of time to delay each message; the delay is chosen from a random distribution and can be positive or negative.

Random target: Malicious landmarks independently randomly choose a target coordinate for each client. Then, each landmark sets the delay so that the client sees as its measurement the metric space distance between its coordinates and the target.

Colluding target: Malicious landmarks randomly choose and agree upon a target coordinate for each client, and set the delay similarly to the random target case.

The latter two kinds of adversaries are motivated by the "repulsion" attacker in [13]; some other attack methods from that and other works [5] are not directly applicable because they employ lying about a faulty node's coordinates, which is not possible in our landmark system.

5.2 Experimental methodology

Based on the discussion in section 3.1.1, we chose $L = 10$. Each different setting of parameters was run in 200 experiments, each initialized with different random seeds. In each experiment, out of the 100 nodes, $2f + L$ were randomly chosen to be landmarks and the remainder were clients. We first ran the simulation with no faulty landmarks as the *control* sample, and then with f faulty landmarks chosen randomly from within the $2f + L$, which we refer to as the *experimental* sample. The control sample represents the best coordinates that can be computed from the landmarks, so that the choice of landmarks and the embedding error are controlled for, and just the effect of introducing faulty landmarks and using our protocol can be measured.

5.2.1 Measuring accuracy

To understand how well our system prevented faulty landmarks from disrupting the coordinate system, we studied several measures of error. The basis for our evaluation of our system's accuracy was the *predicted* and *observed* round-trip times between clients and non-faulty landmarks. In the context of coordinates and measurements, these values are defined for a client-landmark pair as follows.

Consider a client in an experiment. In the control sample, it computes its coordinates to be $\mathbf{x}$; in the experimental sample, it computes its coordinates to be $\mathbf{x}'$. For honest landmark i with coordinates $\mathbf{x_i}$, the observed distance is p_i, the measured inter-node round-trip ping time; the control predicted distance is $d(\mathbf{x}, \mathbf{x_i})$, as given by the metric of the space; and the experimental predicted distance is $d(\mathbf{x}', \mathbf{x_i})$.

To determine how much our experimental computed client coordinates deviate from the coordinates in the control sample, we considered our experimental sample's predicted distances relative to the control sample's predicted distances. This *closeness* is given by

$$\frac{|d(\mathbf{x}', \mathbf{x_i}) - d(\mathbf{x}, \mathbf{x_i})|}{d(\mathbf{x}, \mathbf{x_i})},$$

where $\mathbf{x}'$ is the client's coordinate.

The *relative error* for the predictions, a measure of how accurate they are for predicting the actual round-trip latencies, is given by comparing the predicted distance relative to the measured distance; it is

$$\frac{|d(\mathbf{x}, \mathbf{x_i}) - p_i|}{p_i}$$

for the control sample, and

$$\frac{|d(\mathbf{x}', \mathbf{x_i}) - p_i|}{p_i}$$

for the experimental sample.

These closeness and relative error values are aggregated at the client level. That is, in each experiment, each client's closeness or relative error was computed for all $f + L$ honest landmarks, and the mean of these values was taken to be a data point representing that client:

$$\text{Closeness:} \quad \frac{1}{f+L} \sum_{i \in \text{honest landmarks}} \frac{|d(\mathbf{x'}, \mathbf{x_i}) - d(\mathbf{x}, \mathbf{x_i})|}{d(\mathbf{x}, \mathbf{x_i})} \tag{5.1}$$

$$\text{Control relative error:} \quad \frac{1}{f+L} \sum_{i \in \text{honest landmarks}} \frac{|d(\mathbf{x}, \mathbf{x_i}) - p_i|}{p_i} \tag{5.2}$$

$$\text{Experimental relative error:} \quad \frac{1}{f+L} \sum_{i \in \text{honest landmarks}} \frac{|d(\mathbf{x'}, \mathbf{x_i}) - p_i|}{p_i} \tag{5.3}$$

Note that we measure accuracy using all the honest landmarks, but these might not be the same set of landmarks used to compute the coordinates.

Finally, data for all the clients across all experiments with the same set of parameters is considered together.

5.3 Adversary type

In this section, we vary the type of adversary and fix the other parameters. Since the adversary has no effect on the control sample, the result is that the control sample is identical across adversaries. Hence, we compare just the experimental relative errors to see which adversary is most effective at increasing them.

We first considered the constant delay adversary, with constant delays of -25, -10, 10, 25, 50, and 100 ms added to the round-trip time measurement. Figure 5-1 compares the experimental relative errors (from formula 5.3) for each choice of delay constant across different values of f; the mean, 10th percentile, and 90th percentile across all clients in all experiments are plotted for each choice of the delay. There are two conclusions to be drawn: first, the error is slightly worse for higher values of f; second, the error is worse for higher delays. These numbers appear consistent with Kaafar et al.'s observation that an attacker is not as effective when pulling nodes toward itself as when pushing them away [13].

Next, we compare all the different adversarial behaviors, as shown in figure 5-2. The random delay adversary chooses delays uniformly between -25 and 75 ms; the targets for the random and colluding target adversaries are chosen randomly within

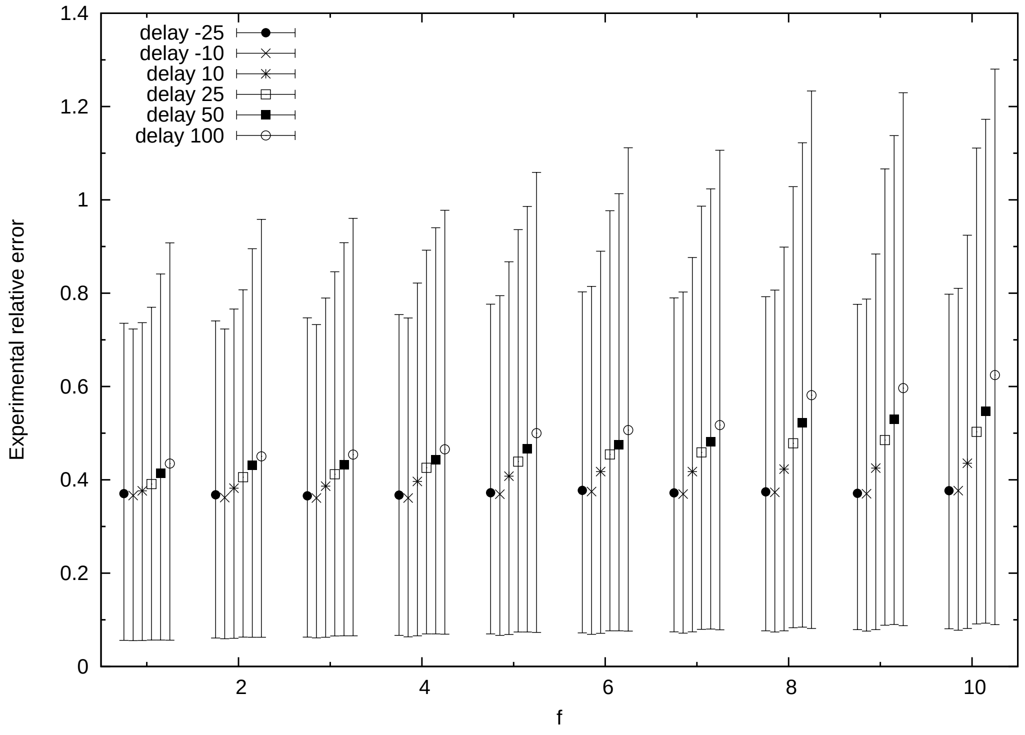

Figure 5-1: $\frac{|d(\mathbf{x}', \mathbf{x_i}) - p_i|}{p_i}$ for different delay values for a constant delay adversary. The mean is shown with 10th–90th percentile error bars.

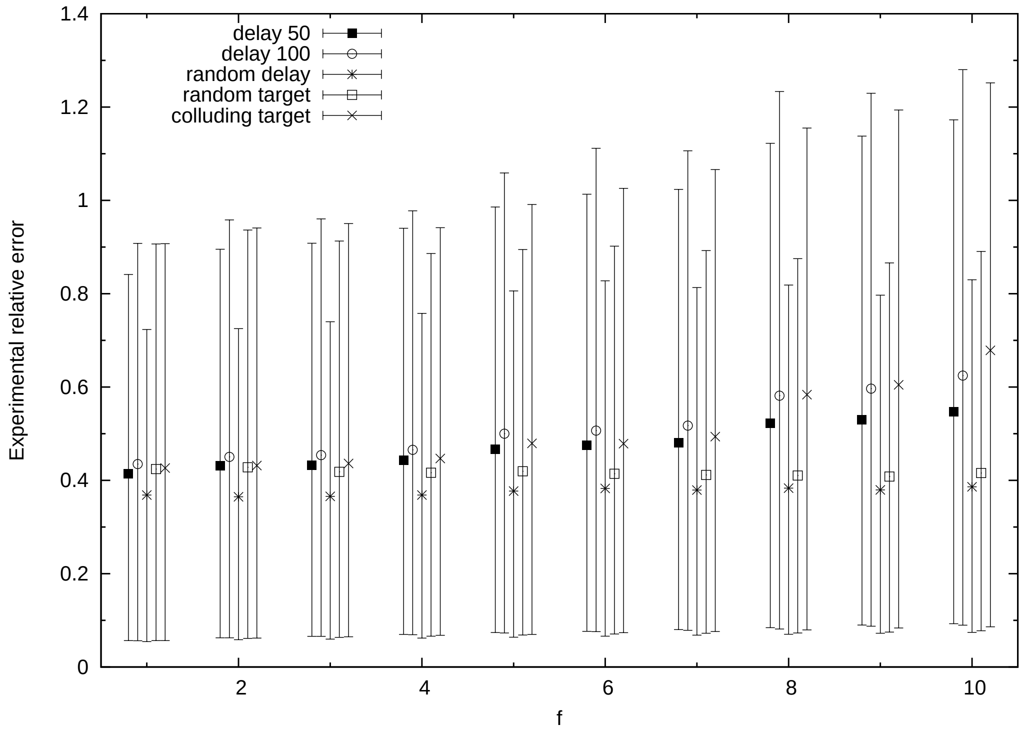

Figure 5-2: $\frac{|d(\mathbf{x}', \mathbf{x_i}) - p_i|}{p_i}$ for different adversary types. The mean is shown with 10th–90th percentile error bars.

a box in the coordinate space that bounds all the original coordinates computed with Vivaldi. Again, the data shows error increasing somewhat with f. It is not surprising that the colluding target adversary can make error worse than the random target adversary, or, based on the previous comparison, that the random delay adversary (for our choice of distribution) is less harmful than the constant delay adversary for longer delays.

Based on our data, since the colluding target adversary seemed to cause the greatest increase in error, especially as f increased, we considered the colluding target adversary for the remainder of our evaluation.

5.4 Closeness

We analyze how close the predicted distances in the experimental sample are to the predicted distances in the control sample. For each client, we plot its average closeness (formula 5.1) in figures 5-3 and 5-4 as probability and cumulative distribution functions respectively. A closeness of 0 indicates that every predicted distance with faulty landmarks is exactly the same as without. Some of a client's predictions may become closer to the observed value, but closeness considers them to be deviations from the original control predictions.

We find that even for $f = 10$, at the 90th percentile, the error is quite low—90% of clients have 27% or less average discrepancy in their predicted distances from the control's predicted distances. This result suggests that for the vast majority of clients, the coordinates computed in the presence of faulty landmarks give approximately the same information about the proximity of other nodes as the case there are no faults. It is also clear that as f increases, the distribution of client closeness extends further out as the experimental predicted distances approximate the control predicted distances less well.

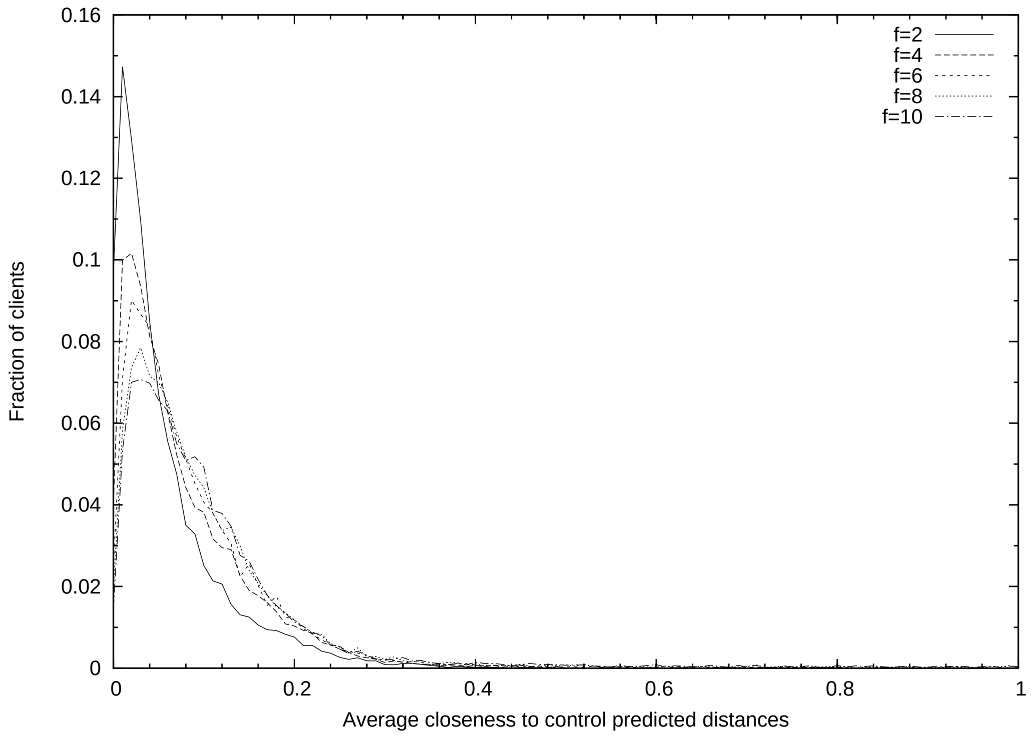

Figure 5-3: Probability distribution of clients' closeness.

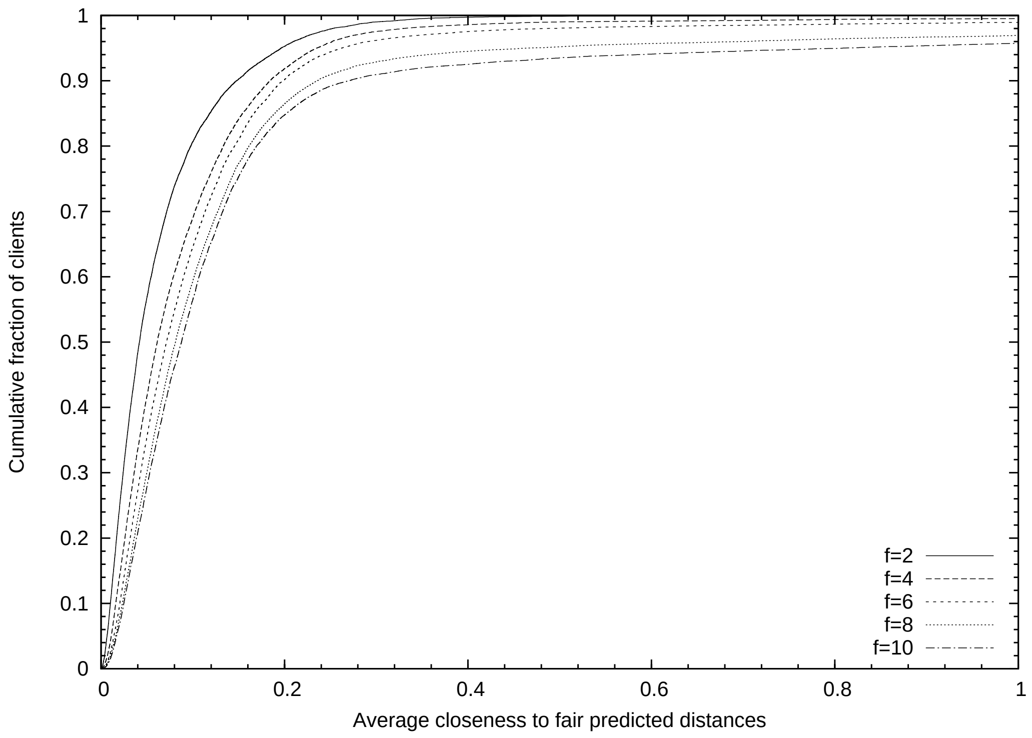

Figure 5-4: Cumulative distribution of clients' closeness.

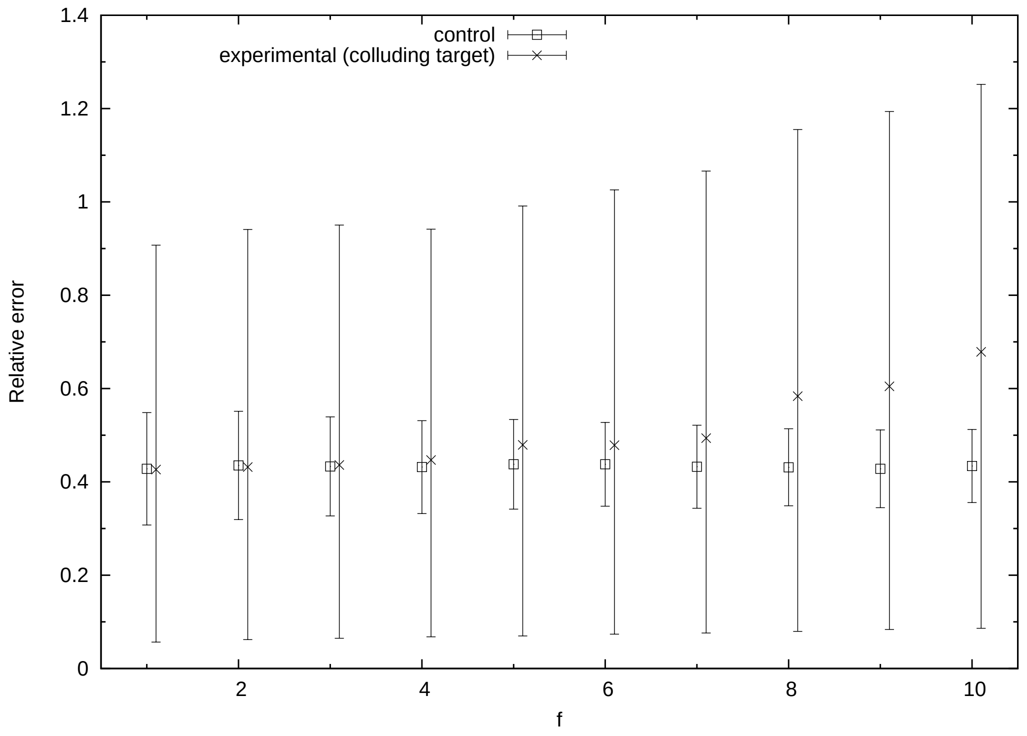

Figure 5-5: Relative error for experimental and control samples. The mean is shown with 10th–90th percentile error bars.

5.5 Relative error ratio

To compare the experimental sample against the control sample, we show in figure 5-5 the mean and 10th and 90th percentiles of the relative error for clients in the experimental and control cases. As expected, the control case remains unchanged as f increases, but beyond $f = 7$ the mean of the relative error over the clients begins to rise and is significantly outside the error bars.

For a more precise quantitative evaluation, for each client we divided its experimental relative error by its control relative error to get that client's ratio of average relative errors,

$$\frac{\dfrac{1}{f+L}\displaystyle\sum_{i\in\text{honest landmarks}}\frac{|d(\mathbf{x}',\mathbf{x_i})-p_i|}{p_i}}{\dfrac{1}{f+L}\displaystyle\sum_{i\in\text{honest landmarks}}\frac{|d(\mathbf{x},\mathbf{x_i})-p_i|}{p_i}}. \tag{5.4}$$

These ratios of average relative errors are plotted in figures 5-6 and 5-7 as probability

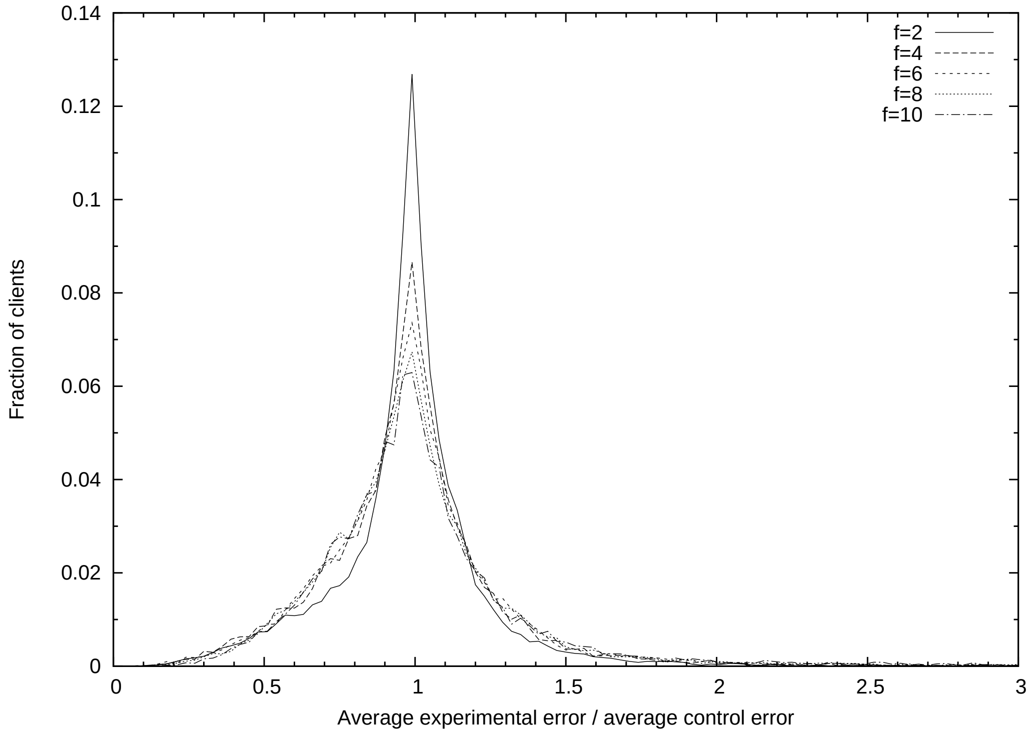

Figure 5-6: Probability distribution over clients of the relative error ratio.

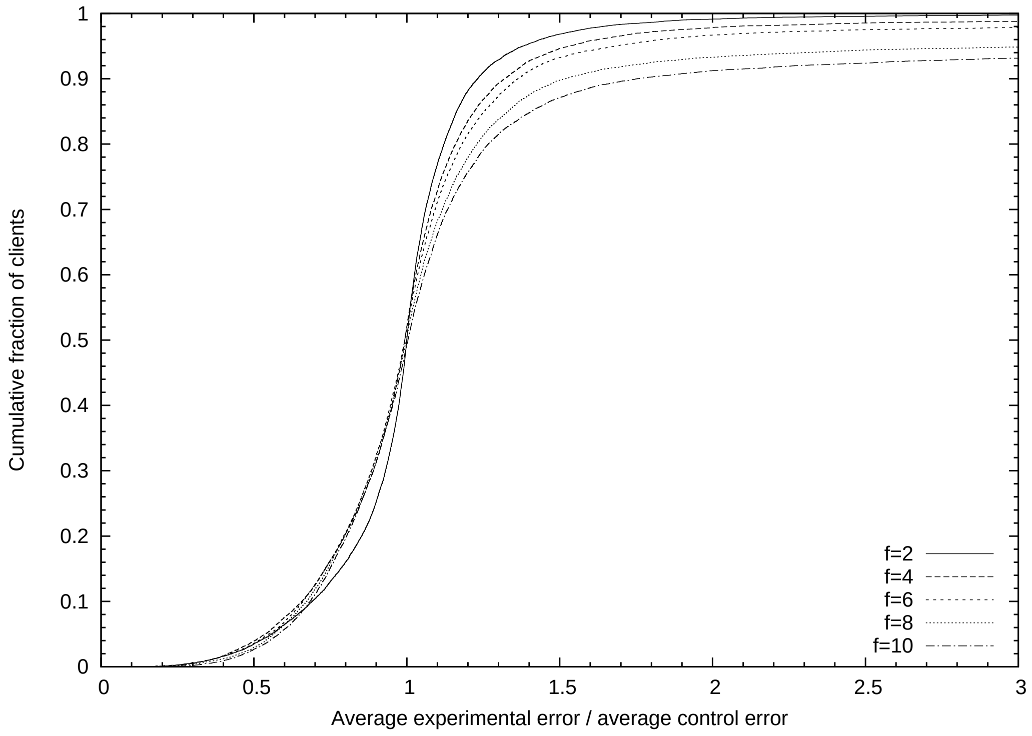

Figure 5-7: Cumulative distribution over clients of the relative error ratio.

and cumulative distribution functions respectively. Here, a ratio of 1 indicates that the client's experimental coordinates are, on average, just as good at predicting its round-trip distance to honest landmarks as the control coordinates. A ratio less than 1 indicates the client's coordinates are even better in the experimental sample than in the control (this effect is not entirely accounted for by the closeness evaluation).

Our results show that the bulk of clients in the experimental sample have very close to the same average relative error compared to the control. In fact, approximately half of the clients have an average relative error that actually improves over the control. Additionally, most of the clients do not have significantly distorted coordinates. For $f = 2$, for example, at the 90th percentile, only 10% of clients were more than 23% percent worse at predicting round-trip times to the honest landmarks than in the experimental case. Similarly, for $f = 6$, 90% of nodes had less than 36% worse relative error, and for $f = 10$, the 90% cutoff is at 75% worse error.

5.6 Summary

These results suggest that our system provides reasonably accurate coordinates even when there are f faults, the maximum tolerated in the system. Compared to the case when there are no faults, the resulting coordinates generally have a similar coordinate-space view of the round-trip distances, and the average relative error either decreases or increases by a small amount for all but a small fraction of the clients. However, our system of $2f + L$ landmarks does become less effective as f increases.

Chapter 6

Related work

Although there are countless works in the literature, some of which use network coordinates, that address the general problem of discovering location information in a network, only a few address the possibility of malicious nodes in the system and mitigating their effects on the rest of the system. We describe some of the relevant works in this area below.

6.1 Network coordinates and positioning

There are three main kinds of network positioning systems, classified by their approach to computing location information: landmark-based coordinate systems, decentralized coordinate systems, and systems that do not use coordinates.

Landmark systems: GNP [21] was a seminal landmark system that showed the then-surprising possibility of embedding network nodes into a low-dimensional Euclidean space with low error. Each client in GNP minimizes an error function using a simplex algorithm [20] to compute its coordinates. Virtual Landmarks [32] uses a Lipschitz embedding, in which the n-dimensional coordinates are the minimum distance to each landmark, and then applies principal component analysis to reduce the dimensionality of the coordinates.

Several systems have a set of global landmarks, but nodes do not have to communicate with them directly. NPS [22] uses a hierarchical structure and is based on GNP. In Lighthouse [25], each node computes its coordinates relative to some neighbors (that do not have to be the landmarks), then transforms its local-basis coordinates into the global basis for the coordinate space.

Decentralized systems: Vivaldi [6] is a frequently studied decentralized coordinate system that introduced the notion of height vectors; it uses the spring potential energy function as a basis for computing coordinates. Big-Bang Simulation [29] uses a model of force fields between points, in which points attract or repel to reduce error. These two systems model a physical simulation in which coordinates change to minimize the potential function. PIC [5] showed that a node's predictions of short distances and long distances were more accurate when the neighbors were chosen to be close to the node or at random, respectively, and that the best coordinates were computed from a mix of close and random neighbors. PCoord [16] uses a similar observation to try to maintain nearby neighbors in its computation; nodes use a simplex downhill algorithm [20] to compute coordinates and the triangle inequality to estimate unmeasured distances.

Non-coordinate systems: Meridian [36] does not use coordinates but places neighbors into concentric rings based on their measured distance; it appears to focus on the problem of routing in overlay networks. Octant [37] is a system for geolocating nodes, rather than placing them with respect to each other in a synthetic coordinate system; the interesting technique it uses is to define regions of certainty based on an error tolerance in each measurement and thresholding to find a region of space consistent with sufficiently many measurements.

iPlane [18] and Netvigator [28] do not treat the network as a black box, but use data about distances to intermediate routers from *traceroute* probes. iPlane attempts to build a structural model of the network topology; Netvigator provides a service to locate nearby landmarks and guesses inter-node latencies by using the triangle inequality on the two endpoints and any landmark.

6.2 Attacks

Kaafar et al. [13] helped to motivate this thesis by showing the vulnerability of Vivaldi to malicious nodes in the system. They identified three possible adversarial behaviors, which they called disorder, repulsion, and colluding isolation, and showed that even with a small percentage of faulty nodes in the system, Vivaldi's accuracy degraded dramatically. Zage and Nita-Rotaru [38] classified adversarial behaviors as inflating, deflating, or oscillating, based on whether they tended to cause nodes to incorrectly move or fail to move in adjusting to measurements. They studied the effects of different adversaries on Vivaldi and came to similar conclusions.

The adversary in PIC [5] is much more powerful; its behavior is determined by an optimization problem to be as harmful as possible, and it is assumed to be able to advertise false coordinates and decrease measurements within some limits. The adversary uses the simplex algorithm [20] to solve a multi-dimensional optimization problem over the coordinates and measurements for every malicious node.

We do not think that a more sophisticated attacker like PIC's would change the fundamental design of our system. However, there may be geometric weaknesses we have not yet discovered that such an attacker might exploit. Furthermore, some parameters, such as the number of landmarks needed to tolerate f faults, may need to be adjusted to maintain an acceptable level of error.

6.3 Reducing error

Several works apply techniques for reducing error, often against malicious adversaries, to construct new systems or to secure existing decentralized systems such as Vivaldi.

6.3.1 Triangle inequality violations

One source of embedding error encountered in Internet measurements is violations of the triangle inequality, where for nodes a, b, c, the measured a–b latency plus the b–c latency is less than the a–c latency. Any such nodes cannot be embedded without distortion in a metric space, which must satisfy the triangle inequality by definition [27]. Yet in many datasets in the literature [34, 6, 32], large fractions of the node pairs (a, b) were subject to a triangle inequality violation, where some node c existed such that $d(a, c) + d(c, b) < d(a, b)$. Many such violations are attributed to measurement uncertainty, but significant fractions (10–37% in some data sets [15]) had severe violations. Nevertheless, coordinate systems such as Vivaldi [6] are still able to compute good coordinates when there are no faults, although a small fraction of predicted distances will be inaccurate.

While many triangle inequality violations exist because of internet routing policies [17], malicious nodes may also be a source of these inconsistencies. Some systems therefore attempt to detect these violations to exclude measurements that violate the triangle inequality.

Nodes in PIC [5] detect triangle inequality violations and iteratively remove neighbors that show the worst violation from their coordinate computation until the remaining error is small.

Wang et al. [34] use the idea that measurements that give a high relative error between the predicted and observed distances are likely to cause severe triangle inequality violations. Neighbors are ranked based on how likely they are to cause triangle inequality violations, and the less likely half are kept in the computation. Unlike PIC, the authors only consider the inherent triangle inequality violations in the space, not malicious attackers.

6.3.2 Statistical analysis

Several other approaches use statistical analysis of the behavior of nodes' coordinates and how they change over time to predict when a measurement is anomalous and thus more likely to be faulty.

Kaafar et al. [12] use Kalman filtering to detect errors that can be introduced by malicious nodes. As a basis for correct behavior of coordinates over time, nodes use filter parameters from a nearby trusted node, while trusted nodes only communicate with other trusted nodes. Unfortunately, this approach relies upon an infrastructure of trusted nodes that are assumed never to be faulty; these nodes may need to constitute as much as 8% of the system if they are chosen randomly [12].

Zage and Nita-Rotaru [38] use the technique of outlier detection, borrowed from network intrusion detection systems. Spatial outliers are neighbors that report distances inconsistent with other neighbors, while temporal outliers are neighbors that are inconsistent over time; outliers beyond a threshold are removed from the coordinate computation.

Ledlie et al. [15] use latency filters on measurements from the same source, similar to our notion of estimates, and update filters to make coordinates more stable (rather than accurate). They also use a technique to incorporate measurements from neighbors that may be in only infrequent communication, a useful adaptation for a system that only includes passive measurements.

6.3.3 Voting

Veracity [30] is a system in which a node's coordinates are verified by a verification set, whose members approve the coordinates by measuring the round-trip time to the node and checking that it is consistent with the coordinate space distance. To prevent attackers from overrepresenting themselves, a node's verification set is chosen deterministically by hashing its IP address and looking up the value in a distributed hash table [2].

Veracity assumes a constrained-collusion Byzantine attacker, as introduced in [3], in which the faulty nodes are divided into small cohorts and only collude within their cohort. This assumption makes it much easier to show the feasibility of voting, as the faulty nodes are much less likely to collude to overwhelm a vote together. However, since the authors assume a minimum of 10 distinct cohorts, and each contains less than 10% of the nodes, it is a much weaker adversarial model than the Byzantine adversary assumed by our work and others [5, 38]. Another drawback is that in order to verify a node's coordinates, one must must contact its entire verification set, requiring an $O(\log n)$ DHT lookup for each of the set members.

Chapter 7

Conclusion

We designed and implemented a landmark-based network coordinate system that is able to provide accurate coordinates even when some of the landmarks are Byzantine faulty. Still, the final word on fault-tolerant network coordinates is far from being written.

Ideally, our system should not see accuracy decrease as f increases. That our experiments show this to be happening indicates that we may not have chosen precisely the number of landmarks to tolerate f faults. It is conceivable that could tolerate f faulty landmarks with fewer than $2f + L$ landmarks when f is small; conversely, we may want more than $2f + L$ landmarks for larger f to minimize the introduced error.

Because it was randomly determined which nodes were chosen as landmarks in experiments, our results could potentially be stronger if our landmarks were carefully selected to be well-distributed, although we attempted to control for this effect with the control sample—our experiments to compare the cases with and without faulty nodes each used the same selection of landmarks within our set of nodes. Conversely, note also that we randomly selected which landmarks were faulty. In the future, we plan to analyze a Byzantine attacker that has sufficient control to choose a particularly bad set of landmarks to become faulty.

It is possible that better algorithms and strategies exist for coping with faulty measurements. One motivation for using gradient descent for the nonlinear programming in the triangulation in section 4.3.2 was its relative ease of implementation.

In the future, it may be preferable to use an existing dedicated package such as OpenOpt [23] to solve the nonlinear programming problem in the triangulation step. Merz and Priebe [19] also recently suggested a replacement for standard nonlinear programming algorithms and claimed to perform better on computing network coordinates. It may also be telling to compare the error in coordinates computed with our estimate-removing strategy against the error from the exponential-time optimal strategy.

More data about short-term variance in measurements due to effects like network congestion may be gleaned by conducting larger-scale experiments on real networks such as PlanetLab, though these real-time measurements will have the disadvantage that it is more difficult for faulty landmarks in our experiments to decrease measured distances on the Internet. We hope that also collecting data on bandwidth usage may further support the argument for our system's practicality for deployment on Internet-scale networks.

Bibliography

[1] Mordecai Avriel. *Nonlinear Programming: Analysis and Methods.* Prentice-Hall, Inc., Englewood Cliffs, NJ, USA, 1976.

[2] Hari Balakrishnan, M. Frans Kaashoek, David R. Karger, Robert Morris, and Ion Stoica. Looking up data in P2P systems. *Communications of the ACM*, 46(2):43–48, 2003.

[3] Miguel Castro, Peter Druschel, Ayalvadi J. Ganesh, Antony I. T. Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02: Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, 2002.

[4] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI)*, New Orleans, Louisiana, February 1999. USENIX Association.

[5] Manuel Costa, Miguel Castro, Antony I. T. Rowstron, and Peter B. Key. Pic: Practical internet coordinates for distance estimation. In *24th International Conference on Distributed Computing Systems (ICDCS)*, pages 178–187, 2004.

[6] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM SIGCOMM '04 Conference*, Portland, Oregon, August 2004.

[7] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium*, San Diego, CA, USA, August 2004.

[9] 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/, 2004.

[10] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[11] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002)*, Marseille, France, November 2002.

[12] Mohamed Ali Kaafar, Laurent Mathy, Chadi Barakat, Kavé Salamatian, Thierry Turletti, and Walid Dabbous. Securing internet coordinate embedding systems. In *Proceedings of the ACM SIGCOMM '07 Conference*, pages 61–72, 2007.

[13] Mohamed Ali Kaafar, Laurent Mathy, Thierry Turletti, and Walid 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, New York, NY, USA, 2006. ACM.

[14] Jon M. Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. In *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS 2004)*, pages 444–453, 2004.

[15] Jonathan Ledlie, Paul Gardner, and Margo Seltzer. Network coordinates in the wild. In *Proceedings of 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2007.

[16] Li-wei Lehman and Steven Lerman. A decentralized network coordinate system for robust internet distance. In *3rd International Conference on Information Technology: New Generations (ITNG)*, pages 631–637, 2006.

[17] Eng Keong Lua, Timothy Griffin, Marcelo Pias, Han Zheng, and Jon Crowcroft. On the accuracy of embeddings for internet coordinate systems. In *IMC '05: Proceedings of the Internet Measurement Conference*, pages 11–11, Berkeley, CA, USA, 2005. USENIX Association.

[18] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iplane: an information plane for distributed services. In *OSDI '06: Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, pages 367–380, Berkeley, CA, USA, 2006. USENIX Association.

[19] Peter Merz and Matthias Priebe. A new iterative method to improve network coordinates-based internet distance estimation. In *ISPDC '07: Proceedings of the 6th International Symposium on Parallel and Distributed Computing*, page 25, Washington, DC, USA, 2007. IEEE Computer Society.

[20] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[21] T. S. Eugene Ng and Hui Zhang. Global network positioning: a new approach to network distance prediction. *Computer Communication Review*, 32(1):61, 2002.

[22] T. S. Eugene Ng and Hui Zhang. A network positioning system for the internet. In *USENIX Annual Technical Conference*, pages 141–154, 2004.

[23] Ukranian National Academy of Sciences. OpenOpt. http://scipy.org/scipy/scikits/wiki/OpenOpt.

[24] V. Paxson and M. Allman. Computing TCP's retransmission timer. RFC 2988 (Proposed Standard), November 2000.

[25] M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris. Lighthouses for scalable distributed location. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

[26] Sylvia Ratnasamy, Mark Handley, Richard M. Karp, and Scott Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM 2002, The 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002.

[27] Walter Rudin. *Principles of Mathematical Analysis, 3rd ed.* McGraw-Hill, Inc., New York, NY, USA, 1976.

[28] Puneet Sharma, Zhichen Xu, Sujata Banerjee, and Sung-Ju Lee. Estimating network proximity and latency. *SIGCOMM Computer Communication Review*, 36(3):39–50, 2006.

[29] Yuval Shavitt and Tomer Tankel. Big-bang simulation for embedding network distances in euclidean space. *IEEE/ACM Transactions on Networking*, 12(6):993–1006, 2004.

[30] Micah Sherr, Boon Thau Loo, and Matt Blaze. Veracity: A fully decentralized service for securing network coordinate systems. In *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008)*, February 2008.

[31] Aleksandrs Slivkins. Distributed approaches to triangulation and embedding. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 640–649, 2005.

[32] Liying Tang and Mark Crovella. Virtual landmarks for the internet. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference*, pages 143–152, 2003.

[33] Liying Tang and Mark Crovella. Geometric exploration of the landmark selection problem. In *Proceedings of Passive and Active Network Measurement, 5th International Workshop, PAM 2004*, pages 63–72, 2004.

[34] Guohui Wang, Bo Zhang, and T. S. Eugene Ng. Towards network triangle inequality violation aware distributed systems. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference*, pages 175–188, New York, NY, USA, 2007. ACM.

[35] M. A. Wolfe. *Numerical Methods for Unconstrained Optimization: an introduction.* Van Nostrand Reinhold Company Ltd., New York, NY, USA, 1978.

[36] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: a lightweight network location service without virtual coordinates. *SIGCOMM Comput. Commun. Rev.*, 35(4):85–96, 2005.

[37] Bernard Wong, Ivan Stoyanov, and Emin Gün Sirer. Octant: A comprehensive framework for the geolocalization of internet hosts. In *4th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[38] David John Zage and Cristina Nita-Rotaru. On the accuracy of decentralized virtual coordinate systems in adversarial networks. In *CCS '07: Proceedings of the 14th ACM conference on Computer and Communications Security*, pages 214–224, New York, NY, USA, 2007. ACM.

[39] Rongmei Zhang, Y. Charlie Hu, Xiaojun Lin, and Sonia Fahmy. A hierarchical approach to internet distance prediction. In *26th International Conference on Distributed Computing Systems (ICDCS)*, page 73, 2006.