On the Treeness of Internet Latency and Bandwidth

Venugopalan Ramasubramanian[*] Dahlia Malkhi[†] Fabian Kuhn[‡] Mahesh Balakrishnan[§]
Archit Gupta[♯] and Aditya Akella[♭]

ABSTRACT

Existing empirical studies of Internet structure and path properties indicate that the Internet is tree-like. This work quantifies the degree to which at least two important Internet measures—latency and bandwidth—approximate tree metrics. We evaluate our ability to model end-to-end measures using tree embeddings by actually building tree representations. In addition to being simple and intuitive models, these trees provide a range of commonly-required functionality beyond serving as an analytical tool.

The contributions of our study are twofold. First, we investigate the ability to portray the inherent hierarchical structure of the Internet using the most pure and compact topology, trees. Second, we evaluate the ability of our compact representation to facilitate many natural tasks, such as the selection of servers with short latency or high bandwidth from a client. Experiments show that these tasks can be done with high degree of success and modest overhead.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network topology, Internet*

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

Keywords

Sequoia, Internet topology, tree embedding, latency, bandwidth

1. INTRODUCTION

Understanding the fundamental structure and properties of large and complex networks such as the Internet can be a daunting task. The internal structure of the Internet is intentionally hidden from its users, providing full, seamless end-to-end connectivity. Furthermore, internal routing decisions are affected by complex and dynamic policies and rules that are hard to predict and model.

[*]Venugopalan Ramasubramanian: Microsoft Research Silicon Valley, Mountain View, CA 94043 rama@microsoft.com

[†]Dahlia Malkhi: Microsoft Research Silicon Valley, Mountain View, CA 94043 dalia@microsoft.com

[‡]Fabian Kuhn: MIT CSAIL, Cambridge, MA 02139 fkuhn@csail.mit.edu

[§]Mahesh Balakrishnan: Microsoft Research Silicon Valley, Mountain View, CA 94043 maheshba@microsoft.com

[♯]Archit Gupta: Data Domain Inc., Santa Clara, CA 95054 agupta@datadomain.com

[♭]Aditya Akella: University of Wisconsin-Madison, Madison, WI 53706 akella@cs.wisc.edu

SIGMETRICS/Performance'09, June 15–19, 2009, Seattle, WA, USA.

Previous work has looked at various features of the Internet graph, and proposed theoretical models to describe its evolvement. Faloutsos et al. [10] discovered that Internet router connectivity obeys a scale-free distribution governed by a simple power law. Barabási and Albert [4] then developed an evolutionary model of preferential attachment, which can be used for generating topologies with power-law degree distributions.

A fundamental property that stems from these works is the inherent tree-like structure of the Internet. Indeed, it was observed before that the core of the Internet is built essentially in layers [38, 40]. At the core, we find a densely connected set of high degree routers. Since these routers are powerful computers interconnected with high-bandwidth links, at a coarse grain, they look like a single root node. Dangling from the core are other routers whose interconnectivity is fairly sparse. At the endpoints, home users and businesses connect locally to ISP gateways. Consequently, the network is inherently hierarchical in structure. These trends are even more pronounced when considering the metric of available bandwidths. For any graph structure, it is well known that the metric of bottleneck-edge along shortest paths forms a hierarchically separated tree.

That said, it should be noted that the Internet is in no way a pure tree. Direct peering interconnects are common between ASs, and routes often do not optimize end-to-end latency or bandwidth. This often results in triangle inequality violations [44], and a fortiori, treeness violations. Nevertheless, our study ascertains that a skeleton tree prevails and dominates end-to-end measures to a certain degree of accuracy.

In addition to the above observations, we can rigorously quantify the treeness of Internet measures such as latency and bandwidth. We empirically study end-to-end measures from three datasets, using a rigorous measure of treeness we introduced in [1]. The measure, called *4PC*-ϵ, gives a value in the range $[0..1]$, where at one end (0) is a perfect tree metric, and at the other (1) an arbitrary metric space. Our evaluation of these real-life measurements reveal encouraging closeness to tree metrics. More than 80% of node quadruplets we test exhibit an *4PC*-ϵ value of $.4$ or less. For a sanity comparison, note that a random graph, a Euclidean space, and a scattering of nodes on a sphere all have more than 30% quadruplets with *4PC*-ϵ of $.5$ or higher.

Our interest goes much further than merely describing this phenomenon. Luckily, there is a strong provable relationship between the *4PC*-ϵ measure of a metric and the ability to embed it in a tree [1]. By embedding, we refer to the mapping of nodes onto leaves of a virtual tree, whose inner nodes are virtual points and whose edge weights are carefully selected so as to represent original graph measurements. We take advantage of these results and produce in this work methods for approximate tree embedding. Note that we do not attempt to discover actual gateways and routers, but rather strive to build a "virtual" model through light-weight, end-to-end mechanisms.

Our construction techniques are an important part of our contribution. They are rooted in a perfect tree-metric procedure, which starts with a full set of measurements of pairs of endpoints from a tree metric space and constructs a tree that perfectly represents

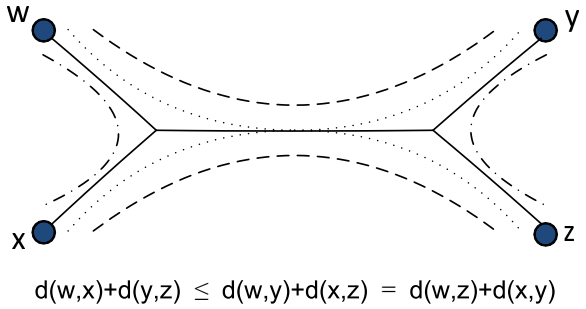


Figure 1: Four-Points Condition: For any four nodes and the three sums of distinct pairs of distances between them, the highest sum is equal to the second highest iff the nodes and distances are embeddable in a tree.

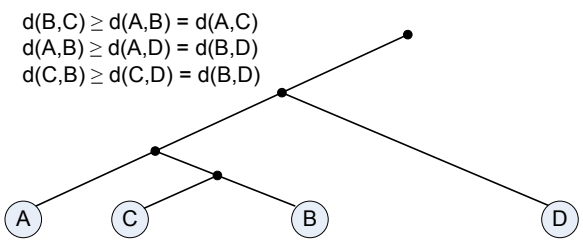


Figure 2: Tree Showing an Ultra-Metric: Hosts B and C with a common ancestor to A have the same distance to A. Similarly for A and B with respect to D.

them [5]. Our tree construction protocols include several substantial modifications to that procedure. First, we work with non-pure tree metrics, and accordingly, develop strategies to reduce the (inevitable) distortion error of the embedding. Second, we heuristically reduce the number of end-to-end measurements required for the construction in order to accommodate partial measurements and reduce measurement overhead. Third, we employ some tree balancing rules in order to enhance the efficiency of services that are implemented on our trees, e.g., closest node selection and distance labeling. Fourth, we found it to be highly beneficial to maintain a small number of trees, tuned in different ways, and pick the best one as needed. These techniques adapt fundamental techniques from graph theory for practical deployment.

The outcome of the above is an elegant and unified method for representing both latency and bandwidth as trees. We built a system called Sequoia that takes as input a set of pairwise end-to-end network measurements, and quickly embeds them into a collection of trees. Sequoia trees provide a strikingly intuitive topological decomposition of the Internet, and facilitate a host of services including distance approximation, closest neighbor selection, and topologically-aware clustering, which can be readily implemented through easily-decentralized, light-weight mechanisms.

This paper presents a comprehensive evaluation of Sequoia over three datasets. It demonstrates that Sequoia is resilient to violation of the triangle inequality condition in network measures, tolerates non-availability of some measurements, and shows good scalability with increasing number of hosts.

The rest of the paper has the following organization: Section 2 provides some background about tree metrics and makes a case for embedding network measures on trees. Section 3 then describes the Sequoia system in detail while Section 4 evaluates Sequoia. Finally, we discuss suitable applications and related work in Sections 5 and 6 and conclude in Section 7.

2. BACKGROUND AND INTUITION

The key intuition behind this work is that Internet path measures such as bandwidth and latency are approximate tree metrics. In this section, we provide some background about tree metrics and present intuitive and analytical arguments to back up this intuition.

2.1 Tree metrics

Consider a set D of pair-wise measurements of some network path measure, say latency or bandwidth, between a set V of networked hosts. This set of measures D is a *tree metric* if there exists a tree T with non-negative weights such that $V \subseteq T$ and $d_V(u,v) = d_T(u,v)$ for all $u, v \in V$, where $d(u,v)$ and $d_T(u,v)$ represent the pair-wise path property. In other words, a set of measures is a tree metric if it can be derived from distances on a tree, that is, it can be embedded on a tree. Note that in the above definition, the tree T may have additional nodes not present in the set V.

There is a convenient condition called the *Four Points Condition* (4PC) to verify whether a set of measures is a tree metric. The four-points condition states that for any four hosts w, x, y, and z ordered such that $d(w,x) + d(y,z) \leq d(w,y) + d(x,z) \leq d(w,z) + d(x,y)$, $d(w,y) + d(x,z) = d(w,z) + d(x,y)$. A set of measures is a tree metric if and only if every set of four hosts satisfies the 4PC. Figure 1 illustrates the four points condition graphically.

Another form of tree metric that is useful for modeling network path measures is the *ultra metric*. An ultra metric embeds network hosts into a hierarchically separated tree, where distance between a pair of nodes depends on the closest common ancestor of the nodes. That is, all pairs of hosts with the same closest common ancestor have the same distance. Note that, once again, the tree might include additional hosts not present in the set of measures. An ultra metric is a stricter form of tree metric in that every ultra metric is a tree metric while the converse is not always true.

Similar to tree metrics, there is a convenient condition to verify whether a set of measures is an ultra metric. This condition, called the *Three Points Condition* (3PC), states that for any three hosts x, y, and z ordered such that $d(y,z) \leq d(x,z) \leq d(x,y)$, $d(x,y) = d(x,z)$. A set of measures is an ultra metric if and only if every set of three hosts satisfies the 3PC. Figure 2 illustrates an ultra metric and the three points condition graphically.

Note that the above definition of the three point condition applies to measures such as latency where smaller values are more desirable than larger values. The corresponding 3PC for bandwidth would state the converse, that is, in any triplet, the two smaller bandwidths are equal.

2.2 Bandwidth and latency as tree metrics

We first argue why tree metric is a fitting representation for bandwidth. We do this by presenting two hypothetical network models in which bandwidth measures turn out to be exact tree metrics.

Best-bandwidth networks: Consider a network of hosts connected by routers and gateways where the path used to route packets between two hosts is the best-bandwidth path, that is, the path with the highest bottleneck bandwidth, where the bottleneck bandwidth of a path is the minimum of the bandwidths of each link in the path.

We can trivially show that the set of bandwidths between the end hosts in the above network is an ultra metric. Assume, for

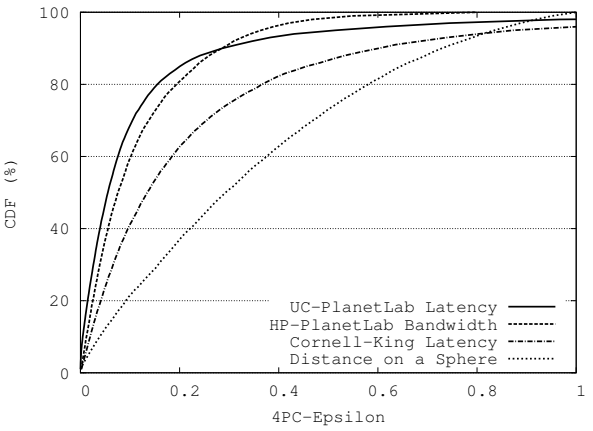


Figure 3: CDF of 4PC-ϵs: The 4PC-ϵs are mostly small indicating that the datasets closely resemble tree metrics, whereas latencies between nodes distributed on the surface of a sphere have much bigger 4PC-ϵs.

instance, that there is a set of hosts, x, y, and z with $d(y,z) \geq d(x,z) \geq d(x,y)$ that violate the 3PC for bandwidth; that is, $d(x,z) \neq d(x,y)$. Then the path $x \rightarrow z \rightarrow y$ would have a higher bandwidth than the current path, indicating that the network is not using the best-bandwidth paths—a contradiction.

Edge-bandwidth networks: Consider a network of hosts connected by routers and gateways where the last-mile access links have lower bandwidths than the links at the inner core. That is, the bottleneck bandwidth between two end hosts only depends on the bandwidths of the access links connecting the two end hosts to the network and is totally independent of what routing policies are used for finding the paths. Then, this is a special case of best-bandwidth networks (all routes from A to B have the best bottleneck bandwidth since the bottleneck occurs at the access link), and 3PC holds a fortiori here.

The Internet, of course, does not always satisfy the above models. In fact, the first scenario of best-bandwidth networks may arise only in rare instances: for example, when a CDN such as Akamai uses the best-provisioned paths in an overlay network[1]. The second scenario, where the path bandwidth depends on the last-mile, access links, is more common. Hu et al. [16] report that, in the Wide-Area Internet, 60% of paths between random end hosts have the bottleneck in the first or second hop. This property is even more prevalent in broadband networks as shown by a recent measurement study by Dischinger et al. [9].

If these simple models hold globally we could model bandwidth in trivial ways—for example, by just keeping track of the bandwidth of the access link. In practice, however, networks only satisfy them *approximately*, motivating the need for more sophisticated models explored in this paper.

A different intuition is required to understand the treeness of end-to-end latency, which depends on all the components in the path. Here, it is useful to look at the graph structure of the network. The hierarchical organization of the Internet, with different tiers of ISPs (Tier 1, Tier 2, etc.), is well known and has been empirically verified by several studies [38, 40]. These studies show that routers at the core are densely connected—as Tier 1 ISPs have many peering relationships with each other. Whereas, routers in the edge networks can be picturized as dangling away from the core since ISPs in other tiers mostly have hierarchical, provider-consumer relationships between them. The result is a jelly-fish-like structure [40], where most Internet routes start at the edge, go up the hierarchy, traverse the core, and descend down, leading to a phenomenon known as *valley-free routing* [38].

In practice, the evident hierarchical organization of Internet routers does not translate to a strict tree since the peering relationships between ISPs are quite complicated; the core has many short cuts, peer-to-peer relationships exist even outside the core, and so on. Further, a variety of sub-optimal routing policies contributes to the violation of symmetry and the triangle inequality condition, requisite properties of a tree (or any) metric.

Hence, we performed an empirical study to quantify how close Internet measures actually are to tree metrics, which we present next.

2.3 Analysis of the tree-ness of the Internet

We quantify the fit of a set of measures to a tree metric by measuring how well it satisfies the four points condition. We use a parameter called the *4PC-ϵ* originally introduced in [1] to quantify the deviations from 4PC. The 4PC-ϵ for a set of four nodes w, x, y, and z ordered such that $d(w,x) + d(y,z) \leq d(w,y) + d(x,z) \leq d(w,z) + d(x,y)$ is the one that satisfies the equation $d(w,z)+d(x,y) = d(w,y)+d(x,z)+2\epsilon\cdot\min\{d(w,x), d(y,z)\}$. The paper [1] provides more intuition about this definition.

The distribution of 4PC-ϵs show how close a network measure is to a tree metric. The 4PC-ϵs are zero for a perfect tree metric and at most one for an arbitrary metric. When the distances violate the triangle inequality condition, the 4PC-ϵ can even be larger than one.

We compute the distributions of 4PC-ϵs for three realworld datasets: 1) a *UC-PlanetLab Latency* dataset of round-trip times between PlanetLab nodes measured at University of Cincinnati [53], 2) a *Cornell-King Latency* dataset of latencies measured between DNS servers using the King technique [14] at Cornell University [47], and 3) a *HP-PlanetLab Bandwidth* dataset of available bandwidth measurements between PlanetLab nodes collected at HP Labs [51] using the pathChirp tool [33]. Table 1 summarizes the details about these datasets.

Figure 3 shows the CDF 4PC-ϵs for each dataset. The 4PC-ϵ values are sometime greater than one because the datasets have many violations of the triangle inequality condition, roughly 15 to 40 % of the triplets as shown in Table 1. We restrict the x-axis between zero and one for clarity. Overall, the 4PC-ϵs are small, over 80% of values are less than 0.2 for UC-Planetlab and HP-PlanetLan datasets and less than 0.4 for the Cornell-King dataset. The tail, however, is quite heavy, especially for the Cornell-King dataset. We found that the larger 4PC-ϵ values mostly come from triangle inequality violations.

To put these numbers in perspective, we also show the distribution of 4PC-ϵs for a metric of latencies on the surface of the earth (approximated as a sphere), as would be the case if the latencies between Internet hosts were based on their geographical distance. This curve in Figure 3 was computed by sampling the shortest geographic distances between 1000 random points on the surface of a sphere. Not surprisingly, the distribution of 4PC-ϵs for geographic routing shows much less resemblance to a tree metric than latencies on the Internet.

2.4 Assumptions

Overall, the above intuition and analysis indicates that approximating Internet path measures as tree metrics is a promising ap-

[1]Akamai have their own proprietary detour routing service called SureRoute [46]. However, we are not aware if the routes are optimized for bandwidth.

proach. Certainly, this approach makes a few assumptions about the path measures. Similar to prior coordinates-based approaches [7, 8, 27] for modeling latency, this approach also assumes that the network measure is a metric. The metric assumption implies at least two properties: 1) the measures are symmetric and 2) the triangle inequality holds. However, these properties do not always hold in the Internet. Policy-based routing (such as hot potato and cold potato) that does not select shortest or bandwidth-optimal paths and transient overheads such as queuing delay lead to violations of symmetry and the triangle inequality condition.

Our approach, described in the next section, does not currently address asymmetric network measures. However, it is resilient to triangle inequality violations and works well in their presence as will be evident from our evaluation.

Finally, we are aware that the term *bandwidth* often refers to related yet different measures: *capacity*, the theoretical maximum bandwidth of a path in the absence of cross traffic, *available bandwidth*, the actual unused bandwidth at a given instant, and the *bulk-transfer throughput* achievable by a TCP connection [30]. The intuition provided above as well as our approach and techniques described below apply well to any of these bandwidth measures. The dataset we use to evaluate our approach, however, measured available bandwidth.

3. SEQUOIA

We next present Sequoia, a system that applies the above insights for embedding network measures such as latency and bandwidth into trees. Sequoia strives to minimize distortion of end-to-end measures despite the fact that Internet latency and bandwidth are not perfect tree metrics. Part of the effort is dedicated towards providing network-aware functionalities such as path quality estimation, server selection, and hierarchical clustering to applications. The rest of this section provides an overview of Sequoia, details its design, and describes how it supports the above functionalities.

3.1 Overview

Sequoia constructs "virtual" *prediction trees* for a system of networked hosts through end-to-end measurements. The prediction tree for a set of networked end hosts looks as follows: The end hosts form the leaf nodes and are connected via a network of virtual inner nodes. The links in this virtual topology have weights that model a network measure, that is, latency or bandwidth.

The inner nodes are virtual in the sense that they are introduced purely for modeling purposes and are not expected to have any real-world associations. The tree topology also does not imply that the end hosts organize into a tree-based distributed system; the virtual trees could merely be an abstraction in the memory of a few or all hosts in the system.

Sequoia designates an end host as a *lever* R for each virtual tree; the lever could be any of the end hosts or a specially provisioned, landmark server. The lever acts as a reference for the tree; that is, link weights are assigned such that the tree path from the lever to an end host has the same distance as the real Internet path between the lever and the end host. Other distances on the tree might distort original end-to-end measures; our construction aims to minimize such distortion. Figure 4 illustrates a virtual tree for three end hosts and a lever.

Besides the benefit of obtaining a compact representation of the given network, prediction trees provide a variety of useful functions.

Distance labels The prediction tree provides a convenient way to label participating hosts so that the distance between every pair is encoded in their labels. Each host has a *distance label* that encodes the path of the host to the lever and the corresponding link weights. Two hosts can then infer their tree distance (which in turn, estimates their true distance in the original metric). Note that this computation need not refer to other parts of the tree. For example, in Figure 4, hosts A and B have the distance label s,t and C the label t. Two nodes, say A and C, can estimate their distance by computing the path A,s,t,C from their distance labels.

Closest node discovery A participating end-host can use the prediction tree to discover a closest or best-provisioned peer node by merely looking at its immediate vicinity in the tree. For example, in Figure 4, host A can find its closest peer host B through a localized search. Alternatively, an external end-host can use the tree to guide its search to find the best server. Such a search can start at the lever and direct itself in the tree along the path from the lever to the chosen server at the leaf.

Topological decomposition The prediction tree itself provides a topologically-aware hierarchical clustering of hosts. For example, in Figure 4, hosts A and B being closer to each other than host C form a lower-level cluster.

3.2 Design

Given a full set of measurements of pairs of endpoints from a tree metric space, the mathematical procedure for reconstructing a tree that perfectly represents them is well understood [5]. With that procedure, if we start with a tree metric, we get a zero-distortion prediction tree. Sequoia's algorithms for constructing prediction trees are based on components of this perfect reconstruction, but differ in substantial ways. We first explain the principles of a perfect tree metric embedding.

The algorithm starts with a fixed lever R and adds nodes to form a tree one by one. A host's position in the tree is determined by the lever and one other node called the *anchor*. A host, say B, tries to maintain exact distances to the lever R and its anchor A; it preserves the three distances $d(A, R)$, $d(B, R)$, and $d(A, B)$, by introducing a virtual node s in the existing tree path between A and R. Note that computing the distances from s to R, A, and B while preserving the real distances is straight forward (for instance, $d(s, R) = 0.5 \cdot (d(A, R) + d(B, R) - d(A, B))$).

In order to preserve zero-distortion for tree metrics, the anchor needs to satisfy the following condition: maximize the distance d(s,R) on the tree [5]. This distance $d(s, R) = 0.5 \cdot (d(A, R) + d(B, R) - d(A, B))$ is called the *Gromov product* and will be denoted as $(A|B)_R$ in the rest of the paper. This basic join algorithm is illustrated in Figure 6.

While this algorithm works perfectly for a pure tree metric, for approximate tree metrics this algorithm has the following problem. Every time a node B is added to the tree, the anchor point s is positioned so as to preserve the joiner's distances to the lever and to its anchor. However, distances to all other nodes may be distorted. That distortion may be further intensified for a new node C that uses B as its anchor. Generally, distortion between a pair of nodes will depend on the number of anchor points along the path between them. In order to guarantee lowest worst-case distortion, one needs to form a particular joining order of the nodes so as to carefully choose the anchor points [1]. The theoretical bounds given in [1] were a good starting point for our algorithm, on which we further refine in a variety of ways described below.

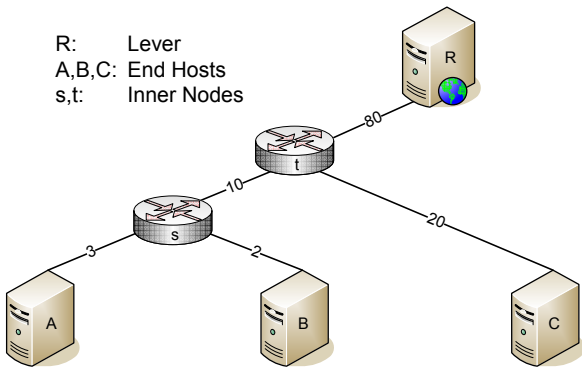


Figure 4: Prediction Tree: An example prediction tree between three end hosts and a lever. The link weights model path qualities such as latency.

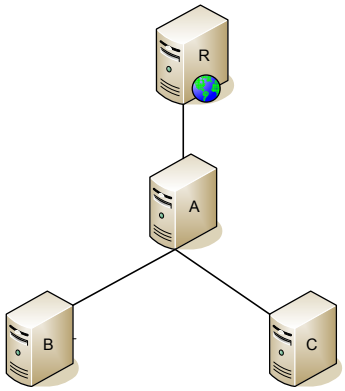


Figure 5: Anchor Tree: An example anchor tree corresponding to the prediction tree in Figure 4.

3.2.1 Anchor tree

Even though the above algorithm provides an elegant construction for prediction trees, it does not help in shaping the structure of the tree. Consequently, the resulting prediction tree may not have a nice balanced structure. For instance, if we start with distances on a linear chain of hosts, then the resulting prediction tree will also be a chain. This has profound impact on the scalability of our approach because the cost of basic primitives such as distance labels and server selection depends on the tree structure.

We propose an alternative, scalable abstraction called the *anchor tree*. An anchor tree is simply a tree showing anchor relationships of end hosts in the prediction tree. Figure 5 shows an example anchor tree for the prediction tree in Figure 4. Unlike the prediction tree, an anchor tree provides more opportunities to shape its structure. When choosing anchor points some priority could be given to ensure that the resulting anchor tree will have a reasonably balanced shape. Even in the adverse case where the distances are induced by a linear chain, we can construct a well-balanced anchor tree.

The anchor tree and its corresponding prediction tree are equivalent—the former is just a more scalable representation of the latter. It provides the exact same distance estimates as the prediction tree although computing distance is a bit more tedious. Just as in a prediction tree, the path from an end host to the lever serves as the distance label in an anchor tree. Two end hosts can estimate the distance between them based on their distance labels. A simple distance computation algorithm is to construct a prediction tree (in memory) based on the anchor relationships encoded in the two distance labels and then use the prediction tree to estimate the distance.

A reasonably balanced anchor tree provides the following advantages: First, it provides short distance labels that scales logarithmically with system size. Second, the paths from the lever to hosts are also short enabling efficient searches for best servers. Finally, unlike the prediction tree, an anchor tree is composed only of real hosts and can be directly used to construct a distributed system.

3.2.2 Best host discovery

The anchor tree representation also helps to reduce measurement overhead during the construction of prediction trees. Note that, the tree construction algorithm outlined earlier involved the key step of finding an anchor that maximizes the Gromov product. This step might involve searching through all hosts in the system and measuring their distance to the joining host, an $O(N)$ probing effort.

Sequoia instead uses the anchor tree to prune this search space and find a suitable anchor with fewer measurements. We first outline a general search algorithm on the anchor tree for finding a host that meets a general search criterion with respect to a target host B. The search starts with the lever as the *candidate* host. It looks at all the hosts within a certain *search_depth* D from the lever and measures their distance to the target host. The best host from this set is chosen as the new candidate and the search is repeated until no progress can be made. Finally, the search returns the best host in the path from the lever to the last candidate. This search algorithm is illustrated in Figure 7.

The search criterion in the above algorithm can be set according to requirements. A simple criterion based on smallest latency or largest bandwidth enables server selection. Tree construction requires the more complex criterion of maximum Gromov product. This criterion can reduced as maximum $d(A,R) - d(A,B)$, where B is the joining node because the third term in the Gromov product, $d(B,R)$ is a constant during the anchor search.

The above search algorithm is a heuristic and is not guaranteed to find the correct anchor or the closest or the best-provisioned server. It is a practical trade-off to keep measurement cost low albeit, as shown in Section 4, an effective one. The number of measurements required by the search algorithm is also not bounded although we expect it to scale logarithmically on average if we bound the number of hosts a node can anchor.

3.3 Practical issues

In this section, we discuss how Sequoia handles other major practical issues.

3.3.1 Bandwidth representation

The previous discussion primarily talked about building prediction and anchor trees for a distance (latency) measure. There are two choices for representing bandwidth on a prediction tree. The first is to use prediction trees as a black-box for modeling any approximate tree metric and represent bandwidth by treating it as a distance metric. The second is to build a bandwidth-specific prediction tree where the prediction tree will resemble an ultrametric tree, similar to Figure 2. One well-known way to build a bandwidth-specific prediction tree is to build a maximum- weighted spanning tree (MST) between the hosts using bandwidths as link weights and then define the distance between two hosts in the MST as the weight of the smallest weighted-link in the path [5].

Figure 6: ConstructTree Algorithm: Basic prediction tree construction algorithm that preserves tree metrics for a set V of hosts.

```
1: return tree T := constructTree(V \ {R}, R);
2:
3: function ConstructTree(V,R):
4:    if |V| > 1 then
5:       choose next host B ∈ V
6:       choose anchor A that maximizes (A|B)_R
7:       add virtual node s between A and R at distance (A|B)_R
         from R;
8:       add node B at distance (A|R)_B from s
9:       T := constructTree(V \ {B}, R);
10:   fi;
11:   return T;
12: end ConstructTree
```

Figure 7: SearchTree Algorithm: Basic search algorithm to find a candidate host meeting some search criterion with respect to a target host B using the anchor tree.

```
1: return candidate C := SearchTree(R, B);
2:
3: function SearchTree(C, B):
4:    S := all hosts at depth D from C in the anchor tree;
5:    C' := best host S that meets the search criterion;
6:    if C ≠ C' then
7:       C := SearchTree(C', B);
8:    fi;
9:    return C;
10: end SearchTree
```

Sequoia chooses the first, black-box approach. We found it to have better accuracy than the second approach and easier to unify the methodology for representing latency and bandwidth. However, bandwidth has converse semantics compared to latency—higher is better as opposed to smaller is better for latency. Hence, Sequoia reverses the order of bandwidth measures by subtracting each measure from a high constant before tree construction and subtracting predicted values from the same constant before providing it to the application. Note that, the choice of this constant does not affect the outcome of the prediction in any way. It just needs to be high enough to avoid negative values in the prediction tree.

3.3.2 Multiple trees

It turns out that the accuracy of the prediction tree depends on the choice of the lever and its location in the network. It is possible that the chosen lever is not well-suited to demarcate the relative positions of some hosts in the network, or the lever may not be able to measure its distance to some hosts. To mitigate the impact of bad levers and network problems, Sequoia uses multiple prediction trees referenced at distinct levers. Choosing the median distances estimated from the trees then helps in removing the outliers and improving the accuracy of prediction and server selection.

3.3.3 Triangle inequality violations

It is crucial for Sequoia to be resilient to triangle inequality violations. First, observe that violations of triangle inequality has a simple effect on Sequoia's prediction trees; they produce negative weights on the links. This is because the Gromov product can be negative if the triangle inequality does not hold. Negative link weights are not particularly a problem for computing distances on trees since trees are acyclic (there are no negative-weight cycles). Nevertheless, they might make a few distance estimates to come out negative.

Sequoia corrects the above aberration and avoids negative link weights by fixing the triangle inequality. It adds a large constant to each measured value before building the prediction tree and subtracts it back before presenting estimated distances to applications. Adding a large constant (similar to bandwidth representation) does not change the accuracy of prediction tree in any way. It simply results in the Gromov product being higher by that constant and consequently avoids any negative link weights.

3.3.4 Non-availability of measurements

Network measurements sometimes fail; a few hosts may not respond to measurement probes, or firewalls and intermediate gateways may block the probes. We designed Sequoia to use measurements opportunistically wherever possible and to be resilient to measurement failures whenever it occurs. Sequoia simply ignores unavailable measurements; it sets the Gromov product to negative infinity if one of the component measurements is not available. In the case that a host cannot measure itself to a lever, Sequoia uses the other prediction trees referenced at different levers.

3.3.5 Changing measures

Finally, latency and bandwidth are dynamic measures and change with time. At a high-level, Internet measures change at two fundamentally different time scales. Inherent properties of a path, such as the minimum latency and maximum capacity, change only when the path alters. Measurement studies show that Internet paths are typically stable for long periods (hours) and consequently provide ample window to adjust the models to these changes [24, 28, 43]. On the other hand, instantaneous latency and bandwidth depend on unpredictable traffic patterns making it expensive to constantly adapt to their changes.

Sequoia adapts to dynamic changes through periodic adjustments. Instead of modifying the prediction trees to reflect each change, it maintains a sliding window of trees constructed at different times. Periodically, it constructs a new tree based on the oldest prediction tree (at the same lever) but with new, updated measurements. Naturally, the periodicity of tree construction determines how well Sequoia can adjust to changes. Similar to other systems [24, 25], we think it is practical to reconstruct trees every few hours rather than minutes and capture the inherent properties of Internet paths well, rather than instantaneous values.

3.4 Architecture

Sequoia is conducive to be deployed in many ways. In this section, we make a few remarks about the choices for Sequoia's architecture.

Centralized service: Sequoia could be a centralized Web service like iPlane [25], exporting a query interface to external clients. Clients could query Sequoia to estimate a network property to a targeted host or choose the best server from a set of target hosts. Sequoia in turn could take advantage of opportunistic measurements reported by the clients or explicitly instruct clients to perform measurements to target hosts. This architecture might require additional effort to keep the central servers responsive and available.

Partially centralized system: A better architecture for Sequoia is to keep the participating hosts actively informed about their current

Dataset	Measure	Technique	Hosts	Measurements	$\triangle$ <> Violations
UC-PlanetLab	Latency	Ping	125	15625	15 %
Cornell-King	Latency	King [14]	2500	3123750	22 %
HP-PlanetLab	Bandwidth	pathChirp [33]	396	65077	40 %

Table 1: Summary of Datasets

"coordinates", that is, distance labels. A centralized server leverages measurements observed by the hosts, builds prediction (and anchor) trees efficiently, computes the distance labels, and informs the hosts. Hosts, in turn, can use the distance labels to perform latency/bandwidth prediction and server selection independently without consulting the centralized server.

Fully decentralized system: Finally, Sequoia could be a distributed system with no centralized server. The anchor trees provide a convenient distributed organization for such a deployment. A host could maintain neighbor relationships with other hosts on its path to the lever. The SeachTree algorithm in Figure 7 can be converted into a network protocol for finding the anchors and best servers through the above neighbor relationships (similar to peer-to-peer systems such as Gnutella [50]). Levers would serve as bootstrap hosts that new hosts can contact to join Sequoia with lever failures compensated by other levers in the system. Finally, if anchor hosts fail then their parents in the anchor tree can take over their role naturally.

4. EVALUATION

We next present an evaluation of Sequoia. First, we measure how accurately Sequoia represents the underlying datasets. Second, we examine Sequoia's practical benefits for selecting the closest and best-provisioned server. Finally, we discuss a few structural properties of Sequoia trees and highlight their topological correlations with the real world.

The evaluation is driven by the three real-world datasets (UC-PlanetLab, Cornell-King, and HP-PlanetLab) mentioned in Section 2. Table 1 summarizes the properties of these datasets. The datasets represent different network properties (two latency and one available bandwidth) measured on geographically spread-out hosts in the wide-area Internet (two between PlanetLab [48] hosts and one, Cornell-King, between infrastructure (DNS) servers). They are also of widely-different scales (125, 396, and 2500 hosts) and sometimes highly incomplete (only 40% of measurements in HP-PlanetLab). Finally, none of them is a strict metric as the datasets have a significant amount of triangle inequality violations (up to 40% in HP-PlanetLab).

Our evaluations were performed on an implementation of Sequoia's tree construction algorithms in C#. We construct different numbers of Sequoia trees with randomly chosen levers for each dataset and keep track and report the number of measurements required to construct the trees. All numbers reported in this section are based on the ConstructTree and SearchTree algorithms, which use selected, partial measurements from the datasets.

4.1 Accuracy of tree embedding

We first examine the accuracy of using Sequoia to model network latency and bandwidth. We measure *accuracy* as a distribution of the relative error: the ratio of the absolute difference between the predicted value from Sequoia trees and the true value from the dataset, over the true value, that is, $abs(tree\_value - graph\_value)/graph\_value$ [2]. In the presence of multiple Sequoia trees, the predicted tree value was taken as the median of the values estimated from all the trees.

We validate the accuracy of Sequoia using Vivaldi [8], the well-known coordinates-based approach to model network latencies, as a benchmark. We used the Vivaldi simulator [49] built at Harvard University to embed the datasets into a three-dimensional coordinate system (2 Euclidean coordinates + height). For Vivaldi, we set the number of neighbors to include all hosts for the two complete datasets and 50% of hosts for the third, incomplete dataset, and the number of iterations to match the size of the dataset so that both Vivaldi and Sequoia would use a similar number of measurements in the best case[3].

Figure 8 shows the CDF of relative errors for Sequoia in reference to Vivaldi for the two latency datasets: UC-PlanetLab (Figure 8(a)) and Cornell-King (Figure 8(b)).

For both latency datasets, Sequoia's accuracy is comparable to Vivaldi. Of course, Sequoia's accuracy depends on the number of trees used and, in general, improves with more trees. The number of trees required to achieve a satisfactory level of accuracy depends on the size of the dataset; Sequoia 5 does well for the UC-PlanetLab dataset while Sequoia 15 works for the Cornell-King dataset. Furthermore, the presence of triangle inequality violations do not seem to deter the effectiveness of Sequoia (or Vivaldi). Finally, Sequoia and Vivaldi tend to have a heavy tail of high errors. In practice, the heavy-tail does not affect the usefulness of either systems as most applications can tolerate the occasional errors. In theory, fortunately, treating network properties as approximate tree metrics provides a good handle in characterizing the worst-case performance of tree embedding algorithms as shown in [1].

For the bandwidth dataset (Figure 8(c)), Sequoia is clearly able to provide a reasonably accurate representation while Vivaldi fails. The inability of Vivaldi to model bandwidth is not surprising as Vivaldi was not designed to model bandwidth in the first place; bandwidth measurements cannot be intuitively tied to a coordinate space with Euclidean distances whereas it fits well into tree metrics as explained in Section 2. Sequoia's accuracy for bandwidth, however, seems to be lesser than for latency. Despite reduced accuracy, Sequoia can still make practically useful bandwidth estimates as shown in the next section.

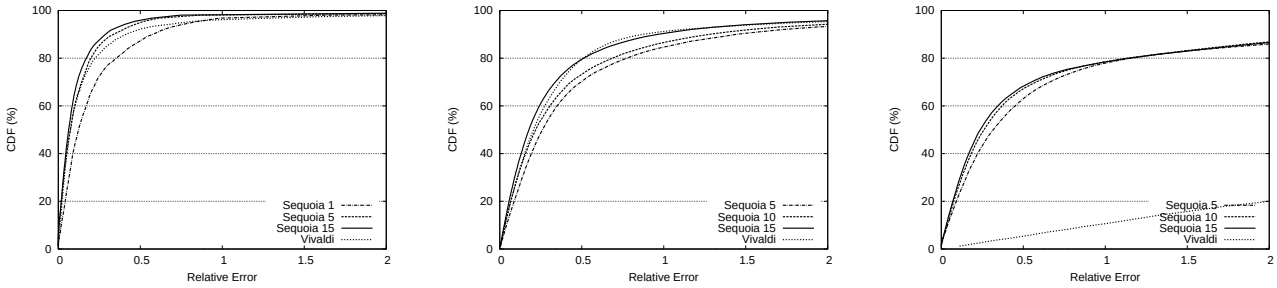
4.2 Accuracy of server selection

Next, we demonstrate some practical benefits of Sequoia by showing how well it enables selection of closest and best-provisioned (highest bandwidth) hosts.

A *target host* trying to find the best server can be both an existing participant of the system or an external entity. In the former case, if the system employs Sequoia, the target is already part of the Sequoia trees; it can then find the best server through the tree search

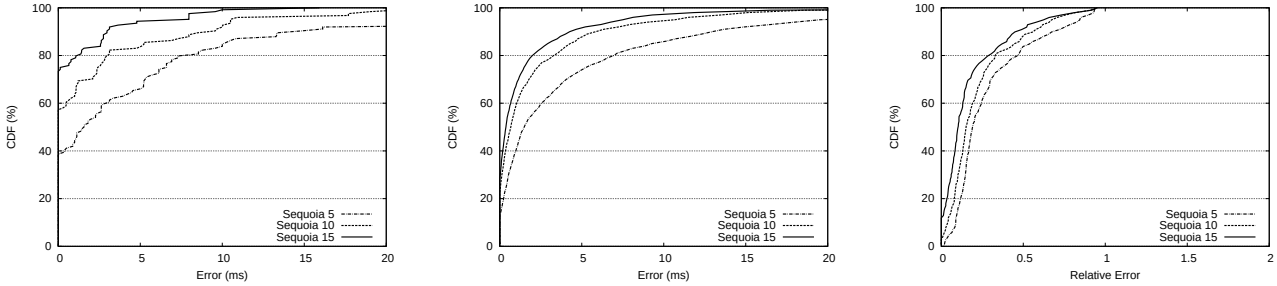
[2]Keong Lua et al. [23] argue for the need to employ other measures such as *relative-rank loss* to assess utility of a network positioning system from the point of view of applications. In this work, we stick with more traditional measures such as relative and absolute error, which are more commonly used in the evaluation of related systems [8, 11, 27].

[3]In general, Vivaldi can provide reasonable accuracy for embedding latency with a much smaller number of measurements. Our intention here is to show that embedding Internet latency into trees has the same level of accuracy as a state-of-the-art coordinate system. The fact that our approach can also embed bandwidth accurately while Vivaldi cannot makes the two systems incomparable.



(a) UC-PlanetLab Latency (b) Cornell-King Latency (c) HP-PlanetLab Bandwidth

Figure 8: Relative Embedding Error: Sequoia and Vivaldi have comparable accuracy in embedding latency. For bandwidth, Sequoia shows reasonable accuracy whereas Vivaldi fails.



(a) UC-PlanetLab Latency (b) Cornell-King Latency (c) HP-PlanetLab Bandwidth

Figure 9: Error in Server Selection: Sequoia finds closest and best-provisioned servers within reasonable error most of the time.

algorithm described in Section 3 without requiring additional measurements. In the latter case, where the target is an external entity, it might need to perform active measurements to guide its search. This results in a trade-off between the number of measurements performed and the quality of the best server found.

Here we show that with Sequoia, an external target can find a good quality server with a modest overhead. We show this through simulations where we set each host in the dataset to be an external entity seeking to find the closest or the best-provisioned server for one of the latency or bandwidth datasets respectively. The Sequoia trees built for the remainder of the dataset were then used to find the best server. The search algorithm on the Sequoia trees was restricted to only search at depth zero at each step, that is, as the anchor tree is descended from the lever only the children of each candidate are used to guide the search. At the end, we chose the best host out of the respective candidates found by each tree as the selected server.

Figure 9 presents the results for closest and best-provisioned server selection. It plots the quality of server selection as the error in the latency or bandwidth of the best host found by Sequoia versus the network measure to the best server in the dataset. We plot the absolute error for closest-server selection and the relative error for best-provisioned-server selection since bandwidth tends to have orders of magnitude variation.

Figure 9 shows that server selection through Sequoia works well in practice. Using a reasonable number of trees (5 for the UC-PlanetLab and HP-PlanetLab datasets and 15 for the Cornell-King dataset), Sequoia finds a closest server within 10 ms about 80% of the time and a best-provisioned server with less than 50% error 80% of the time. This quality of selection of the best-provisioned host is adequate in practice. Server bandwidths vary in orders of magnitude and the challenge often is in selecting a 10 Mbps server over a 1 Mbps server, which is within the ability of Sequoia.

4.3 Cost-benefit analysis

A natural inquiry next is to understand the trade off between accuracy and measurement overhead.

We first show the distribution of the number of measurements used for finding the anchors while each new host tried to join the trees. We plot a distribution because the number of measurements varies for each host. We plot the measurement overhead for different datasets as a CDF in Figure 10. Since each dataset has a different number of hosts, we plot the measurements as a fraction of the total number of hosts measured when a new host joins (x-axis).

Figure 10 shows that tree construction only consumes a small fraction of the measurements. Moreover, the curves are fairly steep indicating that most hosts use a similar number of measurements. Occasionally, however, a host might perform a large number of measurements (as shown by the tail). The most important observation is that the fraction of measurements required decreases with the number of hosts in the datasets; it is much lower (about 5% median) for the largest dataset (Cornell-King) compared to (about 20% median) the smallest dataset (UC-PlanetLab). The average number of measurements per host are 39 for UC-PlanetLab, 76 for HP-PlanetLab, and 149 for Cornell-King. This trend indicates that the measurement overhead scales sub-linearly with the number of hosts. In general, we expect the scaling to be logarithmic since the search is typically a walk down the tree.

Another crucial parameter that affects the tradeoff between accuracy and overhead is the *number of trees*. Figure 11 captures this tradeoff empirically by comparing the accuracy of server selection and the corresponding measurement overhead for different number of trees. In this figure, the accuracy is shown as the percentage of times a close or well provisioned server was selected within an error threshold. We picked 10ms as the error threshold for latency and 50% relative error for bandwidth.

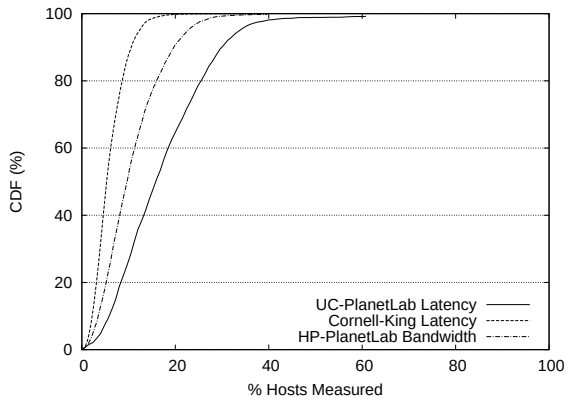


Figure 10: CDF of Measurements Performed for Tree Construction: Tree construction requires measurements to a small fraction of hosts and the amount of measurements scales sub-linearly with the number of hosts.

	Performance (percentile under error threshold)			Overhead (number of hosts measured)		
Trees	**5**	**10**	**15**	**5**	**10**	**15**
UC-PlanetLab	84	92	99	35	59	79
HP-PlanetLab	84	89	91	60	87	115
Cornell-King	86	95	97	79	176	238

Figure 11: Cost-Performance Tradeoff for Server Selection: Using more trees gives better accuracy but also increases measurement overhead. Accuracy here is shown as the percentile under an error threshold of 10ms for latency and 50% for bandwidth.

The accuracy of server selection—and correspondingly the cost—increases as more number of trees are used. The increase, however, is not linear but shows a tapering effect. There is "diminishing returns" for accuracy, and some measurements may be duplicate, and hence reused, when additional trees are included in the search. This tradeoff is clearly illustrated in the case of Cornell-King dataset: the percentile of servers selected within 10ms of the closest server increases from 86 for 5 trees, to 95 for 10 trees, and 97 for 15 trees while the corresponding measurement cost increases from 79 for 5 trees, to 176 for 10 trees, and 238 for 15 trees. An application can pick the right number of trees according to its requirements.

4.4 Topological properties

Finally, we discuss the properties of the tree topology that Sequoia constructs. We first show the path lengths of each host to the lever in the anchor tree; recall that this path represents the distance label associated with each host. Short path lengths are desirable as they reduce the memory footprint for storing distance labels of hosts.

Figure 12 shows the distribution of the host-lever path lengths in the anchor trees for each dataset. As expected, the path lengths are variable since Sequoia's tree-construction algorithms don't guarantee a perfectly-balanced tree. Yet, they are typically small and show low variance. For instance, in the largest Cornell-King dataset of 2500 hosts, the label lengths are all under 16 with the mode of the distribution being a reasonable 9 per tree. Even though these numbers are much higher than the typical number of coordinates in Vivaldi or GNP, they are still small enough for the memory capacities of modern systems.

While we presented aggregate statistics so far, the trees that Sequoia constructs also provide surprising revelations. Figure 13 shows a portion of the Sequoia tree constructed for the UC-PlanetLab latency dataset. At a high level, we found that the Sequoia tree was able to isolate hosts in different continents in the world into well-defined regions in the tree (sub-trees). This figure shows the European portion. The hosts are shaded by countries for clarity.

The Sequoia tree isolates hosts in different regions of Europe into well-defined clusters. For instance, there is a cluster at the top-right consisting of hosts in UK and Ireland (ie) and another at at the bottom-left with hosts in Poland (pl) and Germany (de). Hosts in larger regions, Spain (es) and Portugal (pt) and Norway (no), Sweden (se), and Finland (fi) are also well separated. The clustering is not perfectly geographic, however; a couple of UK nodes seem to be wrongly clustered.

We believe that this result strongly supports our intuition for treating network properties as tree metrics. It indicates that the Internet is largely hierarchical, more hierarchical in some regions (Europe) than others (USA), and the observed latencies follow the hierarchy. This result also opens Sequoia to be valuable for a wider-class of applications that benefit from building topology-aware overlays or hierarchical distributed systems.

4.5 Summary

Overall, the evaluation substantiates three key contributions of Sequoia. First, Sequoia provides the ability to construct an intuitive model for bandwidth using a small set of measurements and enable practical applications to perform effective bandwidth-based server selection. Second, Sequoia extends the same intuition to model latency while providing the same ease-of-use and accuracy as a state-of-the-art, coordinates-based latency model. Finally, Sequoia's tree models are well-correlated with the Internet topology, making it a promising tool to build topology-aware systems. These observations were drawn using datasets with the usual inconsistencies and vagaries representative of the real world.

5. APPLICATIONS

Sequoia promises unique abilities to networked systems. In this section, we discuss how Sequoia could benefit different network applications.

Server selection: Several applications often have the need to select a "best" host from a set of other hosts based on some quality criterion such as distance, bandwidth, load, or a combination of such criteria. Typical scenarios where this need arises include: a) peer-to-peer structured DHTs such as Chord [37] and Pastry [34], which try to connect peers with other closer peers as neighbors, b) peer-to-peer file sharing services and content distribution networks such as BitTorrent, in which, a peer host likes to download torrents from another closer, well-provisioned peer, and c) clients of online video streaming services that like to enrich their experience by connecting to a closer server with high bandwidth. We already showed how Sequoia enables selection of closest and best-provisioned hosts.

Constraint satisfaction: A few applications require host selection based on more complex constraints compared to the simple crite-

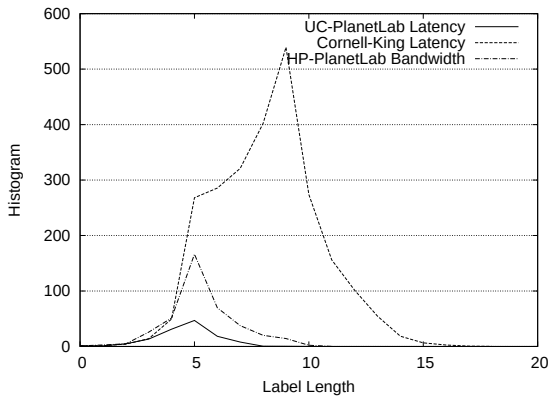


Figure 12: Distribution of Lengths of Distance Labels: The distance labels computed based on the path to the lever are typically small and scales sub-linearly with the number of hosts.

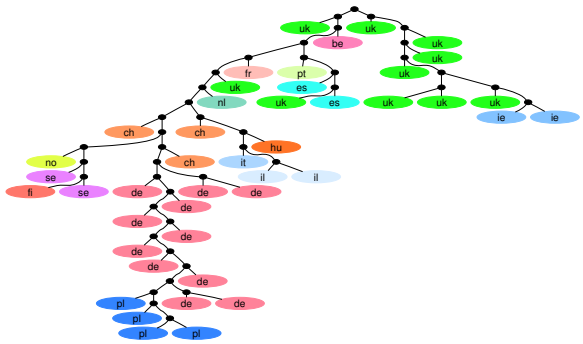


Figure 13: Prediction Tree for PlanetLab Hosts in Europe: The prediction tree seem to separate hosts in different countries in Europe into well-defined regions.

rion above. For instance, Voice-over-IP (VoIP) services, such as Skype, often try to locate an intermediate relay node with good quality paths to two end hosts, while online gaming systems with multiple players, such as XBox LIVE, benefit from a well-placed coordination server with good paths to all the client hosts.

Locating a server with good connections to multiple target hosts often requires extensive search among the set of hosts in the system. Sequoia can serve as an efficient data structure for resolving such constraint satisfaction queries. For instance, the common ancestor in the distance labels of the target hosts might be a good starting point for doing such searches.

Hierarchical organization: Finally, several distributed systems build a topology-aware hierarchy between the participating hosts: Examples include application-level multicast and video streaming protocols such as End System Multicast [6] and Bayeux [45], distributed network monitoring systems such as Astrolabe [32] and SDIMS [42], peer-to-peer overlays such as Meridian [41] and Coral [13]. Sequoia provides an inherent, topology-aware hierarchy for such distributed systems.

6. RELATED WORK

Efforts for representing Internet path measures broadly fall into two categories: 1) approaches that fit end-to-end measurements to abstract models and 2) approaches that construct detailed topology maps by probing routers and gateways in the core.

6.1 End-to-end approaches

IDMaps [11] pioneered the approach of distance approximation based on end-to-end measurements. Hosts in IDMaps measure their latency to a few well-placed "tracer" nodes and use *triangulation* to approximately estimate the latency to other hosts without direct measurements.

GNP [27] then introduced the popular technique of embedding distances into a low-dimensional coordinate system. In GNP, hosts measure their latency to well-placed servers called *landmarks* and then compute best-fit coordinates for themselves, representing Internet latency as Euclidean distances. ICS [22], Virtual Landmarks [39], and BBS [35] are similar to GNP but vary in their techniques to compute the coordinates. While, Lighthouses [29], PIC [7], and PCoord [21] compute coordinates in a fully decentralized manner, obviating the necessity of centralized landmark servers.

Vivaldi [8] is another fully decentralized coordinates-based network positioning system that embeds latency into a non-Euclidean space. It introduces an extended coordinate scheme with a "height" element that distorts the usual definition of Euclidean distance but provides more accurate embedding. Similarly, Lee et al. [20] introduce another extended, non-Euclidean coordinate scheme to model Internet latency. Shavitt and Tankel [36] observe that hyperbolic spaces are more suitable than Euclidean space to model Internet latency and propose a system to embed latency into a hyperbolic poincaré disk. In other related work, Key et al. [17] explore the idea of embedding latency into a non-metric space that retains symmetry but accommodates triangle inequality violations, and Mao et al. [26] embed latency in a non-metric space that supports both asymmetry and triangle inequality violations.

Sequoia is similar to the above systems in creating models of Internet path measures based on end-to-end measurements. It embeds latency and bandwidth onto trees assuming that these measures are approximate tree metrics. It is comparable to coordinates-based approaches in terms of ease of use by providing distance labels similar to coordinates, which can be used to locally estimate distances. It can also be easily decentralized although we have not explored a decentralized architecture in this paper. It assumes symmetry of distance values but can accommodate triangle inequality violations to some extent through the use of multiple trees.

Sequoia trumps prior work in its unique ability to model bandwidth in addition to latency with the same approach. This ability enables Sequoia to support a richer set of criteria for server selection. Moreover, Sequoia's approach is intuitive and matches with the topological properties of the Internet as presented in Section 2 and confirmed in Figure 13.

In addition to the above related work on approximate distance estimation, a few systems have been proposed for latency-based server selection. Meridian [41] organizes hosts into an overlay network and performs a multi-hop search, exponentially reducing the set of closest-host candidates at each hop. Tiers [3] implements a similar multi-hop protocol in a hierarchically organized system. At a high-level, Sequoia's SearchTree algorithm to narrow down best-host candidates on its anchor trees is also similar. In other work, Beaconing [18] proposes a simpler approach where a set of hosts called "beacons" track their distances to all hosts in the system and use this information to find hosts closest to a target, while Oasis [12] finds closest hosts geographically, that is, finds the physically closest host, using off-the-shelf techniques for estimating the geographic position of Internet hosts.

6.2 Topology-aware approaches

In contrast to the above end-to-end approaches, iPlane [25] and its more recent variant, iPlane Nano [24], have taken up the effort to measure the core of the Internet. They probe a large number of known routers and gateways (available in public sources such as Route Views [52]) with a wide-range of tools that can measure or estimate inter-link latency, loss-rate, and bandwidth using PlanetLab hosts as vantage points. Through extensive measurements and clever inference mechanisms, they build a topological map that serves as a useful information service for the Internet. In iPlane Nano, they compress the total size of the map to a size small enough to distribute to the end hosts and facilitate a decentralized deployment. The final compressed map is a a graph connecting routers (more accurately, PoPs) with annotations to represent link properties (capacity, latency, and loss rate) and routing policies.

Clearly, such a detailed model of the Internet would serve well for the purposes of distance estimation and server selection. However, the cost of measuring the internals of the entire Internet is high and needs to be adequately amortized across multiple applications. This deters a deployment of iPlane in small-to-medium-sized systems and hidden parts of the Internet (such as an enterprise network).

The goals of Sequoia are different, namely, to study and model the extent of treeness in Internet measures. This investigation adds to the understanding of the structural properties of the Internet and, in that process, presents a promising low-cost alternative based on end-to-end measurements for helping network services make informed decisions, especially when a complete structural map of the Internet is unavailable or expensive to construct.

6.3 Other related work

Sequoia's underlying theoretical principle has a close connection to the previously mentioned work by Shavitt and Tankel [36], which embeds Internet latency onto hyperbolic spaces. Hyperbolic spaces also define an approximate tree metric but with an additive approximation factor called the curvature. Our *4PC*-ϵ parameter is somewhat similar to the curvature of a hyperbolic space. However, it gives a multiplicative value in a pre-determined range $[0..1]$, which indicates how close a space is to a tree metric, whereas curvature gives an absolute additive term which bears no such insight (e.g., how close is a hyperbolic space whose curvature is 3?). More importantly, we directly produce tree embeddings, whereas Shavitt and Tankel map into a (hyperbolic) coordinate space.

Lebhar et al. [19] introduce another parameterized notion called *inframetric*, which captures the degree of triangle inequality violations in the Internet using a parameter ρ that indicates how close a set of measurements is to being a strict metric (and to an ultra metric). In this respect, ρ resembles our *4PC*-ϵ parameter. Our study may be seen as complementary, as many 4PC violations stem from triangle inequality violations.

Finally, a large body of theoretical work exists on the topic of embedding a metric space into a tree. These works [2, 15, 31], including ours [1], provide lower bounds on the accuracy of tree embeddings and provide algorithms with proven upper bounds for constructing tree models.

7. CONCLUSIONS

This paper presented an empirical study of the treeness of Internet end-to-end measures such as latency and bandwidth. We performed this study by directly embedding network measurements into virtual trees. One of our key contributions is the tree construction technique to perform approximate tree embeddings of network measures taking into account practical constraints such as unavailability of measurements, triangle inequality violations, and the need to scale.

In that process, we introduced a unified approach for modeling bandwidth and latency. Our tree models represent latency and bandwidth with good accuracy and enable the selection of low latency or high bandwidth servers with high fidelity, while roughly capturing the relative positions of hosts in the Internet. This paper quantified the treeness of Internet latency and bandwidth, presented the design and implementation of a system to represent them concisely, and evaluated its benefits.

Acknowledgments

We thank our shepherd, Jia Wang, and reviewers for their insightful comments and feedback.

8. REFERENCES

[1] I. Abraham, M. Balakrishnan, F. Kuhn, D. Malkhi, V. Ramasubramanian, and K. Talwar. Reconstructing Approximate Tree Metrics. In *Proc. of Symposium on Principles of Distributed Computing (PODC)*, Portland, OR, Aug. 2007.

[2] I. Abraham, Y. Bartal, and O. Neiman. Embedding Metrics into Ultrametrics and Graphs into Spanning Trees with Constant Average Distortion. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, New Orleans, LA, Jan. 2007.

[3] S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Scalable peer finding on the Internet. In *Proc. of the Global Internet Symposium*, Taipei, Taiwan, Nov. 2002.

[4] A. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, 8:509–512, Oct. 1999.

[5] P. Buneman. A Note on Metric Properties of Trees. *Journal of Combinatorial Theory*, 17:48–50, 1974.

[6] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. A Case for End System Multicast. In *Proc. of Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Santa Clara, CA, June 2000.

[7] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. In *Proc. of International Conference on Distributed Computing Systems (ICDCS)*, Tokyo, Japan, Mar. 2004.

[8] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proc. of ACM SIGCOMM Conference*, Portland, OR, Aug. 2004.

[9] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing Residential Broadband Networks. In *Proc. of SIGCOMM Internet Measurement Conference (IMC)*, San Diego, CA, Oct. 2007.

[10] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. In *Proc. of ACM SIGCOMM Conference*, Cambridge, MA, Aug. 1999.

[11] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, 2001.

[12] M. Freedman, K. Laskhminarayanan, and D. Mazières. OASIS: Anycast for Any Service. In *Proc. of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.

[13] M. Freedman and D. Mazières. Sloppy Hashing and Self-Organizing Clusters. In *Proc. of International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, Feb. 2003.

[14] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proc. of SIGCOMM Internet Measurement Workshop (IMW)*, Marseille, France, Nov. 2002.

[15] A. Gupta. Steiner Points in Tree Metrics don't (Really) Help. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Washington, DC, Jan. 2001.

[16] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang. A Measurement Study of Internet Bottlenecks. In *Proc. of INFOCOM*

Conference, Miami, FL, Mar. 2005.

[17] P. Key, L. Massoulie, and D.-C. Tomozei. Non-Metric Coordinates for Predicting Network Proximity. In *Proc. of the IEEE INFOCOM Conference*, Phoenix, AZ, Apr. 2008.

[18] C. Kommareddy, N. Shankar, and B. Bhattacharjee. Finding Close Friends on the Internet. In *Proc. of IEEE International Conference on Network Protocols (ICNP)*, Nov. 2001.

[19] E. Lebhar, P. Fraigniaud, and L. Viennot. The Inframetric Model for the Internet. In *Proc. of IEEE INFOCOM Conference*, Apr. 2008.

[20] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha. On Suitability of Euclidean Embedding of Internet Hosts. In *Proc. of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Saint Malo, France, June 2006.

[21] L. Lehman and S. Lerman. PCoord: Network Position Estimation Using Peer-to-Peer Measurements. In *Proc. of IEEE International Symposium on Network Computing and Applications (NCA)*, Aug. 2004.

[22] H. Lim, J. Hou, and C.-H. Choi. Constructing Internet Coordinate System based on Delay Measurement. In *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC)*, Miami, FL, Oct. 2003.

[23] E. K. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft. On the Accuracy of Embeddings for Internet Coordinate Systems. In *Proc. of ACM SIGCOMM Conference on Internet Measurement*, Berkeley, CA, USA, Oct. 2005.

[24] H. V. Madhyastha, E. K. Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: Path Prediction for Peer-to-Peer Applications. In *Proc. of the Usenix Conference on Networked Systems Design and Implementation (NSDI)*, Apr. 2009.

[25] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *Proc. of the Usenix Conference on Operating Systems Design and Implementation (OSDI)*, Nov. 2006.

[26] Y. Mao and L. K. Saul. Modeling Distances in Large-Scale Networks by Matrix Factorization. In *Proc. of ACM SIGCOMM Conference on Internet Measurement (IMC)*, Taormina, Sicily, Italy, Oct. 2004.

[27] E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-based Approaches. In *Proc. of the INFOCOM Conference*, New York, NY, June 2002.

[28] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, Oct 1997.

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

[30] R. Prasad, M. Murray, C. Dovloris, and kc Claffy. Lower Bounds on the Distortion of Embedding Finite Metric Spaces in Graphs. *Discrete & Computational Geometry*, 19, 1998.

[31] Y. Rabinovich and R. Raz. Lower Bounds on the Distortion of Embedding Finite Metric Spaces in Graphs. *Discrete & Computational Geometry*, 19, 1998.

[32] R. v. Renesse, K. P. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *ACM Transactions on Computer Systems*, 21(2):164–206, 2003.

[33] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Avalable Bandwidth Estimation for Network Paths. In *Proc. of Passive and Active Measurement Workshop*, San Diego, CA, Apr. 2003.

[34] A. Rowstorn and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems. In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, Nov. 2001.

[35] Y. Shavitt and T. Tankel. Big-bang Simulation for Embedding Network Distances in Euclidean Space. *IEEE/ACM Transactions on Networking*, 12(6):993–1006, 2004.

[36] Y. Shavitt and T. Tankel. Hyperbolic Embedding of Internet Graph for Distance Estimation and Overlay Construction. *IEEE/ACM Transactions on Networking*, 16(1):25–36, 2008.

[37] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM*, San Diego, CA, Aug. 2001.

[38] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterzing the Internet Hierarchy from Multiple Vantage Points. In *Proc. of the Infocom Conference*, New York, NY, June 2002.

[39] L. Tang and M. Crovella. Virtual landmarks for the internet. In *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC)*, Oct. 2003.

[40] L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos. A Simple Conceptual Model for the Internet Topology. In *Proc. of Global Telecommunications Conference (GLOBECOM)*, San Antonio, TX, Nov. 2001.

[41] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *Proc. of ACM SIGCOMM Conference*, Philadelphia, PA, Aug. 2005.

[42] P. Yalagandula and M. Dahlin. A Scalable Distributed Information Management System. In *Proc. of ACM SIGCOMM Conference*, Porland, OR, Aug. 2004.

[43] Y. Zhang and N. Duffield. On the Constancy of Internet Path Properties. In *Proc. of ACM SIGCOMM Workshop on Internet Measurement*, San Francisco, CA, Nov. 2001.

[44] H. Zheng, E. K. Lua, M. Pias, and T. Griffin. Internet routing policies and round-trip-times. In *Proceedings of the Passive Active Measurement*, pages 236–250, 2005.

[45] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: An Architecture for Scalabe and Fault-Tolerant Wide-Area Data Dissemination. In *Proc. of Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Port Jefferson, NY, June 2001.

[46] Akamai SureRoute. `http://www.akamai.com/dl/feature_sheets/fs_edgesuite_sureroute.pdf`.

[47] Meridian: A Lighweight Approach to Network Positioning. `http://www.cs.cornell.edu/People/egs/meridian`.

[48] PlanetLab: An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services. `http://www.planet-lab.org`.

[49] Network Coordinate Research at Harvard. `http://www.eecs.harvard.edu/~syrah/nc/`, 2006.

[50] The Gnutella 0.4 Protocol Specification. http://dss.clip2.com/GnutellaProtocol0.4.pdf, 2000.

[51] S^3: Scalable Sensing Service. `http://networking.hpl.hp.com/s-cube`.

[52] University of Oregon Route Views Project. `http://www.routeviews.org`.

[53] All-Sites-Pings for PlanetLab. `http://ping.ececs.uc.edu/ping/`, 2006.