Measurement-Based Analysis, Modeling, and Synthesis of the Internet Delay Space for Large-Scale Simulation

Bo Zhang, T. S. Eugene Ng, Animesh Nandi, Rudolf Riedi, Peter Druschel*, Guohui Wang
Rice University and Max Plank Institue*

ABSTRACT

The characteristics of packet delays among edge networks in the Internet can have a significant impact on the performance and scalability of global-scale distributed systems. Designers rely on simulation to study design alternatives for such systems at scale, which requires an appropriate model of the Internet delay space. The model must preserve the geometry and density distribution of the delay space, which are known, for instance, to influence the effectiveness of self-organization algorithms used in overlay networks.

In this paper, we characterize measured delays between Internet edge networks with respect to a set of relevant metrics. We show that existing Internet models differ dramatically from measured delays relative to these metrics. Then, based on measured data, we derive a model of the Internet delay space. The model preserves the relevant metrics, allows for a compact representation, and can be used to synthesize delay data for large-scale simulations. Moreover, specific metrics of the delay space can be adjusted in a principled manner, thus allowing systems designers to study the robustness of their designs to such variations.

1. INTRODUCTION

Designers of large-scale distributed systems rely on simulation and network emulation to study design alternatives and evaluate prototype systems at scale and prior to deployment. To obtain accurate results, such simulations or emulations must include a model of the Internet delay space that accurately reflects those characteristics of real Internet delay that influence system performance. For example, having realistic clustering properties is important because they can influence the load balance of delay-optimized overlay systems, and the effectiveness of server placement policies and caching strategies. Having realistic growth characteristics in the delay space is another key, because the effectiveness of many self-organization algorithms depends on growth characteristics. Many distributed systems are also sensitive to the inefficiency of IP routing with respect to delay. Such inefficiency must be realistically reflected in the delay space as well.

Currently, two approaches are used to obtain a delay model. The first approach, adopted for instance by the P2PSim simulator [18], is to collect actual delay measurements using a tool such as King [10]. This approach yields a realistic delay space. However, due to limitation of the measurement methodology and the $O(N^2)$ space requirement of a delay matrix, the scope of measured delay data is limited. For example, P2PSim currently comes with a 1740$\times$1740 delay space matrix. Moreover, raw measured data often contains errors and missing values.

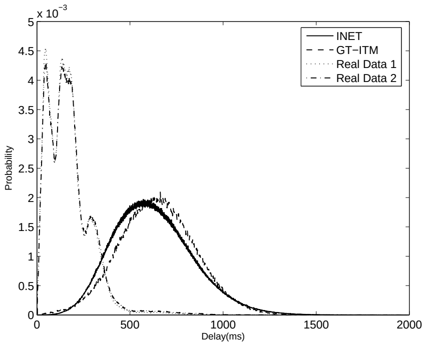


Figure 1: Delay distribution comparison for different data sets.

The second approach is to start with an artificial network topology model (e.g. [36, 38, 8, 9, 13]) and assign artificial link delays. The delay space is then computed using a shortest-path routing algorithm over the topology. Such artificially generated delay models differ dramatically from actual Internet delay spaces. Figure 1 compares the delay distributions of artificial delay spaces based on the GT-ITM [38] and the Inet [37] topology models against two measured Internet delay spaces. The artificial delay distribution does not even remotely resemble that of the measured data. Clearly, there are rich features in Internet delay spaces that are not captured in the artificial delay models.

This study addresses the short-comings of existing approaches for obtaining a delay model. First, we quantitatively analyze measured Internet delay spaces, specifically, the static round-trip propagation delays among edge networks in the Internet. Then, we identify a set of metrics that are relevant to distributed systems design, and characterize the measured Internet delay space with respect to these metrics. Based on these analytical insights, we develop a method to model measured Internet delay spaces. The resulting model has a highly compact $O(N)$ representation (as opposed to the $O(N^2)$ matrix representation) that accurately preserves all the relevant delay space characteristics, and can interpolate missing measurements.

Finally, we develop a method to synthesize an artificial delay space based on the properties of a measured delay space. The method exploits the scaling characteristics found in the measurements and makes it possible to synthesize a realistic delay space larger than the measured delay space. Moreover, specific characteristics in the synthesized delay space can be adjusted in a principled manner, allowing systems designers to study the robustness of their designs to such variations.

We make two primary contributions in this work:

- We systematically quantify the properties of the Internet delay space with respect to a set of statistical, structural, and routing metrics relevant to distributed systems design. This leads to new fundamental understandings of the Internet delay space characteristics that may inform future work.

- We develop a set of building block techniques to model and synthesize the Internet delay space compactly while accurately preserving all relevant metrics. The compact representation enables accurate and memory efficient network simulations and emulations at large scale.

The rest of the paper is organized as follows. First, we describe our methodology for collecting the Internet delay space data in Section 2. We motivate the relevant analytical metrics and present our analytical findings in Section 3. In Section 4, we exploit our quantitative understanding of the Internet delay space to develop a compact model and evaluate its effectiveness. We extend the delay space model to incorporate techniques to synthesize an artificial delay space and evaluate the accuracy of the synthesized delay space in Section 5. We summarize the related work in Section 6 and make several concluding remarks in Section 7.

2. DATA COLLECTION METHODOLOGY

Our measured Internet delay space data sets were collected between October 2005 to November 2005 using a slightly modified version of the King [10] tool. King measures the RTT between DNS servers. We select random DNS servers from the entire Internet. Only one DNS server is used per organization (as inferred from the domain names). This way, each DNS server in the data set approximates the location of one edge network. By measuring RTT among these DNS servers, we get a delay space for the corresponding edge networks. Over this time period, we collected two delay space matrices: the Real Data 1 set has 3997 rows and columns with a missing data rate of 13 percent, the Real Data 2 set has 3994 rows and columns with a missing data rate of 10 percent.

Next, we describe the measurement process in more detail. We start from a list of random IP addresses drawn from the prefixes announced in BGP. This data is published by the Route Views project [24]. For each IP prefix, we perform a reverse DNS lookup to determine the associated DNS servers. Each reverse lookup returns a set of DNS servers D_{IP_i}. We keep only the DNS server sets in which at least one server supports recursive queries, since King requires it. If two DNS server sets D_{IP_i} and D_{IP_j} overlap, then only one of the two sets is kept. If there are more than one server in a set, then all the servers in the set must be physically close. We check this by pinging all the servers in the set to ensure that they have identical RTT from our perspective.

Finally, among the remaining DNS server sets, we choose one server that supports recursive queries per set to conduct our measurements. The RTT between each pair of DNS servers is measured at least 20 times and no more than 50 times in each direction. If fewer than 20 valid measurements are obtained in each direction, the measurement is discarded. The minimum RTT for each direction is recorded. If the RTT from the two directions disagree by more than 10%, the measurement is discarded. Then we eliminate all values that are smaller than 100 microseconds and greater than 2 seconds since such values are considered too extreme to be reasonable and are most likely measurement errors. Finally, we conduct a shortest path analysis over the matrix to eliminate edges that are involved in more than 10000 shortest paths, which indicates a measurement error. Obviously, due to measurement errors, many DNS servers must be discarded in the end. In our data sets, we start with a list of 5000 DNS servers and in the end obtain roughly a 4000x4000 matrix with an acceptable amount of missing data. To understand the properties in the data sets under scaling, we consider 4 different random sub-sample sizes: 800, 1600, 2400, and 3200. To reduce the sensitivity to a particular random sample, for each sub-sample size, we consider 5 random sets. Results presented in this paper are averaged over these 5 random sets.

In addition to the measured data sets, we also use artificial delays generated by connectivity models for comparisons. The two generators we use are Inet [37] and GT-ITM [38]. The Inet generator creates a topology that has power-law node degree distribution properties. The GT-ITM generator is used to generate a topology based on the Transit-Stub model. Although GT-ITM provides an option to generate triangle inequality violations, we did not use this option. For Inet, to generate the delays, we use the standard method of placing nodes randomly on a 2D plane and then use the 2D Euclidean distance between a pair of connected nodes as the link latency. For GT-ITM, we assign transit and stub link delays accordingly. In both cases, all pair shortest path routing is used to generate the resulting artificial delay space matrices. We scale the delays such that the maximum delay is 1600ms. This constant scaling factor does not affect the structure of the generated delay spaces, we do this only to simplify the presentation of results.

3. ANALYSIS OF INTERNET DELAY SPACE

A fundamental problem faced by all data analysis studies is how to select a sensible set of metrics to characterize the data. After all, there is virtually an unlimited number of statistics that one can compute. Keeping in mind that our objective is to understand the Internet delay space with respect to distributed systems design, we first identify a set of metrics that are known to significantly influence the performance of distributed systems. Then, we analyze measured Internet delay data with respect to these and other statistical and structural properties. The results give new insight into the characteristics of the Internet delay space, and they inform the design of an appropriate model.

3.1 Systems-motivated Metrics

We begin by identifying a set of properties of the delay space that are known to strongly influence distributed system design and performance. While there may exist other relevant metrics, we believe the chosen set does capture a wide range of important issues in distributed systems design and evaluation.

Global clustering - This metric characterizes clustering in the delay space at a macroscopic level. For instance, the continents with the largest concentration of IP subnetworks (North America, Europe and Asia) form recognizable clusters in the delay space. This global clustering structure is, for instance, relevant to the placement of large data centers and web request redirection algorithms (e.g. [21]).

Our algorithm to determine the global clustering works as follows. Given N nodes in the measured input data, it first treats each node as a singleton cluster. The algorithm then iteratively finds two closest clusters to merge. The distance between two clusters is defined as the average distance between the nodes in the two clusters. A cutoff delay value determines when to stop the merging process. If the distance between the two closest clusters is larger than the cutoff, the merging process stops. By varying the cutoff value and monitoring the resulting cluster sizes, the global clustering properties can be determined.

Local clustering - This metric characterizes clustering in the delay space at a microscopic level. It is based on a directed graph formed by having each node point to its nearest neighbor in the delay space. This metric is relevant, for instance,

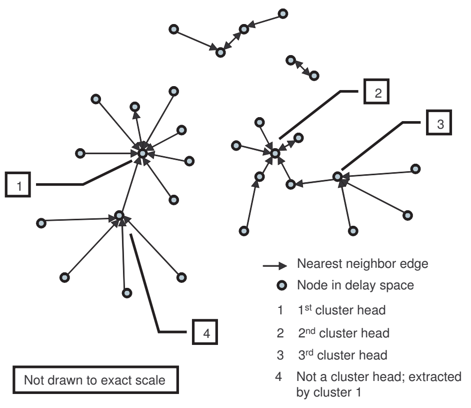


Figure 2: Nearest neighbor directed graph analysis technique.

to the in-degree and thus the load balance among nodes in delay-optimized overlay networks (e.g. [5]). For example, dense local clustering can lead to an overlay node having an unexpectedly high number of overlay neighbors and can potentially create a load imbalance in the overlay.

Based on the measured delay data, for each node in the delay space, we create a directed edge to the node with the smallest delay. The resulting graph is the nearest neighbor directed graph. Then, we can analyze the in-degree of nodes in the graph. Moreover, we can use the graph to identify a set of local cluster heads (or centers). We select the node with the highest in-degree as a local cluster head and remove it and its immediate children from the graph. This step is applied repeatedly to identify the next local cluster head until no more nodes remain. Since a local cluster resembles a star network, we sometimes simply call it a star. The process is illustrated in Figure 2. The importance of the local cluster heads will become clear in subsequent sections.

Growth metrics - Distributed nearest neighbor selection is a hard problem, but efficient algorithms have been identified to solve the problem for growth-restricted metric spaces [12]. These algorithms are used, for instance, in Tapestry [39] and Chord [32] to select overlay neighbors. In a growth-restricted metric space, if the number of nodes with a delay of at most r from some node i is $B_i(r)$, then $B_i(2r) \leq c \cdot B_i(r)$, where c is a constant. We characterize the growth properties of the Internet delay space by evaluating $B(2r)/B(r)$.

Another metric based on nearest neighbors is the $D(k)$ metric. Let $d(i,k)$ be the average delay from a node i to its k closest nodes in the delay space and N be the set of nodes, then $D(k) = \frac{1}{|N|} \sum_{i \in N} d(i,k)$. Structured overlay networks like Chord, Tapestry and Pastry employ proximity neighbor selection (PNS) to reduce the expected delay stretch S, i.e., the ratio of the delay of an overlay route over the direct routing delay averaged over all pairs of nodes [11, 4, 22,

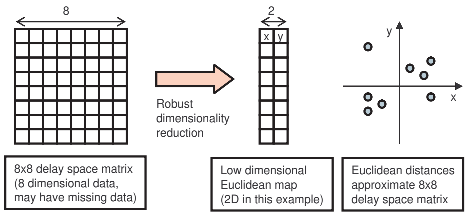

8
2
x
y
Robust dimensionality reduction
8x8 delay space matrix (8 dimensional data, may have missing data)
Low dimensional Euclidean map (2D in this example)
Euclidean distances approximate 8x8 delay space matrix

Figure 3: Robust dimensionality reduction.

5]. We choose to include the $D(k)$ metric because analysis have shown that in Tapestry and Pastry, the expected delay stretch S in the overlay can be predicted based on the function $D(k)$ [5].

Triangle inequality violations - The triangle inequality states that given points x, y and z in a Euclidean space, the distances d_{ij} between points i and j satisfy $d_{xz} \leq d_{xy} + d_{yz}$. The Internet delay space, however, does not obey the triangle inequality, since Internet routing may not be optimal with respect to delay. Unfortunately, many distributed nearest neighbor selection algorithms rely on the assumption that the triangle inequality holds [25, 12, 35]. Thus, it is important to characterize the frequency and severity of the violations in the Internet delay space.

Routing inefficiency - This metric is closely related to triangle inequality violation. In particular, routing inefficiency is the ratio of the delay between two nodes over the delay of the best possible overlay path in the delay space. Some systems [3] exploit overlay routing for performance gain. The routing inefficiency metric quantifies the maximum possible delay reduction achievable by such systems.

3.2 Structural Properties

In Figure 1, we can observe that the delay distributions of the measured data sets have certain characteristic peaks. This suggests that nodes form clusters in the data. In contrast, the delay distributions for the topology models do not indicate such behavior. Analysis of random data sub-samples indicates that the delay distribution is also independent of sample size. To visualize the locations of nodes, we embed the data sets into a 2D Euclidean space using a robust dimensionality reduction procedure. The high level idea of dimensionality reduction is illustrated in Figure 3. Several techniques exist to compute such an embedding robustly even when some missing measurements are present [17, 7, 27, 6, 14, 34]. In this paper, we adopt a slightly modified version of the GNP [17] method to compute the embedding. This modified version simply ignores the missing measurements in the delay space matrix when computing node coordinates. GNP uses Landmarks as global reference points to compute coordinates. We observe that as long as enough random landmarks are chosen, the accuracy of the embedding is not affected by the missing measurements.

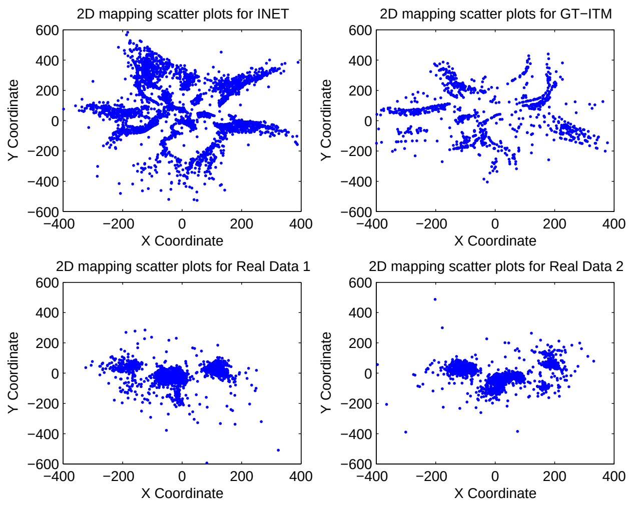


Figure 4: 2D coordinates scatter plot comparison.

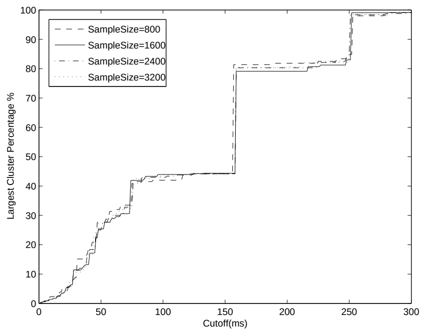
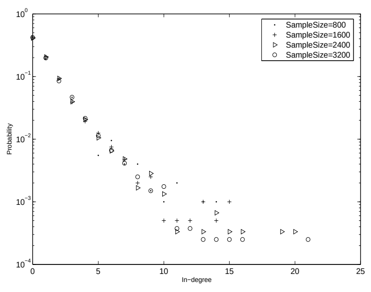


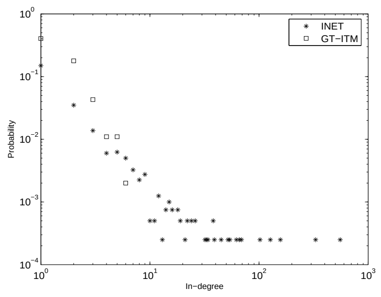
Figure 5: Clustering results for different sample sizes.

Figure 4 displays the scatter plots of the 2D Euclidean coordinates generated for different data sets. The visual differences between the measured data and the topology models are striking. It is easy to see that there are three dominant clusters in the measured data. Using the NetGeo tool [16], we learn that the clusters approximately correspond to three continents: North America, Europe, and Asia. In contrast, the nodes in the topology models tend to cluster around lines radiating from the center. It is likely that this pattern results from the tree-like structure of the topology models; however, a full analysis of this complex patterns is beyond the scope of this paper.

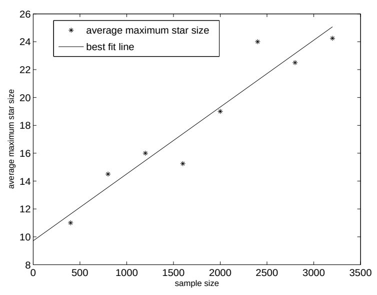
To quantify the global clustering properties in the measured data sets, we apply the described global clustering algorithm and plot the percentage of nodes in the largest cluster against different clustering cut-off thresholds in Figure 5. Regardless of the sample size, the largest cluster's size increases sharply at cutoff values 155ms and 250ms. These sharp increases are caused by the merging of two major clusters at these thresholds. The steps suggest that there are three major



(a)



(b)



(c)

Figure 6: Local cluster analysis. (a) Exponential-like in-degree distribution for measured data (log scale). (b) Power-law-like in-degree distribution for INET and GT-ITM (log-log scale). (c) Average maximum star size versus sample size, with best linear fit line.

Sample size	# Cluster heads	Percentage
800	185	23.1%
1600	363	22.7%
2400	547	22.8%
3200	712	22.3%
3997 (all data)	884	22.1%

Table 1: Average fraction of nodes classified as cluster heads in Real Data 1.

clusters. By setting the threshold to 120ms, we are able to effectively classify nodes into the three major clusters. They roughly account for 45%, 35%, and 9% of the nodes, respectively. The remaining 11% are nodes that are scattered outside of the major clusters.

The global clustering analysis reveals the coarse-grained structure of the delay space. To understand the fine-grained structure, we conduct the nearest neighbor directed graph analysis on the data sets. Figure 6(a) shows the in-degree distributions for different sample sizes. Observe that the in-degree distribution for the measured data is nearly exponential except for the extended tail. The maximum in-degree also increases with the sample size. In contrast, as shown in Figure 6(b), the in-degree distributions for the Inet and GT-ITM topologies follow closely the power-law distribution. This implies that while local cluster sizes vary, they vary more mildly in reality than in the topology models.

We classify the nodes into local cluster heads (or star heads) and non-heads using the procedure described in 3.1. Table 1 shows that the proportion of nodes in the data that are classified as local cluster heads is independent of the sample size. Moreover, as shown in Figure 6(c), the average maximum local cluster size (or star size) scales linearly with the sample size. These properties will become useful when we turn to the synthesis of delay spaces later in the paper.

To conclude our analysis of the structural properties, we turn to analyzing spatial growth. Figure 7 shows the median $B(2r)/B(r)$ growth of the data sets. We plot the median because, unlike the mean, it is insensitive to the extreme outliers and can better characterize the dominant trends. As can be seen, the topology models have far higher peak spatial growth than the measured data (note the log scale). In the measured data, the initial growth is higher when the ball is expanding within a major cluster. As soon as the ball radius covers the nodes in each major cluster, growth slows down as expected. Further more, this growth trend in the measured data is invariant across different sample sizes.

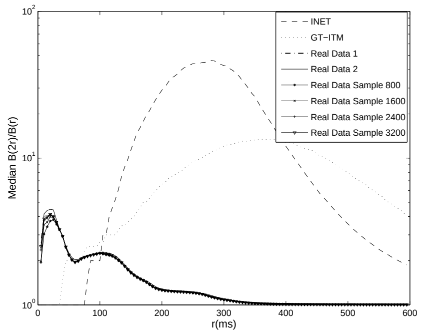


Figure 7: $B(2r)/B(r)$ metric comparison for different data sets (log scale).

In terms of the $D(k)$ metric, we also observe dramatic differences between topology models and the measured data. Figure 8 indicates that in the topology models, from the perspective of an observer node, even the nearest nodes have delays comparable to the overall average delay of all the nodes. Thus, delay is a fairly poor differentiator for sorting nodes in the topology models. In contrast, in the measured data, the delays of nodes from an observer node are more evenly spread throughout the entire range, thus delay can be used as a effective differentiator. Finally, observe that the $D(k)$ metric is invariant across different sample sizes.

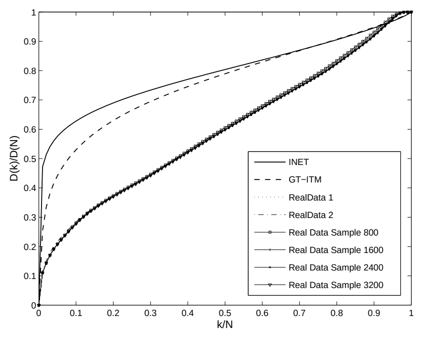


Figure 8: $D(k)/D(N)$ metric comparison for different data sets.

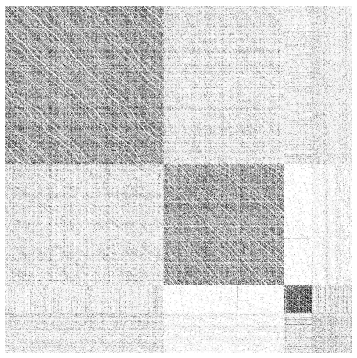


Figure 9: Type 1 triangle inequality violations for Real Data 1 (white color is most severe).

3.3 Routing Properties

We next analyze the measured data sets with respect to properties related to routing: triangle inequality violations and routing inefficiency. We say that an edge ij in the data set causes a Type 1 triangle inequality violation if for some node k, $\frac{d_{ik}+d_{kj}}{d_{ij}} < 1$, and it causes a Type 2 violation if $\frac{|d_{ik}-d_{kj}|}{d_{ij}} > 1$. Intuitively, better overlay paths can be found for edges that cause Type 1 violations, and edges that cause Type 2 violations can potentially provide short-cut overlay paths.

For each edge ij, we count the number of Type 1 violations it causes. To illustrate how the number of triangle inequality violations are distributed over the major clusters, we present a matrix in Figure 9 for Real Data 1; the result for Real Data 2 is similar. To produce this figure, we first reorganize the original data matrix by grouping nodes in the same clusters together, such that the matrix indices of the nodes in the largest cluster are the smallest, the indices for nodes in the second largest cluster are next, and so on. Nodes that do not belong to the three largest clusters are assigned the largest matrix indices. The top left corner has indices (0,0).

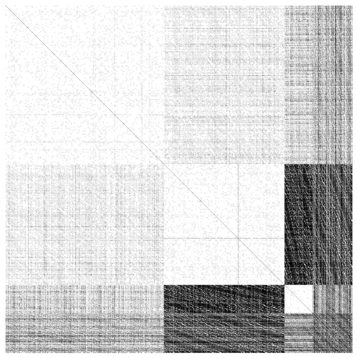


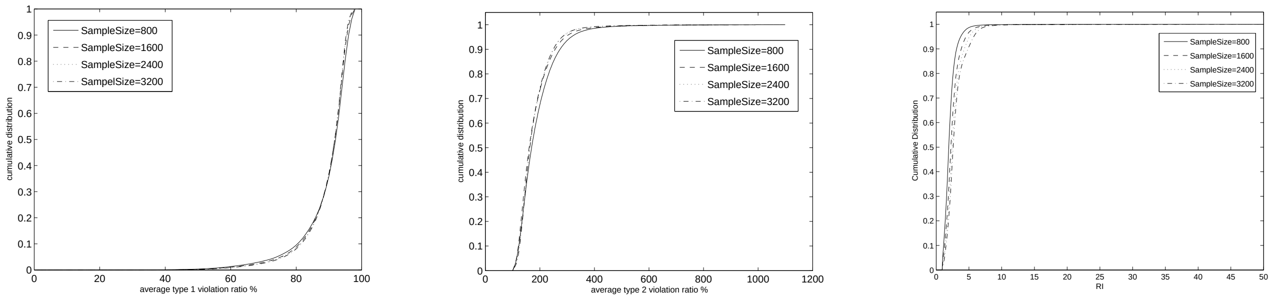
Figure 10: Type 2 triangle inequality violations for Real Data 1 (white color is most severe).

Each point (i, j) in the plot represents the number of Type 1 violations that the edge ij is involved in as a shade of gray. A black point indicates no violation and a white point indicates the maximum number of violations encountered for any edge in the analysis. Missing values in the matrix are drawn as white points.

It is immediately apparent that clustering is very useful for classifying triangle inequality violations. It can be seen that edges within the same cluster (i.e. the 3 blocks along the diagonal) tend to have significantly fewer Type 1 violations (darker) than edges that cross clusters (lighter). Also, the number of violations for edges connecting a given pair of clusters is quite homogeneous. Figure 10 shows the corresponding results for Type 2 violations. Here, the trend is reversed: The short edges within a given cluster cause the most violations. The number of violations for edges connecting a given pair of clusters is again fairly homogeneous. These results imply that, if two nodes are within the same major cluster, then the chance of finding a shorter overlay path is far lower then when the nodes are in different clusters. Moreover, edges that are used to form better overlay paths are most likely found inside a cluster.

We show in Figure 11(a) and Figure 11(b) the cumulative distributions of Type 1 and Type 2 violation ratios for different sample sizes. Note that these distributions are independent of sample size.

Figure 11(c) shows the cumulative distributions of routing inefficiency (RI) across different sample sizes. The potential



(a) (b) (c)

Figure 11: (a) Type 1 violation ratios distribution for different sample sizes. (b) Type 2 violation ratios distribution for different sample sizes. (c) Routing inefficiency distribution for different sample sizes.

gain by performing overlay routing is moderate. We observe that as the sample size increases, the RI increases. This is due to the fact that as the sample size increases, more short edges that can form better overlay paths are discovered. The distributions of the underlying triangle inequality violation ratios, however, does not change with sample size as shown in Figure 11(a) and (b). Interestingly, we also observe that only 42,702 edges among the 6,883,700 possible edges are used to construct the better overlay paths. This suggests that there exists some small, low delay ”backbone” networks that can be exploited by overlay routing.

3.4 Summary

In summary, our empirical analysis of the measured Internet delay spaces between edge networks provides a number of new insights:

- The three continents, North America, Asia and Europe dominate the coarse-grained cluster structure of the Internet delay space.
- The in-degree distribution of the directed nearest neighbor graph resembles an exponential distribution, but with a long tail; it does not follow a power law. The maximal in-degree increases with the sample size.
- The relative number of local clusters appears independent of the sample size, though the maximal size of the clusters increases linearly with the sample size.
- The growth metrics clearly reflect the coarse-grained structure of the delay space. The observed spatial growth rate is low.
- The potential benefit of overlay routing for a pair of nodes ij and the utility of the pair for overlay routing can be predicted by the clusters where i and j belong to. The distributions of triangle inequality violation ratios are independent of the sample size.
- Less than one percent of all edges account for the gains in overlay routing delay.
- None of the above characteristics of the delay space are accurately reflected in existing Internet topology models.

4. MODELING THE INTERNET DELAY SPACE

Using measured Internet delay spaces to drive distributed system simulations allows system designers to evaluate their solutions under the most realistic delay space characteristics. However, there are two potential concerns. First of all, measuring Internet delay spaces is a time consuming and difficult process. Due to network outages and measurement errors, there are usually a significant number of missing measurements. These missing values make it impossible to simulate the delay characteristics of the network completely. The second potential concern is that the $O(N^2)$ storage requirement of the matrix representation does not scale gracefully.

To address these concerns, we develop techniques to model a measured Internet delay space. This model can sensibly interpolate the missing measurements, and its storage overhead is only $O(N)$.

4.1 Building Block Techniques

Technique 1: Low-dimensional Euclidean embedding - The first technique we use is to model an Internet delay space using a low-dimensional Euclidean embedding. That is, we compute Euclidean coordinates for each node and use Euclidean distances to approximate the delays in the delay space. Such an Euclidean embedding has a scalable $O(N)$ representation. Moreover, previous studies have shown that an Internet delay space can be well approximated by a Euclidean embedding with as little as 5 dimensions, and several techniques exist to compute such an embedding robustly even when some missing measurements are present in the measured delay space [17, 7, 27, 6, 14, 34, 31, 30]. In this paper, we adopt a slightly modified version of the GNP [17] method to compute the embedding. This modified version simply ignores the missing measurements in the delay space matrix when computing node coordinates. GNP uses Landmarks as global reference points to compute coordinates. We

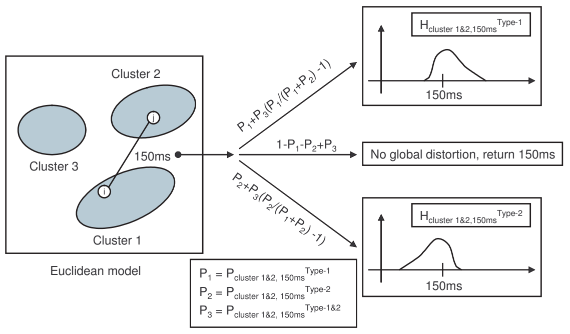

Cluster 2
Cluster 3
Cluster 1
150ms
j
i
Euclidean model
$P_1+P_3(P_1/(P_1+P_2)-1)$
$1-P_1-P_2+P_3$
$P_2+P_3(P_2/(P_1+P_2)-1)$
$H_{\text{cluster 1\&2,150ms}}^{\text{Type-1}}$
$H_{\text{cluster 1\&2,150ms}}^{\text{Type-2}}$
150ms
No global distortion, return 150ms
$P_1 = P_{\text{cluster 1\&2, 150ms}}^{\text{Type-1}}$
$P_2 = P_{\text{cluster 1\&2, 150ms}}^{\text{Type-2}}$
$P_3 = P_{\text{cluster 1\&2, 150ms}}^{\text{Type-1\&2}}$

Figure 12: Global distortion technique.

observe that as long as enough random landmarks are chosen, the accuracy of the embedding is not affected by the missing measurements.

As we will show in Section 4.3, a 5D Euclidean embedding of the measured Internet delay space can preserve some of the key metrics quite well. In the 5D Euclidean map, the overall delay distribution, the global clustering properties, and the growth metrics are very well preserved. Moreover, the accuracy of the 5D Euclidean map is good enough to allow a reasonable estimation of missing measurements. For Real Data 1, the 5D Euclidean map can predict the measured delays to within 50% error for 93.08% of the data. We expect this level of accuracy to hold also for the missing data values. However, a 5D Euclidean map fails to preserve triangle inequality violations, routing inefficiency, and local clustering properties.

Obviously, Euclidean embedding cannot preserve triangle inequality violations, thus no routing inefficiency. The Euclidean map also fails to preserve nodes with high in-degree in the nearest neighbor directed graph. This is because a node cannot have a high number of nearest neighbors in a low dimensional Euclidean metric space. Specifically, the maximal in-degree is 26 for the real data and only 9 for the 5D map. Moreover, in the 5D map, there are much fewer nodes with in-degree 0 than the real data. For comparison, we generate random points in a 10D space and the maximal in-degree achieved is only 6. Thus, even in a higher dimensional space, high in-degree is not something that occurs naturally.

To address these limitations of a basic 5D Euclidean model, we use two additional techniques in order to reconstruct the properties lost as a result of the Euclidean embedding.

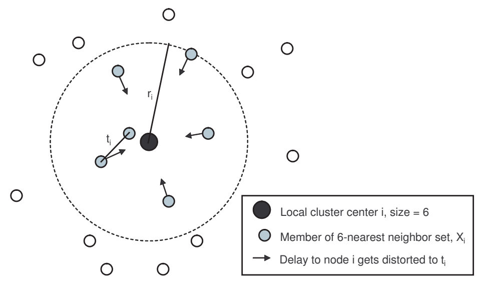
Technique 2: Global distortion - The basic technique to create triangle inequality violations in the 5D Euclidean model is to distort the delays computed in the 5D embedding as part of a post-processing step. Since the frequency of triangle inequality violations in the measured data is relatively small, it suffices to distort only a small subset of node pairs or edges. The key questions to decide are (1) how many edges to distort, (2) what distortion distribution to use, and (3) how to ensure the distortion to an edge is deterministic and repeatable. The last property ensures that the model always produces the same delay for a given pair of nodes.

The key idea is to identify the edges in the measured data that cause violations above a certain severity threshold, characterize the distortion distribution for these edges when they are mapped into the 5D Euclidean model, then use this same distortion distribution to introduce distortions when delays are generated from the 5D embedding. The process is made deterministic by using the node identifier to generate pseudo-random numbers. By choosing different severity thresholds, we can vary the number of edges that get distorted in the model and experimentally determine the threshold that best matches the empirical data. The technique is illustrated in Figure 12.

We define a violation severity threshold R. A violation caused by an edge ij is severe if for some node k, $\frac{d_{ik}+d_{kj}}{d_{ij}} < R$ (called Type 1 violation), or if $\frac{|d_{ik}-d_{kj}|}{d_{ij}} > \frac{1}{R}$ (called Type 2 violation). We then compute the distortion distribution for these special edges and store it along with the coordinated of the 5D embedding. The distribution is then used to introduce distortions when generating delays from the 5D embedding. Since the violation characteristics vary dramatically across different cluster-to-cluster groups (as discussed in Section 3.3), we compute and store the distortion distribution separately for each cluster-to-cluster group.

For each cluster-to-cluster group g, all edges with the same 5D Euclidean model delay l (rounded down to the nearest 1ms) form a subgroup. For each subgroup (g, l), we compute the fraction of edges in this subgroup that are involved in severe Type 1 violations in the real data, $P_{g,l}^{Type-1}$, and a histogram $H_{g,l}^{Type-1}$ to characterize the real delay distribution of those severe violation edges. Similarly, for Type 2 violations, we compute the fraction $P_{g,l}^{Type-2}$ and the histogram $H_{g,l}^{Type-2}$. We also compute the fraction of edges that incur severe Type 1 and Type 2 violations simultaneously, $P_{g,l}^{Type-1\&2}$. This extra statistical information incurs an additional constant storage overhead for the model.

With these statistics, the model delay with global distortion between node i and j is then computed as follows. Draw a pseudo-random number ρ in [0,1] based on the IDs of i and j. Let the Euclidean distance between i and j be l_{ij} and the cluster-cluster group be g. Based on $P_{g,l_{ij}}^{Type-1}$, $P_{g,l_{ij}}^{Type-2}$, $P_{g,l_{ij}}^{Type-1\&2}$, and using ρ as a random variable, decide whether the edge ij should be treated as a severe Type 1 violation (with probability $P_{g,l_{ij}}^{Type-1} + P_{g,l_{ij}}^{Type-1\&2}$.



r_i
t_i
Local cluster center i, size = 6
Member of 6-nearest neighbor set, X_i
Delay to node i gets distorted to t_i

Figure 13: Local distortion technique.

$(\frac{P^{Type-1}_{g,l_{ij}}}{P^{Type-1}_{g,l_{ij}}+P^{Type-2}_{g,l_{ij}}} - 1))$, or a severe Type 2 violation (with probability $P^{Type-2}_{g,l_{ij}} + P^{Type-1\&2}_{g,l_{ij}} \cdot (\frac{P^{Type-2}_{g,l_{ij}}}{P^{Type-1}_{g,l_{ij}}+P^{Type-2}_{g,l_{ij}}} - 1))$, or to return the value l_{ij} without distortion. If the edge ij is treated as a severe Type 1 violation, then we use the histogram $H^{Type-1}_{g,l_{ij}}$ and ρ to draw a value from the histogram and return that value. Similarly, if the edge is treated as a severe Type 2 violation, then we use the histogram $H^{Type-2}_{g,D_{ij}}$ instead.

By experimenting with different threshold R, we have determined that a value of 0.7 produces Type 1 and Type 2 violation distributions similar to those observed in the real data. This is also the threshold we use for the remainder of this paper.

Technique 3: Local distortion - To preserve the local clustering in the measured data, we introduce additional local distortion. The principal idea is to pull some nodes within a radius around a local cluster center closer to create the needed in-degree, as illustrated in Figure 13. From the nearest neighbor directed graph analysis on the measured data, we identify local cluster centers and note their sizes. Suppose a local cluster center node i has a cluster size of s_i in the original data. We identify the set of its s_i nearest neighbors, X_i, in the model after global distortion. Then, we compute a radius r_i as $max_{j \in X_i}(d_{ij})$, and a threshold t_i as $min_{j,k \in X_i}(d_{jk}) - \epsilon$. Currently, ϵ is set to $0.01 \cdot min_{j,k \in X_i}(d_{jk})$. Then we associate the values r_i and t_i with node i in the model. r_i is essentially the radius within which distortion may be necessary. t_i is the delay needed to beat the smallest delay among the nodes in X_i. This additional information adds a constant storage overhead.

The model delay with local distortion between node i and j is then computed as follows. Suppose the delay for the edge ij after global distortion is l_{ij}. If neither i nor j is a local cluster center, l_{ij} is returned. Suppose i is a local cluster center and j is not, then if $l_{ij} \leq r_i$, we return $min(t_i, l_{ij})$; otherwise, we return l_{ij}. The t_i threshold is used to ensure that the nodes in X_i cannot choose one another as their nearest neighbors. After the distortion, they will choose i as their nearest neighbor unless there is a closer node outside of the radius r_i. If both i and j are local cluster centers, we pick the one with the smaller node identifier as the center and perform the above steps.

4.2 Modeling Framework

Based on the basic techniques described above, the overall framework for modeling a measured Internet delay space is as follows:

Step 1. Perform global clustering on the data to assign nodes to major clusters. Perform nearest neighbor directed graph analysis to identify local cluster centers and their sizes.

Step 2. Compute a 5D Euclidean embedding of the data using a robust method. In this paper, we use a slightly modified version of GNP.

Step 3. For each cluster-cluster group g and Euclidean model delay l, compute the global distortion statistics $P^{Type-1}_{g,l}$, $P^{Type-2}_{g,l}$, $P^{Type-1\&2}_{g,l}$, $H^{Type-1}_{g,l}$, $H^{Type-2}_{g,l}$ using a severe violation threshold R. For each local cluster center i, compute the local distortion statistics r_i and t_i.

Step 4. At this point, the original measured data is no longer needed. To compute the model delay between node i and j, first compute the Euclidean model delay, then apply global distortion if necessary, and finally apply local distortion if necessary. Return final value.

The total storage overhead of the model is $O(N)$ and calculating the delay of an edge at run time is simple.

4.3 Evaluating the Model

We evaluate the effectiveness of our modeling framework by comparing the properties found in Real Data 1 against properties in the resulting model of Real Data 1. Figure 14 presents our results.

Overall, we can see that all the important trends are preserved by the model very well. As expected, there are some small discrepancies. We believe the primary reason for the discrepancies is a slight shift in the overall delay distribution in the 5D Euclidean map. This can be seen in Figure 14(a), as there are fewer long delays and more short delays in the model than in the real data. The impact of this shift is observed in the shifted cluster merging points in Figure 14(b), the higher initial ball growth in Figure 14(d), and the slight shift downward in $D(k)/D(N)$ in Figure 14(e). The nearest neighbor graph in-degree distribution is nearly identical to that in the real data (see Figure 14(c)). Unlike a simple 5D map, the number of low in-degree nodes in the model matches well against the real data. The maximal in-degree achieved in the model is however only 22 compared to 26 in Real Data 1. This variation is not surprising given that the maximal in-degree is highly sensitive to the exact values

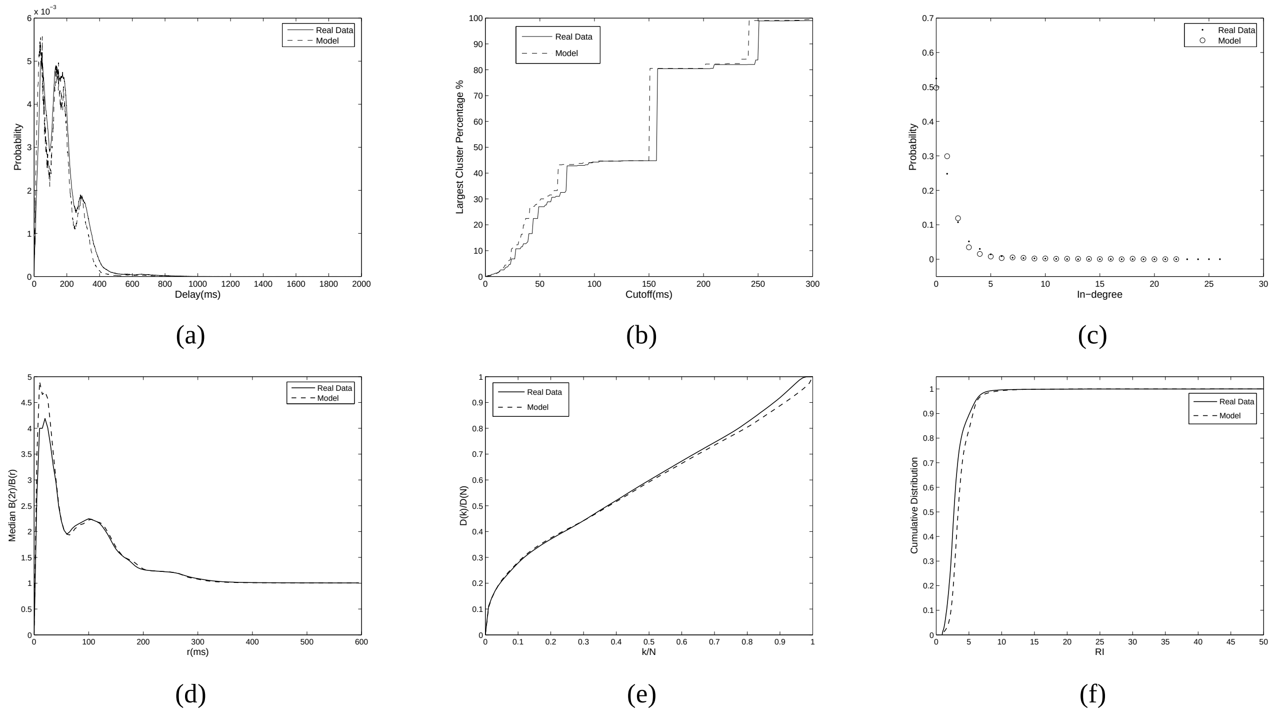


Figure 14: Model vs Real Data 1. (a) Delay distribution. (b) Clustering cutoff. (c) In-degree distribution. (d) Median B(2r)/B(r). (e) D(k)/D(N). (f) Routing inefficiency distribution.

of delays. Finally, observe in Figure 14(f) that the routing inefficiency distributions are also similar. We continue to investigate different optimization settings in GNP to try to eliminate the delay shift in the 5D map.

In summary, the model is highly successful at capturing all the important properties in the measured Internet delay space.

5. SYNTHESIS OF INTERNET DELAY SPACE

System designers wish to evaluate distributed systems at large scale. As we have shown, using either measured delay data or our Internet delay space model to drive simulations brings substantial improvements in realism over topology-models. However, the achievable scale is still limited by the difficulty of measuring and storing empirical delay data from more than, say, tens of thousands of nodes. Moreover, system designers often wish to vary certain properties of the delay space in a principled manner, in order to test the robustness of their designs to such variations. It is difficult to do this with measured delay data.

In this section, we build upon our empirical understanding of the Internet delay space and our delay space modeling techniques and study additional techniques to enable artificial *synthesis* of a realistic delay space based on the actual properties of a measured Internet delay space. The goals are to (1) allow synthesis of delay spaces at scales that exceed our capability to measure real data and (2) provide methods to vary the properties of the synthesized delay space in a principled manner.

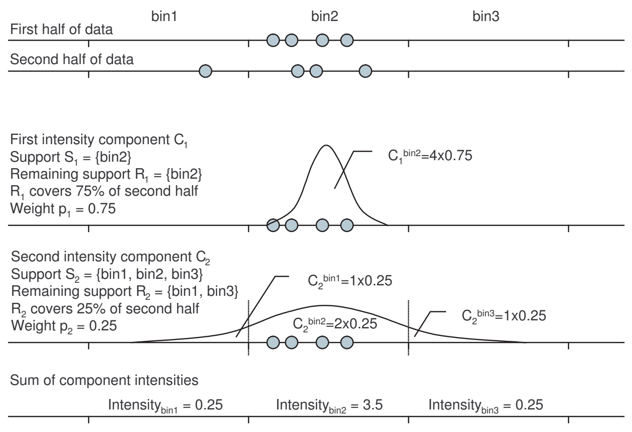

bin1
bin2
bin3
First half of data
Second half of data
First intensity component C1
Support S1 = {bin2}
Remaining support R1 = {bin2}
R1 covers 75% of second half
Weight p1 = 0.75
C1bin2=4x0.75
Second intensity component C2
Support S2 = {bin1, bin2, bin3}
Remaining support R2 = {bin1, bin3}
R2 covers 25% of second half
Weight p2 = 0.25
C2bin1=1x0.25
C2bin2=2x0.25
C2bin3=1x0.25
Sum of component intensities
Intensitybin1 = 0.25
Intensitybin2 = 3.5
Intensitybin3 = 0.25

Figure 15: Computing intensities in Euclidean map synthesis technique.

5.1 Building Block Techniques

The new techniques introduced in this section exploit the scaling properties found in the measured Internet delay space to enable accurate extrapolation to a larger delay space.

Technique 4: Euclidean map synthesis - Given a 5D Euclidean map of an Internet delay space, we seek to capture its locality and growth characteristics so that we can synthesize an artificial map based on these characteristics and create realistic structure in the synthesized delay space.

A simple idea is to divide the Euclidean space into equal sized hyper-cubes, count the number of points in each hyper-cube, and use these counts as relative intensities. With appropriate scaling of the relative intensities, one can synthesize an artificial map of a certain size by generating random points in each hyper-cube according to the intensities using an inhomogeneous Poisson point process [15, 23][1]. Indeed, this simple method can mimic the point distribution of the original map and generate realistic overall delay distribution and global clustering structure. However, this method ignores the growth characteristics in the data. As a result, synthetic points can only appear in hyper-cubes where points were originally found.

To incorporate growth characteristics, the key idea is to introduce uncertainties in the locations of each point and compute intensities that predict growth. The idea is best explained with a simple example illustrated in Figure 15. In the example, there are 8 points in a 1-dimensional Euclidean space divided into 3 equal size bins. We randomly divide the points into two halves, the first half happens to lie in bin2, while the other half is spread across bin1 and bin2. We will iteratively compute the i^{th} *intensity component* C_i, which is a vector of intensities for the bins, to predict the growth observed in the second half. The location uncertainty of a point in the first half is represented by a Gaussian probability distribution with a certain variance or width. To compute the first intensity component C_1, we place a Gaussian with a small *width* w_1 that represents a low level of uncertainty in the center of each bin and scale it by the number of first half points in the bin. As a result, the 99% bodies of the Gaussians lie within bin2. We call the bins occupied by the 99% bodies of the Gaussians the *support* of the first component, S_1. We also defined the *remaining support* of a component to be the support of the current component subtracted by the support of the previous component, i.e. $R_i = S_i \backslash S_{i-1}$. Since this is the first component, R_1 is simply S_1.

The *intensity* I_1 generated by the Gaussians is spread in the 3 bins as 0, 4, 0 respectively. Now we ask, how well does R_1 cover the second half of the points? If all points in the second half are covered by R_1 then I_1 can account for the growth in the second half and we are done. However, in the example, R_1 is only covering 75% of the points in the second half. As a result, we weight the intensity I_1 by a factor $p_1 = 0.75$ to obtain the intensity component C_1. Since we have not completely accounted for the growth in the second half, we need to increase the location uncertainty and compute the second intensity component C_2. To do so, we use a wider Gaussian (width w_2) for the second iteration. The aggregate intensity is still 4, but this time, it is spread across all 3 bins. Suppose the intensities generated in the 3 bins are 1, 2, 1 respectively. The 99% body of these wider Gaussians occupy all three bins, thus the support of the second component S_2 is the set $\{$bin1, bin2, bin3$\}$. The remaining support R_2 is $S_2 \backslash S_1$, i.e. $\{$bin1, bin3$\}$. The fraction of the second half covered by R_2 is 25%. Thus, the intensity I_2 is weighted by $p_2 = 0.25$ to obtain C_2.

[1]The number of points lying in any two disjoint sets in space are independent random numbers distributed according to a Poisson law with mean given by the intensity.

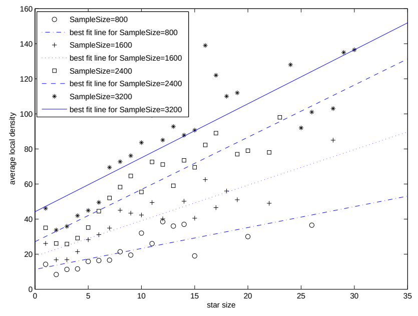


Figure 16: Average local density vs local cluster (star) size for different data sample sizes.

This iterative process continues until either all points in the second half are covered by R_i, or when a maximum Gaussian width has been reached. The intensity of each bin is simply the sum of all the intensity components C_i. Finally, we repeat the procedure to use the second half to predict the growth in the first half and use the average intensity of each bin as the final intensity. In practice, we divide the 5D space into 100 bins in each dimension and vary the Gaussian variance or width from one-tenth to ten times the bin width. Moreover, we ignore bins with intensity smaller than a threshold to reduce the memory requirements.

Technique 5: Local cluster size estimation - In Section 3, we have shown that, for the data sizes that we have studied, nearly all properties of the Internet delay space we consider are invariant under scaling. The exception is that the local cluster sizes grow with scale. It also turns out that the maximal local cluster size grows linearly with the delay space sample size. With this linear relationship, it is easy to estimate the appropriate maximum local cluster size at a particular scale. What remains unclear is how to assign different cluster sizes to local cluster centers.

For this, we exploit another relationship that we have discovered: For each sample size, the size of a local cluster is linearly related to the local node density (i.e., the number of nodes within 15ms) around the cluster centers. Figure 16 plots the average local density versus local cluster size (or star size) for different sample sizes. Exploiting this relationship, our technique works as follows. First, select the maximum local cluster size based on the scale. The minimum local cluster size is always 1. Assume that we have already identified a subset of nodes to act as local cluster centers. We compute the local node densities around the cluster center. Then, using the maximal and minimal densities, maximal and minimal cluster sizes, and assuming a linear relationship, we can easily assign a local cluster size to each center.

5.2 Synthesis Framework

Based on the basic techniques described above and in Section 4.1, the overall framework for synthesizing a realistic delay space is as follows:

Step 1. Perform global clustering on the data to assign nodes to major clusters. Perform nearest neighbor directed graph analysis to identify local cluster centers.

Step 2. Compute a 5D Euclidean embedding of the data using a robust method. In this paper, we use a slightly modified version of GNP.

Step 3. For each cluster-cluster group g and Euclidean delay l, compute the global distortion statistics $P_{g,l}^{Type-1}$, $P_{g,l}^{Type-2}$, $P_{g,l}^{Type-1\&2}$, $H_{g,l}^{Type-1}$, $H_{g,l}^{Type-2}$ using a severe violation threshold R.

Step 4. At this point, the original measured data is no longer needed. Split the 5D Euclidean map into two, one containing only local cluster centers, and one containing all other nodes. Based on these two maps, separately synthesize Euclidean maps of local cluster centers and non-centers to the appropriate scale using the Euclidean map synthesis technique. Recall that the ratio between these two types of nodes is invariant. Merge the two resulting synthesized maps back into one synthesized map.

Step 5. Perform global clustering on synthesized map to assign nodes to major clusters.

Step 6. Assign a local cluster size to each synthesized center using the local cluster size estimation technique. For each local cluster center i, compute the local distortion statistics r_i and t_i.

Step 7. To compute the synthesized delay between node i and j, first compute the Euclidean delay. Apply global distortion, if necessary, according to the statistics from the real data, and finally apply local distortion if necessary. Return final value.

The above framework can be tuned to synthesize delay spaces with varying properties. For example, the characteristics of the local clusters, the severity of triangle inequality violations, and even the spatial distribution of nodes can all be adjusted in a well controlled manner. System designers can take advantage of this flexibility to evaluate their solutions under different delay space properties.

5.3 Evaluating Synthesized Delay Data

To evaluate the effectiveness of our synthesis framework, we compare a synthesized delay space directly against a measured delay space. To do so, we first extract a 2000 node random sub-sample from Real Data 1. Then, we apply our synthesis framework on just this 2000 node sub-sample to synthesize a delay space with 4150 nodes. If the synthesis framework correctly predicts and preserves the scaling trends, then the synthetic 4150 nodes delay space should have properties very similar to those found in Real Data 1 which has 3997 nodes.

The comparison results are displayed in Figure 17. As can be seen, even though the synthesis is based on a limited subset of data, the synthesis framework is highly successful at predicting the properties in the 3997 node measured data. All the important trends are preserved very well, although there are still a few Small differences. For example,the 5D map causes a slight shift in delay distribution towards smaller delays. This results in the faster initial ball growth as seen in Figure 17(d), and the slight shift downward in the $D(k)/D(N)$ metric (Figure 17(e)).

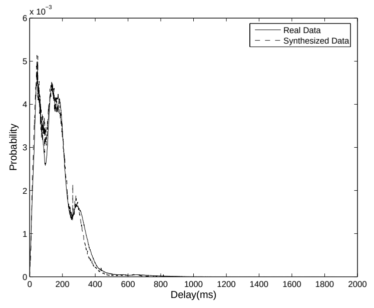
In summary, the synthesis framework is highly effective in creating realistic delay spaces. Our ultimate goal is to validate the synthesis framework against much larger data sets, and we continue to work towards this goal by collecting more measurements in order to gain higher confidence about the delay space scaling properties.

6. RELATED WORK

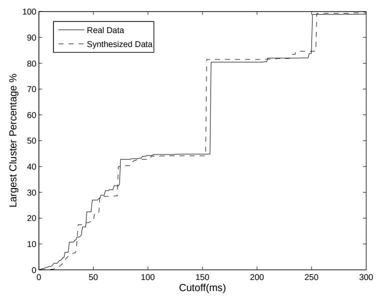
Our work on modeling the Internet delay space is complementary to existing work on modeling network connectivity topologies. There is an opportunity for future work to incorporate delay space characteristics into topology models.

Early artificial network topologies had a straight-forward connectivity structure such as tree, star, or ring. A more sophisticated topology model that constructs node connectivity based on the random graph model was proposed by Waxman [36]. However, as the hierarchical nature of the Internet connectivity became apparent, solutions that more accurately model this hierarchy, such as Transit-Stub by Calvert *et al* [38] and Tier by Doar [8], emerged. Faloutsos *et al* [9] studied real Internet topology traces and discovered the power-law node degree distribution of the Internet. Li *et al* [13] further showed that router capacity constraints can be integrated with the power-law node degree model to create even more realistic router-level topologies.

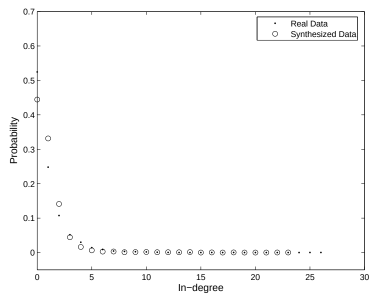
There are many on-going projects actively collecting delay measurements of the Internet, including Skitter [29], AMP [2], PingER [20], and Surveyor [33] to name just a few examples. Some of these projects also collect one-way delays



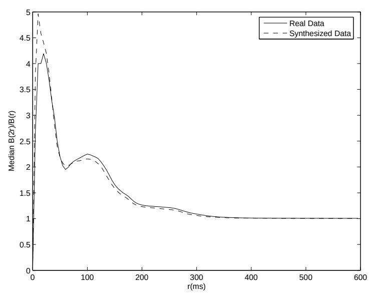
(a)



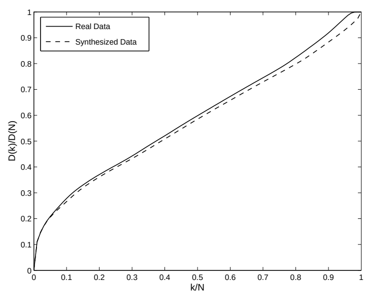
(b)



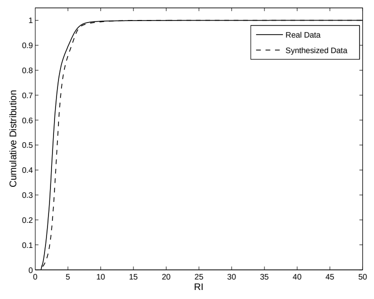
(c)



(d)



(e)



(f)

Figure 17: Synthesized vs Real Data 1. (a) Delay distribution. (b) Clustering cutoff. (c) In-degree distribution. (d) Median B(2r)/B(r). (e) D(k)/D(N). (f) Routing inefficiency distribution.

and hop-by-hop routing information. These projects typically use a set of monitoring nodes, ranging roughly from 20 to 100, to actively probe a set of destinations. The Skitter work probes on the order of 1 million destinations, which is the largest among these projects. The active monitoring method can probe any destination in the network, but the resulting measurements cover only a small subset of the delay space as observed by the monitors. Many of these measurements are also continuously collected, allowing the study of changes in delay over time. Our work uses the King tool to collect delay measurements, which restricts the probed nodes to be DNS servers, but produces a symmetric delay space matrix, which lends itself to a study of the stationary delay space characteristics.

Some of the delay space properties that we report in this paper have been observed in previous work. For example, triangle inequality violations and routing inefficiencies have been observed in [26] and [17]. Some of the characteristics of delay distributions and their implications for global clustering have been observed in Skitter. However, many of the observations made in this paper are new. These include the local clustering properties, and in particular the approximately exponential in-degree distribution, spatial growth properties, detailed properties of triangle inequality violations of different types and across different clusters, and the examination of these properties under scaling. In addition to the ”static” properties of delay, previous work have also studied the temporal properties of Internet delay [1]. Incorporating temporal properties into a delay space model is an area for future work.

One key technique used in our work is computing a low dimensional Euclidean embedding of the delay space to enhance the completeness and scalability of the delay space representation. Many approaches for computing such an embedding have been studied [17, 7, 27, 6, 14, 34, 28, 19]. We have not considered the impact of using different computation methods or using different embedding objective functions. This represents another area for potential future work.

7. CONCLUSIONS

To the best of our knowledge, this is the first study to systematically analyze, model, and synthesize realistic Internet delay spaces. We make two primary contributions in this work. First, we quantify the properties of the Internet delay space with respect to a set of metrics relevant to distributed systems design. This leads to new fundamental understandings of the Internet delay space characteristics, which may inform future work. Second, we develop a set of building block techniques to model and synthesize the Internet delay space compactly while accurately preserving all relevant metrics.

There are many interesting areas for future work. For example, how can the edge network delay space characteristics be integrated with a connectivity topology model so that we can have realistic hop-by-hop delay characteristics? Secondly, how can we extend the edge network delay space model to incorporate hosts within each edge network? Finally, how can the models be extended to handle extremely dense networks that are hard to embed accurately in a low dimensional Euclidean space? Answering these questions should generate new fundamental insights.

8. REFERENCES

[1] A. Acharya and J. Saltz. A Study of Internet Round-Trip Delay. Technical Report CS-TR-3736, University of Maryland, College Park, 1996.

[2] Active measurement project, NLANR. http://watt.nlanr.net.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. The Case for Resilient Overlay Networks. In *Proceeedings of HotOS VIII*, May 2001.

[4] Miguel Castro, Peter Druschel, Y. Charlie Hu, and Antony Rowstron. Exploiting network proximity in peer-to-peer overlay networks. Technical Report MSR-TR-2002-82, Microsoft Research, May 2002.

[5] Miguel Castro, Peter Druschel, Y. Charlie Hu, and Antony Rowstron. Proximity neighbor selection in tree-based structured peer-to-peer overlays. Technical Report MSR-TR-2003-52, Microsoft Research, June 2003.

[6] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet coordinates for distance estimation. Technical Report MSR-TR-2003-53, Microsoft Research, September 2003.

[7] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceeding of ACM SIGCOMM*, August 2004.

[8] M. Doar. A better model for generating test networks. In *Proceeding of IEEE GLOBECOM*, November 1996.

[9] C. Faloutsos, M. Faloutsos, and P. Faloutsos. On Power-law Relationships of the Internet Topology. In *Proceedings of ACM Sigcomm*, August 1999.

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

[11] Krishna P. Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proc. ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[12] David R. Karger and Matthias Ruhl. Finding nearest neighbors in growth restricted metrics. In *Proccedings of ACM Symposium on Theory of Computing*, 2002.

[13] L. Li, D. Alderson, W. Willinger, and J. Doyle. A first-principles approach to understanding the internet's router-level topology. In *Proceeding of ACM SIGCOMM*, August 2004.

[14] H. Lim, J. Hou, and C.-H. Choi. Constructing internet coordinate system based on delay measurement. In *Proceedings of Internet Measurement Conference*, Miami, FL, October 2003.

[15] Jesper Møller and Rasmus Waagepetersen. *Statistical Inference and Simulation for Spatial Point Processes*. Chapman and Hall/CRC, 2004.

[16] D. Moore, R. Periakaruppan, and J. Donohoe. Where in the World is netgeo.caida.org. In *Proceedings of the Internet Society Conference (INET)*, 2000.

[17] T. S. E. Ng and H. Zhang. Predicting Internet networking distance with coordinates-based approaches. In *Proceedings of IEEE INFOCOM*, June 2002.

[18] p2psim. http://www.pdos.lcs.mit.edu/p2psim/.

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

[20] PingER. http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html.

[21] S. Ranjan, R. Karrer, and E. Knightly. Wide area redirection of dynamic content by internet data centers. In *Proceedings of IEEE INFOCOM*, Hong Kong, China, 2004.

[22] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. In *Proc. 1st Int'l Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, March 2002.

[23] Rolf-Dieter Reiss. *A Course on Point Processes*. Springer Series in Statistics. Springer, 1993.

[24] Route views. http://www.routeviews.org/.

[25] B. Bhattacharjee S. Banerjee and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM*, August 2002.

[26] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-end Effects of Internet Path Selection. In *Proceedings of ACM Sigcomm*, August 1999.

[27] Y. Shavitt and T. Tankel. Big-bang simulation for embedding network distances in euclidean space. In *Proceedings of IEEE INFOCOM*, San Francisco, CA, March 2003.

[28] Y. Shavitt and T. Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *Proceedings of IEEE INFOCOM*, April 2004.

[29] Skitter. http://www.caida.org/tools/measurement/skitter/.

[30] A. Slivkins. Distributed Approaches to Triangulation and Embedding. In *Proceedings 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[31] A. Slivkins, J. Kleinberg, and T. Wexler. Triangulation and Embedding using Small Sets of Beacons. In *Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.

[32] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM*, 2001.

[33] Surveyor. http://www.advanced.org/csg-ippm/.

[34] L. Tang and M. Crovella. Virtual landmarks for the internet. In *Proceedings of Internet Measurement Conference*, Miami, FL, October 2003.

[35] Marcel Waldvogel and Roberto Rinaldi. Efficient Topology-Aware Overlay Network. In *First Workshop on Hot Topics in networks (HotNets-I)*, October 2002.

[36] B. Waxman. Routing of multipoint connections. *IEEE J. Select. Areas Commun.*, December 1988.

[37] J. Winick and S. Jamin. Inet-3.0: Internet topology generator. Technical Report UM-CSE-TR-456-02, University of Michigan, 2002.

[38] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of IEEE INFOCOM*, March 1996.

[39] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for wide-area fault-tolerant location and routing. *U.C. Berkeley Technical Report UCB//CSD-01-1141*, 2001.