Accurate and Provably Secure Latency Estimation with Treeple

Eric Chan-Tin and Nicholas Hopper
University of Minnesota
Minneapolis, MN
{dchantin, hopper}@cs.umn.edu

Abstract

A network latency estimation scheme associates a short "position string" to each peer in a distributed system so that the latency between any two peers can be estimated given only their positions. Proposed applications for these schemes have included efficient overlay construction, compact routing, anonymous route selection, and efficient byzantine agreement. This paper introduces Treeple, a new scheme for latency estimation, that differs from previous schemes in several respects. First, Treeple is provably secure in a strong sense, rather than being designed only to resist known attacks. Second, Treeple "positions" are not based on Euclidean coordinates, but reflect the underlying network topology. Third, Treeple positions are highly stable, allowing peers to retain the same position information for long periods with no maintenance. Finally, Treeple positions can be assigned to peers that do not participate directly in the scheme. We evaluate Treeple on a large internet dataset (with over 200,000 measurements) and find that on average, its latency estimates are within 26% of the true round-trip time. By comparison, Vivaldi, a popular but insecure scheme, has a median relative error of 25% on the same dataset.

1. Introduction

Network coordinate systems [7, 22, 23, 5, 24, 29] assign a *coordinate* to each node in a distributed system such that the distance between two nodes' coordinates provides a good estimate of the internet round-trip time between the nodes. The ability to estimate the network distance between arbitrary peers is useful in many cases: for example, finding the closest node to download content from in a content distribution network or file-sharing system [32]; reducing inter-ISP communication [18, 4]; reducing state in internet routers [1, 11, 15]; detecting Sybil attacks [8, 2]; and conducting byzantine leader elections [6]. Early network coordinate systems were shown to have reasonable accuracy and fast convergence, while later schemes added features such as coordinate stability under churn and measurement uncertainty [17, 16, 7].

Unfortunately, Kaafar *et al.* [12, 13] demonstrated that these early network coordinate systems were vulnerable to very simple attacks, in which adversarial nodes disrupt the coordinate system by claiming to have randomly chosen positions and adding random delays to their outgoing messages. In response, several schemes to mitigate these attacks have been proposed [35, 14, 30, 28, 33]. None of these schemes seems to have an explicit security goal, and at least one of the schemes [35] has already been shown to be insecure against more subtle attacks [3], potentially leading to a "penetrate and patch" cycle of schemes and attacks.

In this paper, we introduce a strong definition of security for latency estimation schemes that is robust to new attack types. Informally, our condition states that an adversary should be unable to influence the estimated distance between two honest nodes, and additionally, should only be able to increase the distance between pairs of nodes involving at least one adversarial node. We then consider whether previous secure network coordinate systems meet our definition. We demonstrate by simulation that several "secure" schemes are vulnerable to a variant of the "Frog-Boiling" attack [3], where an attacker injects a sequence of small inaccuracies – each of which appear plausible – that cumulatively result in the partitioning of the whole network into two independent clusters. We additionally show that even schemes that rely on trusted "landmark nodes" [22] can fail by allowing adversarial nodes to reduce their coordinate distance to targeted victim nodes in the network. The common fault underlying both of these vulnerabilities is that current network coordinate systems ignore the underlying network topology, and thus cannot distinguish between anomalous round-trip times due to adversarial manipulation and anomalies due to the topology and changing conditions in the network. We describe the common faults of current systems in more details in Section 2.3.

After defining security, we describe Treeple, our scheme for secure network latency estimation. Node positions in

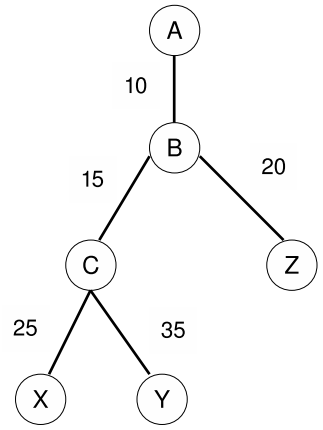

Figure 1. A tree built from one source to three destinations with link latency in milliseconds.

Treeple are not abstract coordinates but instead represent their position in the network graph in a way that allows efficient computation of the estimated latency between a pair of nodes. Assuming a small collection of trusted "vantage points", we prove that Treeple meets our security definition even in the presence of an arbitrary number of adversarial nodes.

To illustrate the idea behind Treeple, suppose we have a system with trusted vantage point A and peers X, Y, and Z. Using techniques similar to traceroute, A can discover the network paths from itself to each of the peers, constructing a tree with routers at the internal nodes and link latencies along the edges. A could then use this tree to compute an upper bound on the latency between X and Y as follows: first identify C, the least common ancestor of X and Y – $lca(X, Y)$ – in the tree. Then compute the sum of the latency between C and X and the latency between C and Y. Since there exists a path of this latency between X and Y, it represents an upper bound on the latency of the actual network path between the nodes. [1] For example, suppose that the tree is built as shown in Figure 1, and A is estimating the distance between nodes Y and Z. In this case, $lca(Y, Z) = B$ and the estimated distance is $35 + 15 + 20 = 70$ ms.

Notice that this approach does not suffer from the same issues as network coordinate systems. A malicious node can only affect the distance (real or estimated) between itself and another node, but not between two honest nodes. For example, if node Y from Figure 1 was malicious, it could affect the distance estimation between Y and Z. However, node Y cannot affect the distance estimation between nodes X and Z because it will not be on the same network path. Furthermore, using basic techniques like unpredictable nonces when measuring the RTT from A to Y can prevent Y from decreasing the distance estimation to other nodes. We describe the full scheme based on this idea in Section 3.

We evaluate Treeple on a large, real-world dataset in Section 4. For this dataset, Treeple has a median relative error of 0.26; This means that on average, half of the estimated latencies are within 26% of the true latency. For comparison, we simulated Vivaldi [7], a popular but insecure network coordinate system, on the same data set and the resulting coordinate scheme has a median relative error of 0.25. We note that the choice of Vivaldi is not arbitrary: all of the recently proposed "secure" or "stable" network coordinate schemes [35, 14, 30, 28, 33, 17] work by adding additional measures to Vivaldi coordinate calculations that attempt to discard anomalous inputs, and thus would have the same accuracy in the absence of an attacker. This shows that Treeple has accuracy comparable to network coordinate systems while providing provable security.

We additionally demonstrate that Treeple positions are highly stable: positions calculated on the first day of our dataset provide nearly the same accuracy nearly three weeks later. This has several important implications. First, because peers do not need to recalculate their positions the bandwidth overhead is significantly reduced compared to network coordinate schemes. Second, for the same reason, the "centralized" vantage points do not present a significant scaling challenge for Treeple. Finally, trusted vantage points do not present a central point of failure: the system can continue to function with high accuracy even if all vantage points are unavailable for an extended period.

2. Security

2.1. Threat Model

We model a network as a collection of N *end-hosts* connected by M *routers*, forming together a set of $N + M$ nodes. At any time t (we assume synchrony for simplicity only) there is a *network condition* $\chi(t)$ which assigns a *route* $route_\chi(n_1, n_2)$, a *return route* $rroute_\chi(n_1, n_2) = route_\chi(n_2, n_1)$ and a resulting round-trip time $rtt(n_1, n_2)$ to every pair of nodes (n_1, n_2). We allow each end-host n to measure both $rtt(n, n')$ and $route(n, n')$ for arbitrary hosts n'. We may extend our model to allow $\chi(t)$ to assign other conditions to peers and routers as well (for example, to model peer churn or packet loss), but we omit these details for clarity of presentation.

We allow an adversary to control an arbitrary number of end-hosts, but no routers. Thus an adversary can send mes-

[1] We note that due to the Internet's policy-based routing: (a) the RTTs measured to the intermediate nodes of the traceroutes will not typically reflect the link latencies exactly due to asymmetric return paths, and (b) "triangle inequality violations" [36, 19] can occur, in which case the path X-C-Y is shorter than the actual network path. However, in practice these measurements still provide a good approximation, as we show in Section 4

sages to arbitrary hosts, with arbitrary apparent origination, deviate from protocols in arbitrary collusive fashion, and arbitrarily inflate the measurement of $rtt(n, m)$ when m is adversarially controlled; however the adversary cannot effect the measurement of rtt between honest end-hosts, and cannot intercept, drop, or delay communication between honest end-hosts.

It may seem at first that excluding routers from adversarial control is a strong assumption. We argue, however, that excluding routers from adversarial control is reasonable in this setting; if an adversary could arbitrarily delay or redirect packets between any pair of peers (and thus affect the round-trip time between honest nodes) then a scheme to estimate network latency cannot succeed, since *any* latency estimate can be invalidated by the adversary. We note that under the current Internet architecture, an adversary that controls a *single* BGP speaker can exploit longest-prefix matching to receive traffic directed to arbitrary hosts. Furthermore, as recently shown by Goldberg, *et al.* [10], the Internet's policy-based routing is such that there exist single attackers that can intercept over 90% of all routes even when constrained by SBGP to use only existing routes. Goldberg *et al.* showed that their strategy was suboptimal, meaning the actual fraction of routes that a single attacker can intercept under SBGP may be even higher. Since we seek to provide provable security for the current Internet architecture, we must therefore assume that any attacker that controls a router is the worst-case attacker and can intercept and delay *every message sent* between peers.

We note additionally, that while the most desirable situation would be to resist such attacks, both the attacks in the current literature, and the attacks on existing schemes that we describe, fit within our threat model (and in fact, do not fully exploit the abilities we ascribe to an adversary.) Thus our threat model is *strictly stronger* than that considered in every previous work on the topic, and a scheme that provably resists our threat model will already rule out all of the known methods of attack.

2.2. Definitions

2.2.1 Latency Estimation Scheme

A *latency estimation scheme* for a set Peers of end-hosts consists of four distributed protocols:

- A global initiation protocol GlobalInit that initializes the global parameters of the scheme.
- An interactive protocol LocalInit which initializes the state of a peer.
- An interactive protocol Update(P) in which the peer P uses its local state and global parameters to compute a new value for its *position* $\rho(P)$, possibly after interacting with other peers.
- An algorithm Distance(ρ_1, ρ_2) which computes an estimated latency between positions ρ_1 and ρ_2.

Peer P runs LocalInit on joining the system, and then at regular time intervals τ, it calls Update(P) to update its position. The most important functional goals for a latency estimation scheme are:

Accuracy. Informally, a scheme is accurate if two nodes that compute positions ρ_1 and ρ_2 have network latency that is close to Distance(ρ_1, ρ_2). The typical measure of a scheme's accuracy used in the literature is the *median relative error* of peer P,

$$\underset{P' \in \text{Peers}}{\text{median}} \frac{|\text{Distance}(\rho(P), \rho(P')) - rtt(P, P')|}{rtt(P, P')} .$$

We note that this measure inherently compares a scheme's accuracy to the "ground truth:" stating that a scheme has median relative error c is stating that on average, 50% of the estimates are within a factor c of the true latency, while 50% are not. Thus, lower values for median relative error equate to better estimates. Ideally, a scheme would achieve relative error of 0 on all estimates. However, it is not hard (logically) to construct a network that has incompressible latencies, so that any scheme to represent positions by strings of length $o(N)$ must be incorrect on at least a constant fraction of the estimates.

Stability. A latency estimation scheme is *stable* if positions computed at time t still provide good accuracy at time steps $t' > t$. This can be computed by computing the distance in coordinates at time t and comparing it to the latency at time t', in the calculation of median relative error.

Efficiency. A latency estimation scheme is not very useful if the bandwidth required to transmit positions exceeds the $O(N^2)$ bandwidth required to simply have all nodes measure pairwise RTTs, and similarly if the distance computations from positions is inefficient. Ideally, the size of positions and the time required to compute distances should be essentially independent of the number of peers.

We note that all of these aspects of a scheme may depend to a large extent on the topology of the underlying network. In Section 4, we use a large set of Internet measurements to compare the predictions of Treeple to measured latencies, measure the stability of Treeple positions, and evaluate the size of Treeple positions, along with the average computational load.

Triangle Inequality Violations. We note that several measurement studies [36, 19] have reported that as many as 5% of all node "triangles" (N_1, N_2, N_3) violate the triangle inequality, that is, $rtt(N_1, N_3) > rtt(N_1, N_2) + rtt(N_2, N_3)$;

we call these triangles "triangle inequality violations" or TIVs. These occur due to the fact that Internet routing is policy-based, rather than distance-based: each autonomous system chooses among possible routes to a given destination based primarily on the cost it will incur by sending packets along the various routes. Thus any system for estimating latencies that satisfies the triangle inequality – including Euclidian distances as in Vivaldi and GNP, and tree distance as in Treeple – must be inaccurate on at least one pair of nodes in each TIV. However, this does not preclude having acceptable accuracy on the remaining 95% of node pairs; indeed, previous studies have shown that in the absence of attacks, Vivaldi achieves low median and 90th percentile relative errors, while we show in Section 4 that the same is true of Treeple.

2.2.2 Security Goal

To motivate our Security definition, we consider an hypothetical ("ideal") system in which we can instantaneously ask any node to measure its RTT to another node. In this setting, an adversary is unable to alter the RTT between any pair of honest nodes. On the other hand, any measurement in which at least one of the nodes is an adversary can be increased, but if the query includes some challenge value, it cannot be decreased below the actual network latency. Since an adversary can *always* increase apparent RTTs involving an adversarial node by delaying responses, this hypothetical scheme would provide the best security we could hope to provide without complete knowledge of the underlying topology. However, since an explicit goal of Treeple is to have short position strings, we will relax this notion slightly to only consider how the adversary can influence Treeple's latency estimates. We will consider a latency estimation scheme to be secure if an adversary cannot influence the estimated latency between honest nodes, and by deviating from the protocol, can only increase the estimated latency between a pair involving at least one malicious node.

Formally, we define security as follows. Let Π be a latency estimation scheme. Fix a network, a sequence of network conditions, and a set $\mathcal{A}$ of adversarial nodes. We will compare random variables $\mathcal{H}$ and $\mathcal{M}$, where $\mathcal{H}$ is an execution trace of Π in which all peers behave according to Π at all time steps; whereas in the execution trace $\mathcal{M}$ the nodes in $\mathcal{A}$ behave arbitrarily, subject to computational restrictions. Each execution trace consists of the set of all messages sent, rtt and $route$ measurements taken, and positions $\rho_{i,t}$ computed by a peer. We note that in the adversarial trace, misbehaving nodes are not constrained to use a new position at each time step. We compare the sequence of positions $\mathcal{H}.\rho_{i,t}$ and $\mathcal{M}.\rho_{i,t}$ held by each peer i at each timestep t in the traces $\mathcal{H}$ and $\mathcal{M}$. We say that Π is *secure* if the following properties hold with all but negligible probability:

1. For all times t, for all $i, j \in \mathsf{Peers} \setminus \mathcal{A}$,

$$\mathsf{Distance}(\mathcal{H}.\rho_{i,t}, \mathcal{H}.\rho_{j,t}) = \mathsf{Distance}(\mathcal{M}.\rho_{i,t}, \mathcal{M}.\rho_{j,t})\,.$$

2. For all times t, for all i, j such that $i \in \mathsf{Peers}$ and $j \in \mathcal{A}$, there exists time $t' \in (t - \tau, t]$ such that $\mathsf{Distance}(\mathcal{M}.\rho_{i,t}, \mathcal{M}.\rho_{j,t}) \geq \mathsf{Distance}(\mathcal{H}.\rho_{i,t}, \mathcal{H}.\rho_{j,t'})$.

Informally, condition 2 relaxes our intuitive notion so an adversary can use positions computed at a different time within the update period, since there is no way to prevent an adversary from choosing when it updates its position.

Relationship between security and accuracy. We note that under our definition, security and accuracy are orthogonal. In particular, a scheme may be accurate but not secure: schemes like Vivaldi, GNP, Big Bang, and so on can produce latency estimates that are within 10%-20% of the actual latency, depending on the evaluation, but fall to trivial attacks. On the other hand, it is trivial to produce a scheme that is secure but not accurate: if we assign each node a distinct position and then predict that the latency between any pair of distinct nodes is 100ms, then the adversary clearly cannot influence latency estimates between any pair of nodes, trivially satisfying our definition. Of course this scheme will be highly inaccurate; the challenge lies in simultaneously achieving security and acceptable accuracy. We note that once a scheme is *both* secure, and accurate with no attackers, then it will continue to accurately estimate pairwise latencies between honest nodes when under attack. This definition therefore allows us to evaluate security and functionality in a modular way: given a proof that a scheme is secure, it is sufficient to evaluate its accuracy with no adversarial nodes.

2.3. Failures of Previous Network Coordinate Systems

Network coordinate systems can be categorized into centralized schemes with trusted nodes [23, 22] and decentralized schemes [7, 29]. Centralized network coordinate systems consist of some trusted nodes which communicate with each other to compute their coordinates; other nodes can contact a subset of those trusted nodes to obtain their coordinates. In a decentralized scheme, each node contacts a different set of peers to compute and update its coordinates. Here we briefly demonstrate that (1) *having trusted nodes does not ensure that a scheme meets our security definition*, and (2) previous decentralized schemes designed to withstand attacks are vulnerable to a variant of the "frog-boiling" attack [3].

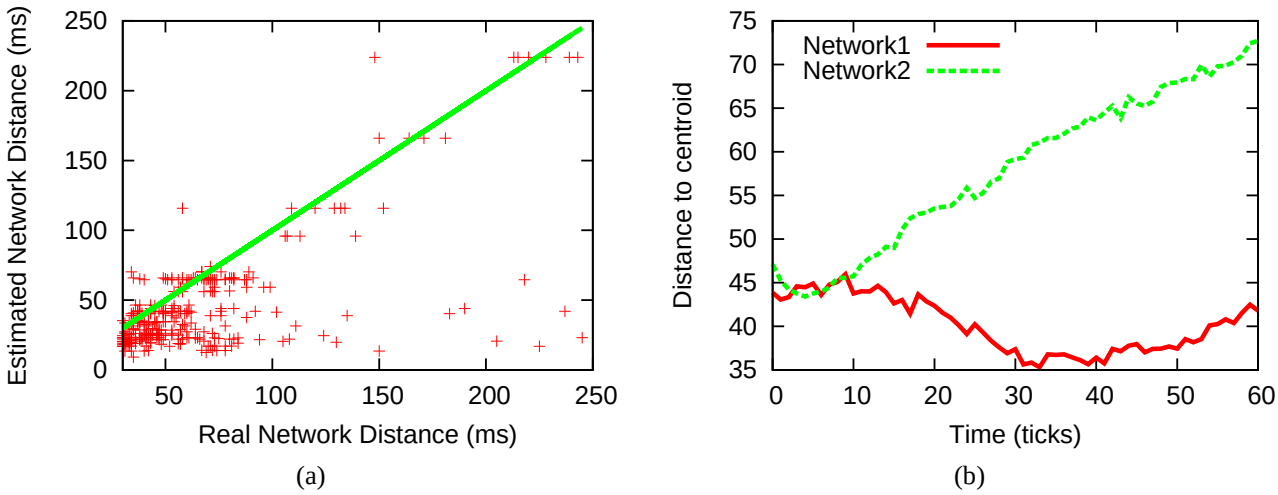

Figure 2. (a) Results of the targeted close node attack on GNP. Each point represents one simulated attack, and compares the resulting coordinate distance to the underlying network distance between the attacker and the targeted node. (b) The Network-Partition attack on Veracity showing that the network is being divided into two subnetworks.

2.3.1 Manipulating coordinates in GNP

GNP [22] is a landmark-based network coordinate scheme; the authors do not claim security against attacks but it might seem intuitively appealing that if landmarks' coordinates are digitally signed, and nodes obtain a digitally signed messages from each landmark attesting to RTT measurements, the scheme could be secure. Although this would seem to prevent an adversary from influencing the estimated distance between honest nodes, however, we show that it is possible for an adversary to influence the protocol to artificially decrease the estimated distance between an adversarial node and a targeted honest node.

We implemented a very simple attack that demonstrates the feasibility of this goal. In our attack, the adversarial node A knows the coordinates of the node T it wants to target, so A can compute the distance between T and each landmark node. Then when a given landmark Λ_i measures $rtt(\Lambda_i, A)$, A attempts to make the result as close to $rtt(\Lambda_i, T)$ as possible, subject to the constraint that the adversary cannot cause the measurement to be smaller than the underlying network distance. We repeatedly simulated this process using the code and matrix topology from [31]. For each simulation run, we randomly picked one victim node and one attacker node from a set of 101 nodes with 10 of these nodes as landmarks, subject to the constraint that the victim was not already "close" to the attacker (RTT less than 30ms). Figure 2(a) shows that this simple attack is very successful: in nearly all cases, A receives a coordinate that is closer to T than the underlying network distance.

2.3.2 Partitioning a Veracity network

Decentralized network coordinate systems use a fully distributed algorithm to compute the optimal coordinates of the nodes in the network. Several secure mechanisms have been proposed. Broadly speaking, they fall into anomaly/outlier detection systems [35, 14], reputation system [28], and distributed reputation systems [30, 33]. It was shown in [3] that an outlier detection mechanism using the Mahalanobis distance [35] can be defeated using the "Frog-Boiling" attack. The Frog-Boiling attack is analogous to the popular account that if a frog is put in hot boiling water, it will jump out, but if it is put in cold water, and the temperature of the water slowly increased to boiling, the frog will stay in the water and boil. We show that other secure mechanisms can be defeated in a similar fashion. Figure 2(b) shows the network partition variant of the attack on a Veracity [30] simulation. Veracity is a distributed reputation system which verifies the self-reported coordinates of nodes and prevents nodes from artificially delaying their RTT by too much. The figure shows that the Frog-Boiling attack is effective in partitioning a Veracity-secured network coordinate system – 10% of the network were attackers.

The main problem with current network coordinate systems is that they have to work under dynamic network conditions. Thus, secure network coordinate systems have to accept changes in RTT and link conditions. The Frog-Boiling attacker produces small and consistent lies so that its updates are accepted since they are mistaken for the normal fluctuations in RTT. Current secure schemes thus cannot differentiate between a malicious update and a normal

update due to changing link conditions. Therefore, some sort of attack (Frog-Boiling or other) will still be possible to disrupt the network coordinates, rendering the estimated network distance to be useless since it will be greatly different from the real network distance. This problem is inherent in both centralized and decentralized network coordinate systems. An attacker can lie in small but consistent ways to disrupt a decentralized network coordinate system, as shown by partitioning the network using the Frog-Boiling attack. In centralized network coordinate systems, a node, knowing the coordinates of its target node, can lie in such a way that its coordinates, as computed by the landmarks, are close to the coordinates of the target node. Although the attacker is not directly lying to the target node by changing its RTT, and only lying to the landmarks, the attacker's estimated distance to the target victim will be affected, which contradicts our definition of a secure latency estimation system. We note that neither of these attacks requires an overwhelming fraction of malicious peers, and both attacks work even in the presence of trusted routers.

3. Treeple

Suppose that we have a single trusted vantage point and wish to build a secure network latency estimation scheme for the current Internet. A very simple way to incorporate network topology into Treeple positions is to have the trusted node measure the network path to each peer, for example using repeated calls to traceroute, and measure the RTT to each router along the path. The position that the vantage point assigns to each peer would then be the signed path from the trusted node to the peer, including the RTTs to each node along the path. To compute the distance between two peers A and B given their positions, we could find the last "common ancestor" C on each of the paths and then estimate that the distance between the peers is the distance between A and C plus the distance between C and B, since a path of this distance exists between A and B.[2] It is easy to see that while it may be inaccurate, this scheme meets our stated security goal, assuming that malicious peers cannot interfere with the routing infrastructure: in this case the paths from the trusted node to any two honest nodes would not involve malicious nodes, and only the final hops to malicious nodes could be impacted, by delaying responses to the trusted node's traceroute request.

An alternate view of this system is that the trusted node is computing the tree of shortest paths between itself and other peers, and "embedding" the network into this tree metric. It is clear that for some pairs of nodes the tree distance could be larger than the actual network distance, because many network links will not be included in the trusted node's shortest path tree.

To address this, we choose k topologically distinct vantage points to repeat this process: a peer's coordinate becomes an ordered k-tuple of signed routes (one from each trusted vantage points), and the distance between A and B becomes the minimum of the distances computed from each of the k trees. Again, it is easy to see that the security of the scheme holds for this variant: no adversarial node can interfere with the route and RTT measurements involved in computing honest nodes' positions; so all honest node coordinates will be the same regardless of adversarial behavior. And since adversarial nodes can only increase RTT measurements they cannot appear closer to other nodes by deviating from the protocol. Intuitively, adding the extra vantage points increases the probability of discovering the network links used by the actual route between two nodes, thus improving the accuracy of latency estimates.

3.1. Complete Description

For completeness, pseudocode for the component algorithms of Treeple is shown in Figure 3. Treeple assumes the existence of a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ that is existentially unforgeable against chosen message attack: any efficient program given access to a verification key vk and a signing oracle for the corresponding signing key sk cannot produce a correct (message, signature) pair with a message that was not a previous signing oracle query, except with negligible probability in the security parameter. Additionally we assume that each end-host h has access to two functions: $\mathtt{h}.route(\mathtt{g})$ returns a list of routers along the path from h to g; and $\mathtt{h}.rtt(\mathtt{g})$ returns the round-trip time between h and g. For clarity, the "system parameters" generated in GlobalInit are passed as implicit arguments to other procedures. Finally, we assume a fixed set of k trusted vantage points whose addresses are included in the global parameters, but are not chosen by the protocols. These nodes serve as a "root of trust" in the scheme. We briefly discuss algorithms for choosing from among several possible vantage points in Section 4.

3.2. Security

Theorem 1. *Assuming the set $\{T_1, \ldots, T_k\}$ of vantage points are honest,* Treeple *is a secure network latency estimation scheme.*

Proof. Fix a network graph and condition sequence χ, along with a set of adversarial peers $\mathcal{A}$ such that $\mathcal{A} \cap \{T_1, \ldots, T_k\} = \emptyset$. Suppose that there exists a pair of peers (A, B) that violate the security condition of section 2; we will show that, except with negligible probability, the existence of this pair must imply a signature forgery. We let

[2] Due to the complexities of Internet routing this path is unlikely to be used, but it may still represent a good approximation.

Define Treeple.GlobalInit:
For each trusted $T_i \in \{T_1, \ldots, T_k\}$:
T_i: choose $(vk_i, sk_i) \leftarrow \mathsf{Gen}(\lambda)$.
T_i: **send** (i, vk_i) to $\{T_1, \ldots, T_k\}$.
Output: $\langle vk_1, \ldots, vk_k \rangle$.

Define Treeple.LocalInit(N):
set $\mathsf{pos_N} \leftarrow \mathsf{getPosition}(\mathsf{N}, time())$.

Define Treeple.Udpate(N, t):
if $length(\mathsf{pos_N}) < k$
or $\mathsf{pos_N}$ is stale:
set $\mathsf{pos_N} \leftarrow \mathsf{getPosition}(\mathsf{N}, t)$.

Define Treeple.getPosition(N, t):
N: choose $rid_t \leftarrow_R \{0,1\}^\lambda$. Set $\mathsf{pos_N}.\mathsf{rid} = rid_t$.
Foreach $T_i \in \{T_1, \ldots, T_k\}$ do:
N: choose $rid_i \leftarrow_R \{0,1\}^\lambda$.
N: **send** `pos-request`(rid_t, rid_i, t) to T_i
T_i: **on** `pos-request`(rid_t, rid_i, t) from N:
T_i: set $\mathsf{rt} \leftarrow \langle\rangle$
T_i: $\mathsf{rt}.rid \leftarrow rid_i$.
T_i: $\mathsf{rt}.t \leftarrow t$.
T_i: if $|t - time()| > \tau$: **abort**.
T_i: compute $\langle r_1, \ldots r_\ell \rangle = T_i.route(\mathsf{N})$.
T_i: for each $r_j \in \langle r_1, \ldots, r_\ell = \mathsf{N} \rangle$:
T_i: $\mathsf{rt.host}_j \leftarrow r_j$.
T_i: $\mathsf{rt.rtt}_j \leftarrow T_i.rtt(r_j)$
T_i: $\mathsf{rt.sig} \leftarrow \mathsf{Sign}_{sk_i}(\mathsf{rt.host}, \mathsf{rt.rtt}, rid_t, rid_i, t)$.
T_i **send** `route-reply`(rt) to N.
N: **on** `route-reply`(rt) from T_i:
N: if $\mathsf{Verify}_{vk_i}((\mathsf{rt.host}, \mathsf{rt.rtt}, rid_i, rid_t, t), \mathsf{sig})$:
N: set $\mathsf{pos_N.route}_i = \mathsf{rt}_i$.
N: else, set $\mathsf{pos_N.route}_i = \perp$.
Output: $\mathsf{pos_N}$.

Define Treeple.Distance($\mathsf{pos}_A, \mathsf{pos}_B$):
set $dist \leftarrow \infty$.
if ($|\mathsf{pos}_A.t - \mathsf{pos}_B.t| \leq \tau$):
For each $T_i \in \{T_1, \ldots, T_k\}$ do:
if ($\mathsf{verify}(i, \mathsf{pos}_A)$ and $\mathsf{verify}(i, \mathsf{pos}_B)$):
set $lca \leftarrow$ find-lca($\mathsf{pos}_A.\mathsf{route}_i, \mathsf{pos}_B.\mathsf{route}_i$).
set $d_i \leftarrow (\mathsf{pos}_A.\mathsf{route}_i.\mathsf{rtt}_A - \mathsf{pos}_A.\mathsf{route}_i.\mathsf{rtt}_{lca}) +$
$(\mathsf{pos}_B.\mathsf{route}_i.\mathsf{rtt}_B - \mathsf{pos}_B.\mathsf{route}_i.\mathsf{rtt}_{lca})$.
Update $dist \leftarrow \min(dist, d_i)$.
Output: $dist$

Define find-lca(rt_1, rt_2):
Output: $max\{j | rt_1.\mathsf{host}_j = rt_2.\mathsf{host}_j\}$.

Define verify(i, ρ):
if $\rho.\mathsf{route}_i = \perp$: **output** False
else:
set $m_i \leftarrow \langle \rho.\mathsf{route}_i.\mathsf{host}, \rho.\mathsf{route}_i.\mathsf{rtt}, \rho.\mathsf{route}_i.\mathsf{rid}, \rho.\mathsf{rid}, \rho.t \rangle$.
output $\mathsf{Verify}_{vk_i}(m_i, \rho.\mathsf{route}_i.\mathsf{sig})$.

Figure 3. Treeple algorithms, assuming the existence of a digital signature scheme (Gen, Sign, Verify) **and a pre-selected set of trusted vantage points** $\{T_1, \ldots, T_k\}$**. Here a** *position* **consists of a time** t**, a global random identifer** rid**, plus an indexed array** route**, where each entry** route_i **is either** $\perp$ **or a record consisting of indexed arrays** host **and** rtt **along with a signature field and a local identifier.**

α denote the total number of messages sent by adversarial nodes in the adversarial trace.

First, notice that for any honest peer h, at any time t, $\mathsf{pos_h}$ must be identical in both the adversarial and non-adversarial execution traces, except with negligible probability. This is because the "request identifier" rid_i generated in getPosition(h), combined with signature verification on received route messages, ensures that h only updates its coordinates when it receives authentic responses to its own requests, from trusted nodes. Since calls to getPosition(h) and are only initiated by h and our adversarial model excludes dropping and interception of honest messages, it follows that in any pair of execution traces with identical network conditions, an honest node will have equivalent positions.

Specifically, an honest node only changes its position when it receives a message that is signed and contains its most recent, randomly chosen identifier. Since adversarial nodes do not see the requests generated by honest nodes, the probability of correctly guessing a request identifier in α attempts is at most $\frac{\ell n \alpha}{2^\lambda}$ (where ℓ is the length of the trace, n is the number of peers, and λ is the length of request identifiers). Given that the adversary does not correctly guess request identifiers, it can only cause an honest node to accept a position that is different than the non-adversarial trace by generating a response message route′ that is apparently signed by one of the trusted vantage points. Notice that given the sequence of network conditions and a signing oracle, it is easy to generate all the messages a set of adversarial nodes would see in the adversarial execution (because the honest nodes follow the protocol), and thus it follows that this route′ and its signature would constitute a forgery. If we denote the (negligible) probability of a forgery with $n\ell + \alpha$ signature queries by ϵ, then the probability of this event is at most $k\epsilon$ by the standard reduction that guesses which trusted party the adversary will forge against.

Thus, if the pair (A, B) is honest, the presence of adversaries is irrelevant: $\mathsf{Distance}(\mathsf{pos}_A, \mathsf{pos}_B)$ will be the same in both traces. On the other hand, consider the variable pos_A assigned to an adversarial node A at time t. Since the adversarial node initiates the same requests in both traces, it can only manipulate its position by either manipulating the

measurement of $T_i.rtt(A)$ for some T_i (by assumption the measurement of $route(T_i, A)$ and $T_i.rtt(x)$ for $x \neq A$ are not vulnerable to manipulation) or by substituting a different value for some route_i. By assumption on the rtt functionality, the first type of manipulation will only inflate the distance between A and its "last hop", which will inflate the distance to other nodes (uniformly). So the only remaining option to violate the security condition is to replace some route_i. The requirement that all route messages in a position have the same request identifier prevents "mix-and-match" substitution of routes between requests. Dropping any route message that does not give the minimal distance to a particular position will not affect the distance calculation, while dropping the minimal distance route will only increase the distance. Finally, producing a signed route_i with different rtt or host entries would constitute a forgery, and thus an adversary would again successfully produce this forgery with probability at most $k\epsilon$. □

4. Evaluation

4.1. Experimental Setup

To evaluate our approach, we used the iPlane [20] dataset. This data set contains the results of periodic traceroutes from 250 Planetlab [25] nodes to all other Planetlab nodes and thousands of other IP addresses. We note that the iPlane dataset is "live"; every day, each of the PlanetLab nodes performs multiple traceroutes to over 100,000 IP addresses and publishes the results. We downloaded the traceroute datasets from Dec 1st, 2009 to Dec 22nd, 2009. The dataset presented 250 possible trusted nodes to choose from; each trusted node contacted more than $130,000$ IP addresses on average, and each tree constructed from the traceroutes contained on average $200,000$ unique nodes (including intermediate nodes). In constructing the paths from trusted nodes to peers, we always used the minimum RTT measured at each hop, and when repeated traceroutes resulted in different routes, we selected the shorter route.

To determine the accuracy of a given set T of vantage points, we considered all pairs of nodes (A, B) such that (i) all vantage points in the set had successfully completed a traceroute to both A and B, allowing us to compute positions pos_A, pos_B; and (ii) we had measured the RTT between A and B. For each such pair, we computed the relative error,

$$\frac{|\mathsf{Distance}(\mathsf{pos}_A, \mathsf{pos}_B) - RTT_{A,B}|}{RTT_{A,B}},$$

to measure the accuracy in estimating the latency between A and B using T; lower relative error indicates a better estimate of the latency. Because the iPlane measurement aparatus does not retry traceroutes that fail, the size of the evaluation set will vary across sets T; in all cases the size was over 200,000. In order to compare between these evaluation sets, we used the median relative error for each. Additionally, for our "best choice" of 20 vantage points T, we also measured the size of the coordinates pos_A and pos_B and the number of node comparisons required to estimate the distance.

4.2. Selecting a set of vantage points

Determining the accuracy of Treeple is not as straightforward as for a network coordinate system. The relative error is still used to determine whether an estimated network distance is accurate – the lower the error, the more accurate the estimation. However, using the iPlane dataset, there are 250 possible vantage points. It is clear that, in general, different vantage point sets will produce different accuracy. An important question, both for evaluation and eventual deployment, is how to select a good set of vantage points.

Determining the "best possible" single vantage point is relatively straightforward – compute the median relative error for each pair of end hosts' network distance estimation, for each of the 250 vantage points, and the one with the lowest median relative error is the most accurate. When $k = 2$, we could pick the very best combination of **any** two vantage points such that the combination would result in the lowest median relative error among all the possible combinations (total of $250 \times 249 = 62,250$ combinations). However, this approach is not scalable, as when $k = 3$, there are a possible $250 \times 249 \times 248 = 15,438,000$ combinations to choose from. Although this approach provides the very best combination of vantage points to produce the lowest median relative error, it does not scale past $k = 3$, and worse, would not give confidence in the future performance of this set, due to overfitting.

In this paper, we selected vantage sets of different sizes k using a *greedy sampling* algorithm, which works as follows. First, we chose at random a set S of $1,000$ pairs (A, B) between which we had measured latencies. We start by picking the best vantage point T_1 for the pairs in S among the 250 possible choices. Then we pick the best second vantage point T_2 that combined with T_1 would produce the lowest median relative error on S, and so on until k vantage points have been chosen. Using this algorithm to select k of n possible vantage points requires $O(nk)$ steps, as compared to n^k for the previous approach. Furthermore, because we evaluate only on a sample, we avoid overfitting vantage points to our test set.

Although it is clear that the "greedy" approach is scalable for arbitrary k, it remains to be seen whether it produces a good result, that is, whether it can pick vantage points that can accurately estimate network distances between any pair of nodes. Figure 4 shows the CDF for the

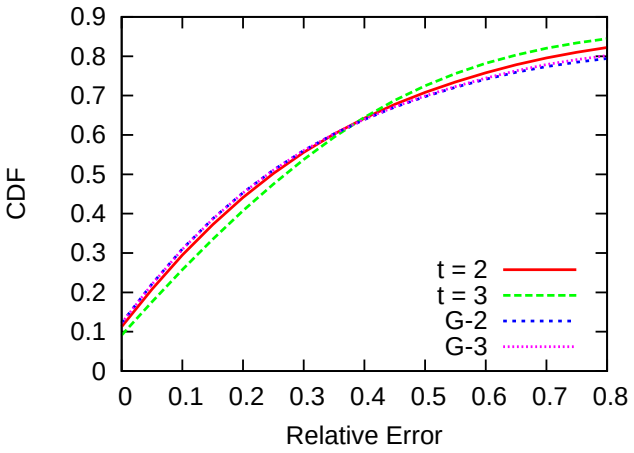

Figure 4. The CDF for the relative error for the estimations for the "best" ($t = 2, 3$) and "greedy" sets, with varying k.

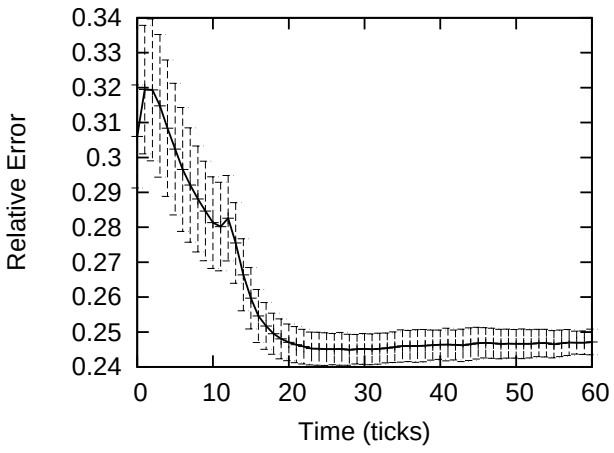

Figure 5. The Vivaldi simulation of 100 runs for our 250x250 dataset with error bars representing the standard deviation.

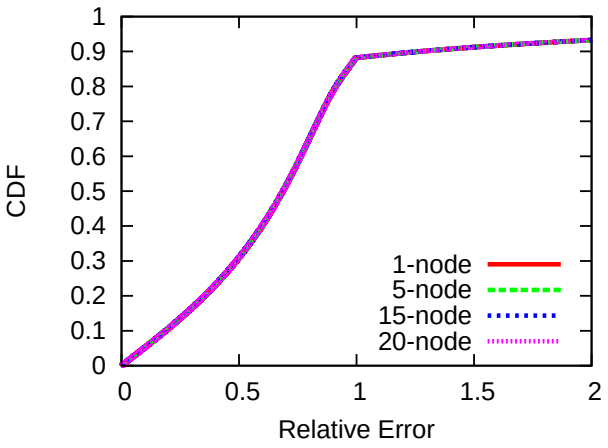

Figure 6. Trivial secure schemes: CDF of relative error for the "star topology" scheme

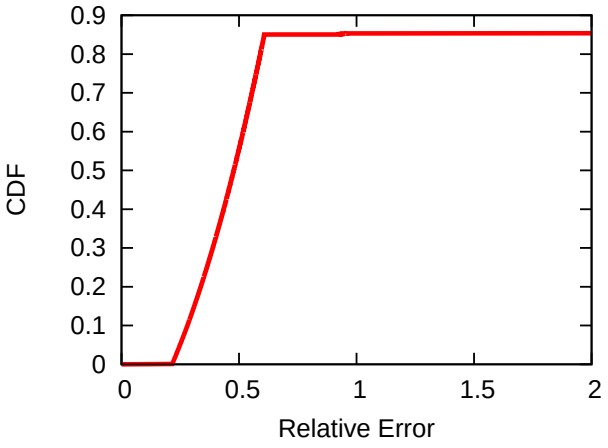

Figure 7. Trivial secure schemes: CDF of relative error for the "always predict median RTT" scheme.

relative error (amongst all pairs) using greedy sampling to select vantage sets of size $k \in \{1, 2, 3\}$. For comparison, the figure also includes the CDF of relative error for the k-node vantage sets that achieve the best median relative error on the iPlane data set from December 1, 2009. The labels $t = 2$ and $t = 3$ represent the optimal result, that is, picking the very best 2 and 3 vantage points respectively. The median relative error is similar for both algorithms for various k. The primary difference in accuracy can be seen at the $90th$ percentiles. The $90th$ percentile relative error for the optimal set for $k = 2$ is 1.7 compared to 2.3 for the greedy approach – a difference of 35.2%. The $90th$ percentile relative error for $k = 3$ for the optimal set is 1.45, compared to 2.15 for the greedy algorithm – a difference of 48.3%.

From Figure 4, it is clear that the greedy algorithm is nearly as accurate as the optimal algorithm. In the remainder of this section, we use results from the greedy sampling algorithm.

4.3. Baselines: Vivaldi, Star Topology, and Median

To establish a comparative baseline for the accuracy of Treeple, we evaluated three "basic" schemes on the same dataset used to evaluate the performance of Treeple.

Vivaldi [7] is a popular but insecure network coordinate system that is implemented in the Vuze file-sharing network [32] and used as the basis for computing coordinates in the various "secure" network coordinate systems [30, 35, 14]. We evaluate Vivaldi's performance on our dataset via simulation. Figure 5 shows the median relative error of all the nodes in our Vivaldi simulation over time. To obtain a non-sparse matrix of RTTs, we only used the 250 trusted nodes as source and destination, thus obtained a

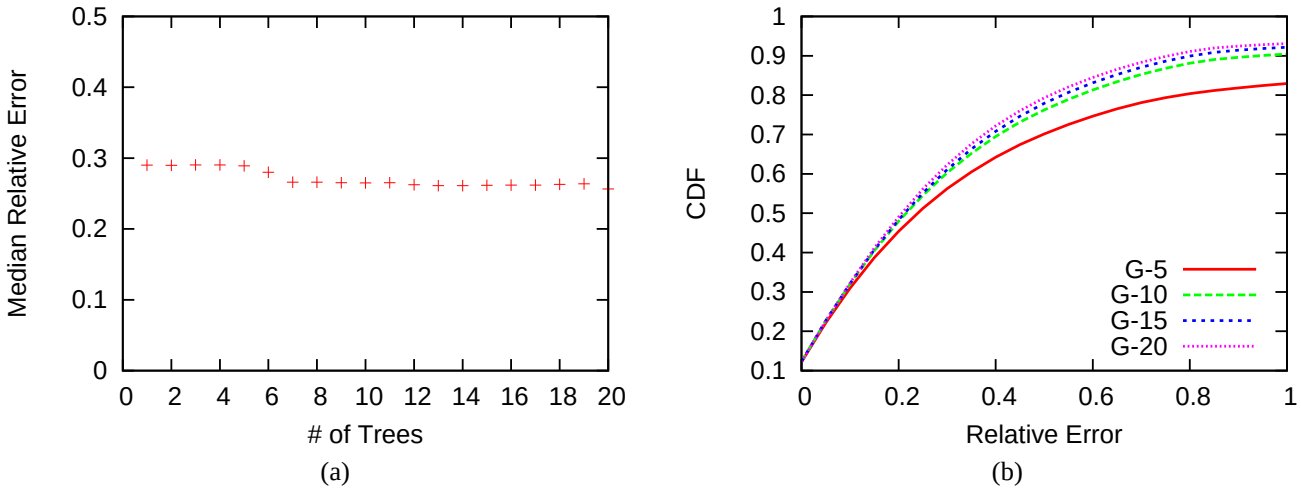

Figure 8. (a) The median relative error when varying k for the greedy approach, (b) The CDF for the relative error of the estimations for $k = 5, 10, 15, 20$ when using the greedy approach

matrix of 250×250 entries. The median relative error for Vivaldi was 25% at the end of the experiment. The figure shows the error bars representing the standard deviation and the mean over 100 runs.

As further point of comparison, we evaluated two simpler, provably secure schemes. In the first scheme, which we refer to as the "star topology," we assume a set of trusted vantage points (as in Treeple) but assign coordinates to peer A based solely on the (signed) distance rtt between the vantage point and A. When peer A wants to estimate its distance to peer B using vantage point C, it calculates $rtt(C, A) + rtt(C, B)$. The scheme extends to multiple vantage points in the same way as Treeple. The results are shown in Figure 6: for our data set, regardless of the number of vantage points used, the star topology yields a median relative error of 0.68. The second trivial scheme we evaluate is the "median" scheme, in which the predicted distance for every pair of distinct nodes is simply the median observed RTT for the data set. The scheme is trivially secure, but as shown in Figure 7, also provides poor accuracy, acheiving a median relative error of 0.5.

4.4. Accuracy

Figure 8(a) shows the median relative error for vantage sets of varying size k, computed as in the previous section. As k is increased, the median relative error decreases from 0.29 to 0.26, which is comparable to the median relative error obtained when using the Vivaldi network coordinate system. Figure 8(b) shows the CDF for the relative error of the estimations for varying number of vantage points. The different lines indicate the number of vantage points used. Using fewer than 5 vantage points produces the same accuracy. A gain in accuracy is obtained when $k > 6$. There is a significant difference in the $90th$ percentile between $k = 5$ and $k = 10$. Increasing $k > 10$ provides minimal gain in accuracy (either median relative error or $90th$ percentile relative error). Figure 8 shows that increasing the number of vantage points used for estimations also increases the accuracy of the system. From our experiments, having only 20 trusted nodes perform network measurements is enough to accurately estimate the network distance between two nodes.

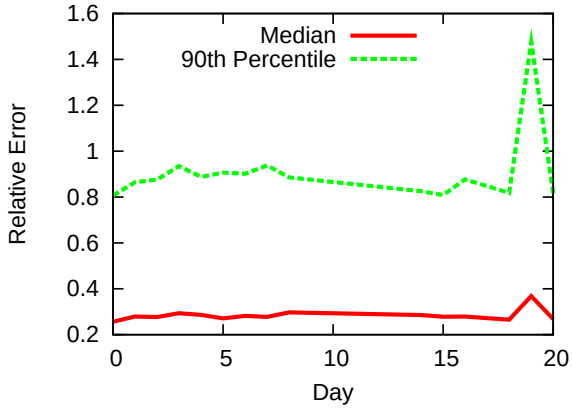

Figure 9. Median and 90th percentile relative error using 12/01/09 coordinates for estimation measurements through 12/21/09, by day.

4.5. Stability

If Treeple's accuracy depends on frequent updates to a node's position, then the vantage points may become a central point of failure, since they would need to be constantly available. Thus it is important to know how the ac-

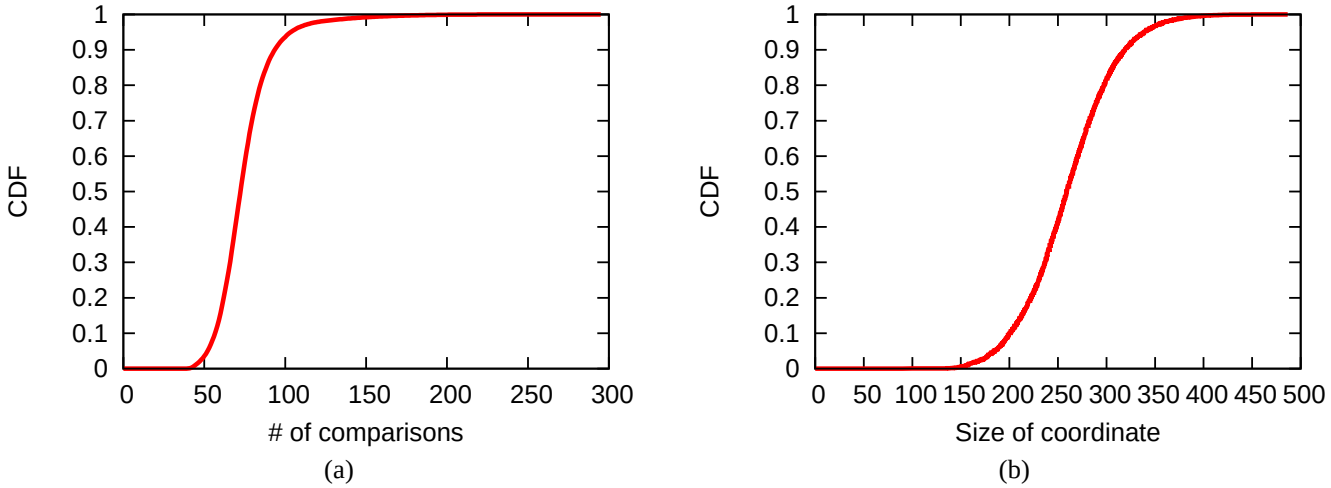

Figure 10. (a) The CDF for the number of comparisons required to estimate the network latency between two nodes. (b) The CDF for the total number of nodes in each assigned coordinate.

curacy of Treeple positions changes over time. We used the best 20 vantage points identified by the greedy method on 12/01/2009 to estimate the network distances for end hosts from 12/01/2009 to 12/21/2009 (three weeks). Figure 9 shows the median and $90th$ percentile relative errors when using the 12/01/2009 positions to estimate the network distances for other days. That figure shows that Treeple's accuracy remains nearly constant over time. Thus, a side effect of using Treeple is that frequent network measurements are not needed – the same positions can be used for long time periods τ. The spike in relative error on the $20th$ day is attributed to some possible faulty network measurements from the iPlane dataset.

4.6. Overhead

In order to show that Treeple imposes acceptable communication and computational costs, we measure these quantities across all nodes in our data set using the "greedy 20" set of vantage points. Figure 11 shows the Treeple positions of nodes X and Y. In this example, A is the vantage point, and B, C', and C'' are the routers on the Internet. The "size" of each position is the total number of routers along the 20 paths from each T_i to X or Y. Computing the distance under a given trusted node T_i is straightforward: we start at the first node of each Treeple position. They should be the same since it would be the trusted node. From that first node, we repeatedly advance to the next nodes until the two corresponding nodes in X and Y's positions are different. In the example shown, node C' is different from node C''. The estimated distance between nodes X and Y can

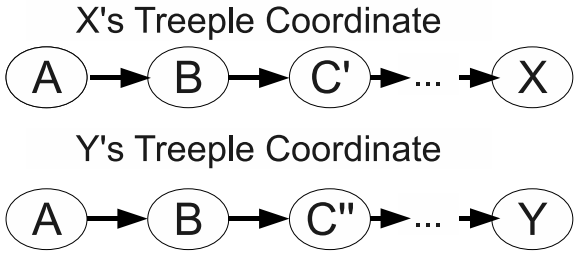

Figure 11. An example showing the positions of two end hosts X and Y

then be calculated as $(rtt_{A,X} - rtt_{A,B}) + (rtt_{A,Y} - rtt_{A,B})$. To compute the distance in this case, we had to compare three pairs of nodes.

Figure 10(a) shows the CDF for the number of comparisons required to compute a network latency estimate among all eligible pairs. The median number of comparisons is 73, with 20 coordinates, this means that each computation requires 3.65 comparisons on average. Figure 10(b) shows the CDF for the total number of nodes for the 20 coordinates of each end-host. The median total coordinate size for each end-host is 260. Each coordinate then contains $260/20 = 13$ hops. Each hop consists of an IP address (32 bits) and a RTT in ms (12 bits). Each hop size is then 44 bits. Each node's total coordinate size is then $44 \times 260 = 11,440$ bits or roughly 1.4 KB.

We note that there is a substantial opportunity for reduction of coordinate size using compression. In particular, it is not necessary to label the individual hops in a coordinate's path with globally unique names: as long as it is possible to determine the position along the path at which two coor-

dinates diverge, the distance can be computed. By examining the distribution on node degrees in the trees within our trusted vantage points, we found that on average each next hop identifier can be encoded with only 2 bits. Similarly, it is not necessary to encode the full RTT between the vantage point and each hop; using difference encoding on the RTTs, we found that on average each hop's RTT can be encoded with 6 bits. These techniques can reduce the size of a coordinate to 260 bytes.

Although this reduced coordinate size is still substantially larger than that of Vivaldi, Treeple requires only one single interaction to compute a peer's coordinate, which can then be used for weeks. On the other hand, Vivaldi requires the constant exchange of coordinates. Thus, over any reasonable period of time, Treeple's bandwidth overhead is smaller than that of Vivaldi.

It can be argued that the cost of the vantage points to perform traceroutes to other nodes on the Internet is very high. However, the vantage points only need to performed this measurement once every few weeks, as we showed in Section 4.5. We also note that iPlane is performing these measurements on a daily basis on the PlanetLab nodes.

4.7 Summary

On the large, real-world iPlane dataset, Treeple performs essentially as well as the Vivaldi network coordinate system. The median relative error when using 20 trusted nodes (note that GNP, a centralized network coordinate system, uses 20 trusted landmarks) is 0.26 for our system and 0.25 for Vivaldi. Thus, our system's latency estimation performs roughly as well as that of previous network coordinate system, while being the first system to provide provable security, meeting our goal of simultaneously providing security and accuracy. As expected, using more vantage points improves accuracy, at the cost of relying on more trusted nodes. The communication and processing overheads for Treeple are also relatively small. Finally, Treeple positions can be used to accurately predict network distances over long periods of time, including the full 21-day period of data used in our evaluation.

5. Other Related Work

We note that while there have been other systems published [9, 34, 4, 27] that do not use network coordinates and find close nodes, none of these systems can estimate the distance between arbitrary nodes. Several other projects have produced services that allow one peer to estimate its latency to another. IDMaps [9] was one of the first systems to estimate the network latency between two hosts. In IDMaps, hosts perform measurements to the central nodes called *tracers*, and the hosts can then approximate their latency to other hosts without having to contact those hosts directly. IDMaps cannot be used by a third party to predict the latency between pairs of hosts.

Sequoia [26] approximately represents Internet latencies and bandwidth as a tree metric. The authors showed that, for a small data set, using this metric allows accurate prediction of latency as well as bandwidth, with latency prediction comparable to Vivaldi. However, Sequioa relies on a complete view of the reconstructed tree metric; every node must know the complete tree, and in order to be assigned a position in the tree a node must participate in the protocol.

iPlane [20] and iPlane Nano [21] attempt to map the whole Internet to produce an atlas of link-to-link latency, bandwidth, and loss rate. In iPlane, a central database server uses the atlas to respond to path queries from Peers. In iPlane Nano, the atlas is compressed to about 7MB and distributed to end hosts; end hosts then download about 1MB of deltas produced each day. Hosts use this atlas to compute predicted latencies using shortest-path algorithms. Thus the computational and bandwidth overhead associated with using iPlane Nano are quite high.

We note that all of these systems essentially construct a "global map" of the peer topology by relying on the peers to report pairwise latency and traceroute measurements. None of these systems considers security in any way; essentially the measurements taken by all peers when constructing the maps are trusted. In contrast, Treeple takes security as a starting point and produces "local" positions for each peer, while trusting only a small set of end-hosts.

6. Discussion & Conclusion

Traceroute Complications

It is well-known that several issues such as multiple interfaces, load balancers, multi-protocol label switching, and non-responding hosts can interfere with the accuracy of paths returned by traceroute; in our evaluation, we did not use any special techniques to deal with these issues. We note that none of these issues presents a *security* challenge for Treeple, but that resolving them could well improve the accuracy of Treeple: for example, consistently resolving aliases would place the common ancestor later in a path and thus decrease the estimated distance between two nodes. Thus the performance evaluation in Section 4 can be seen as conservative in this sense. We naturally expect such measures to improve the accuracy of Treeple.

Vantage Point Migration

Although we treat vantage points as fixed in our theoretical treatment of Treeple, we point out that in practice, the

set of vantage points $\{T_1, \ldots, T_k\}$ is amenable to standard mechanisms used to manage migration of trusted servers for services such as DNS, DHTs, and Tor: it is straightforward to implement the Distance function so that it is robust to additional or missing vantage points – identify vantage points by verification key, and compute distances using only the positions computed by vantage points in common – and executables can eventually phase out support for discarded vantage points when sufficiently old versions are not supported, while peers running older executables will continue to function with slightly reduced accuracy.

Router Compromise

As we have discussed in section 2, under the current Internet architecture compromise of even a single BGP router is sufficient to render meaningless the measurements or estimates of any latency estimation scheme. Thus: *on the current Internet, network coordinate systems (and schemes based on them [1, 11, 15, 18, 4, 2, 6]) are insecure against compromised routers.* However, since router compromises are possible in practice, it would be desirable to consider whether, in some future network that secures path discovery and does not use policy-based routing, it is possible to reason about the security Treeple (or any other latency estimation scheme) can offer against router compromise.

To be concrete, we imagine a network (not the current Internet) in which a corrupted router can only intercept or delay traffic between a small fraction of the pairs of peers. In this case, it might be possible to use the trusted vantage points in Treeple to build a simple reputation system for router edges: if a particular edge exhibits variable behavior, as measured by a vantage point, then paths containing that edge could be marked as untrustworthy. As long as some paths remain, the influence of these adversarial edges would be limited. Similarly, individual nodes can maintain a "local" reputation which calculates whether a given path consistently leads to poor performance and if so, drop the path from their coordinates (replacing it with $\perp$). Of course, the exact nature of the security guarantees these mechanisms can provide would clearly depend on the nature of the routing protocol in this hypothesized future network architecture.

Conclusion

We propose Treeple, a provably secure network distance estimation service, where the estimated network distance differs from the real network distance by 26%. The accuracy of the system is comparable to using a network coordinate system, where the median relative error was 25%. In addition, Treeple is provably secure, whereas all previously proposed schemes are vulnerable to attacks within the Treeple threat model. Moreover, because Treeple positions are extremely stable, they can be maintained with very low overhead compared to traditional network coordinate schemes.

Acknowledgments

We thank our Shepherd, Suman Banerjee, for his input on improvements to this paper. We also thank Yongdae Kim, Zhi-Li Zhang, Roger Dingledine, and Brighten Godfrey for helpful comments and discussions about Treeple. This work was supported by the National Science Foundation under grant CNS-0716025.

References

[1] Ittai Abraham and Dahlia Malkhi. Compact routing on euclidian metrics. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 141–149, New York, NY, USA, 2004. ACM.

[2] R. Bazzi and G. Konjevod. On the Establishment of Distinct Identities in Overlay Networks. *In ACM PODC*, 2005.

[3] Eric Chan-Tin, Daniel Feldman, Yongdae Kim, and Nicholas Hopper. The Frog-Boiling Attack: Limitations of Anomaly Detection for Secure Network Coordinates. *SecureComm*, 2009.

[4] D. Choffnes and F. Bustamante. Taming the Torrent: A practical approach to reducing cross-ISP traffic in P2P systems. *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2008.

[5] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2004.

[6] James Cowling, Dan Ports, Barbara Liskov, Raluca Ada Popa, and Abhijeet Gaikwad. Census: Location-Aware Membership Management for Large-Scale Distributed Systems. *In the proceedings of USENIX Annual Technical Conference*, 2009.

[7] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proceedings of ACM SIGCOMM*, 2004.

[8] John R. Douceur. The sybil attack. In *Proc. of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[9] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Trans. Netw.*, 9(5):525–540, 2001.

[10] Sharon Goldberg, Michael Schapira, Peter Hummon, and Jennifer Rexford. How secure are secure interdomain routing protocols? In *SIGCOMM'10: Proceedings of the ACM SIGCOMM 2010 Conference on Data Communication*. ACM, 2010. to appear.

[11] R. Gummadi, R. Govindan, N. Kothari, B. Karp, Y. J. Kim, , and S. Shenker. Reduced state routing in the internet. *HotNets*, 2004.

[12] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous. Real attacks on virtual networks: Vivaldi out of tune. *Proceedings of the SIGCOMM workshop on Large-scale Attack Defense*, 2006.

[13] M.A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous. Virtual Networks under Attack: Disrupting Internet Coordinate Systems. *ACM/e-NEXT International Conference on Future Networking Technologies (CoNext)*, 2006.

[14] Mohamed Ali Kaafar, Laurent Mathy, Chadi Barakat, Kave Salamatian, Thierry Turletti, and Walid Dabbous. Securing Internet Coordinate Embedding Systems. *Proceedings of ACM SIGCOMM*, 2007.

[15] Jonathan Ledlie, Michael Mitzenmacher, and Margo Seltzer. Wired geometric routing. *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

[16] Jonathan Ledlie, Peter Pietzuch, and Margo Seltzer. Stable and accurate network coordinates. *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[17] Jonathan Ledlie, Peter Pietzuch, and Margo Seltzer. Network coordinates in the wild. *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[18] C. Lumezanu, D. Levin, and N. Spring. Peer wise discovery and negotiation of faster path. *In Proceedings of HotNets-VI*, 2007.

[19] Cristian Lumezanu, Randy Baden, Neil Spring, and Bobby Bhattacharjee. Triangle inequality variations in the internet. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 177–183, New York, NY, USA, 2009. ACM.

[20] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An Information Plane for Distributed Services. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.

[21] Harsha V. Madhyastha, Ethan Katz-Bassett, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iplane nano: path prediction for peer-to-peer applications. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 137–152, Berkeley, CA, USA, 2009. USENIX Association.

[22] T. S. Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. *Proceedings of IEEE INFOCOM*, 2002.

[23] T. S. Eugene Ng and Hui Zhang. A network positioning system for the internet. *Proceedings of the USENIX annual technical conference*, 2004.

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

[25] PlanetLab. `http://planet-lab.org`.

[26] Venugopalan Ramasubramanian, Dahlia Malkhi, Fabian Kuhn, Mahesh Balakrishnan, Archit Gupta, and Aditya Akella. On the treeness of internet latency and bandwidth. In *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 61–72, New York, NY, USA, 2009. ACM.

[27] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Topologically-Aware Overlay Construction and Server Selection. In *Proceedings of Infocom*, 2002.

[28] Damien Saucez, Benoit Donnet, and Olivier Bonaventure. A Reputation-Based Approach for Securing Vivaldi Embedding System. In *EUNICE Open European Summer School and IFIP TC6.6 Workshop on Dependable and Adaptable Networks and Service*, 2007.

[29] Y. Shavitt and T. Tankel. Big-Bang Simulation for embedding network distances in Euclidean space. *IEEE INFOCOM*, 2003.

[30] Micah Sherr, Matt Blaze, and Boon Thau Loo. Veracity: Practical Secure Network Coordinates via Vote-based Agreements. In *USENIX Annual Technical Conference*, 2009.

[31] GNP Simulator. http://www.cs.rice.edu/~gw4314/ncs-configurable.tar.gz.

[32] Vuze. http://azureus.sourceforge.net.

[33] Guohui Wang and T. S. Eugene Ng. Distributed Algorithms for Stable and Secure Network Coordinates. *ACM/USENIX Internet Measurement Conference (IMC)*, 2008.

[34] B. Wong, A. Slivkins, and E. Gün Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2005.

[35] David Zage and Cristina Nita-Rotaru. On the accuracy of decentralized virtual coordinate systems in adversarial networks. In *Proceedings of the 14th ACM conference on Computer and communications security (CCS)*, 2007.

[36] Han Zheng, Eng Keong Lua, Marcelo Pias, and Timothy Griffin. Internet Routing Policies and Round-trip Times. *Passive and Active Measurement Workshop*, 2005.