Scalable Cooperative Latency Estimation

Michal Szymaniak   Guillaume Pierre   Maarten van Steen
Vrije Universiteit Amsterdam
Department of Computer Science
De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
{michal,gpierre,steen}@cs.vu.nl

Abstract

This paper discusses SCoLE, a scalable system to estimate Internet latencies. SCoLE is based on GNP, which models Internet latencies in an N-dimensional Euclidean space. In contrast to GNP and other GNP-based systems, however, SCoLE does not employ any global space whose parameters must typically be negotiated by the participating hosts. Instead, it allows each host to construct its "private" space and model inter-host latencies in that space. The private space parameters as well as the modeling algorithm can be adjusted on a per-host basis, which improves system flexibility. More importantly, the mutual independence of private spaces results in higher SCoLE scalability, which is bound neither by the global negotiation of space parameters nor by global knowledge of any kind. We show that latency estimates performed in different private spaces are highly correlated. This allows SCoLE to be used in large-scale applications where consistent latency estimates need to be performed simultaneously by many independent hosts.

1 Introduction

Contemporary distributed systems often need to know latencies between their member hosts. Given latency information, a system is able to construct efficient data dissemination trees, select optimal locations for its various services, or redirect clients to a service instance close to them.

Latency discovery becomes difficult in systems that span a wide-area network or contain many hosts. Measuring latencies between each possible pair of hosts is infeasible. Such a "greedy" measurement forces *all* the hosts to probe each other to collect the latency information. This is clearly impossible in systems connecting millions of hosts, such as peer-to-peer applications.

A promising approach to the latency estimation problem has been presented in GNP, which models Internet latencies in an N-dimensional geometric space [10]. GNP first selects a set of $N+1$ global reference hosts, called landmarks. Then, it measures latencies between each pair of them and assigns coordinates to the landmarks based on these latencies. Provided with the landmark coordinates, any host can compute its own coordinates based on its latencies to each of the landmarks. Given any two hosts, GNP approximates their latency with the Euclidean distance between their corresponding coordinates in the N-dimensional space. Assuming that M is the number of hosts in the system, this approach requires only $O(N \cdot M)$ measurements, instead of $O(M^2)$ for the brute-force approach. A typical value for N is 6. Note that the idea of GNP can be implemented with many different positioning algorithms using various methods to determine host coordinates.

GNP requires that all hosts use the same global system parameters, such as the space dimension, the landmark set, and the positioning algorithm. However, in large-scale applications such as peer-to-peer systems, latencies often need to be estimated by many hosts. Selecting an appropriate set of system parameters that meets the needs of every host may become hard, since different hosts have different local resources, network connections, and expectations with respect to the estimation accuracy. These differences may require that each host uses a different set of space parameters (dimensions, landmark set), as well as other parameters such as the frequency at which latencies should be re-measured and hosts re-positioned.

This paper discusses SCoLE, a positioning system that allows each host to select its own parameters for the positioning process. In SCoLE, each host chooses its preferred space parameters, constructs its own "private" space, and positions other hosts in that space. We show that SCoLE respects application correctness: even though multiple members of a given application may derive their estimations from private spaces with different parameters, the estimated latencies are highly correlated among all spaces. This allows applications to ignore the fact that estimations are made autonomously at different locations, and consider the set of independent SCoLE instances as a single latency prediction service. Moreover, decentralization improves scalability, as it eliminates the need for global negotiation of parameters and for global knowledge of these parameters. Independent per-host latency estimation and complete decentralization are unique features of SCoLE that are not provided by any existing system.

The rest of this paper is structured as follows. Section 2 discusses relevant research efforts. Section 3 presents the concept of our decentralized positioning scheme. For the sake of clarity, this section discusses the scheme in its most elementary form, which is then evaluated in Section 4. Section 5 shows how the elementary positioning scheme can be optimized for efficient implementation in a real-world system. Finally, Section 5 concludes.

2 Background

2.1 GNP

The idea of modeling the Internet as an N-dimensional space has been introduced in GNP [10]. GNP approximates the latency between any pair of hosts as the Euclidean distance between their corresponding N-dimensional coordinates.

GNP relies on the assumption that latencies can be triangulated in the Internet. The coordinates of any host X are computed based on the measured latencies between X and k "landmark" hosts, whose coordinates have been computed earlier. By treating these latencies as distances, GNP triangulates the coordinates of X. The number of landmarks k must be at least $N + 1$ to ensure unambiguous positioning in an N-dimensional space.

GNP works in two phases. In the first phase, the coordinates of the "landmark" hosts are computed. In the second phase, in turn, any host may determine its own coordinates based on its measured latencies to all the landmarks.

Landmarks are positioned as follows. First, they measure their latencies to each other, and report these $k(k-1)$ latencies to a selected host. That host calculates all the landmark coordinates so that the measured latency between any pair of the landmarks is equal to the Euclidean distance between their coordinates. In an ideal case, all measured distances would form a consistent set that fits in a N-dimensional space. We could then position the first landmark anywhere in this space, and subsequently position the other landmarks, one by one, based on their distances to the landmarks positioned so far. For example, in Figure 1, a set of three landmarks has been placed in a 2-dimensional Euclidean space (in fact, only *relative* coordinates of landmarks are important, as the L1-L2-L3 triangle can be arbitrarily translated or rotated; this leads to an infinite number of correct sets of *absolute* landmark coordinates).

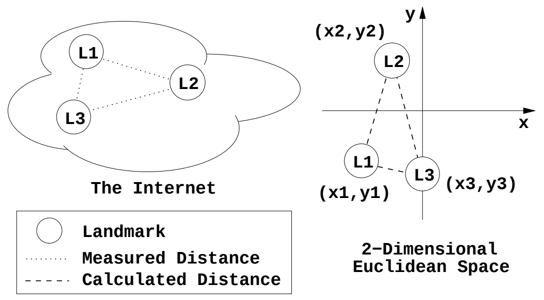

Figure 1. GNP Landmark Positioning

However, since the Internet is not a perfect N-dimensional space, the $k(k-1)$ latencies between hosts seldom match exactly their corresponding distances among k points in an N-dimensional space, whatever the mapping between the hosts and the points. In other words, measured latencies are usually not consistent enough with each other to directly treat them as distances in an N-dimensional space for any N. GNP addresses this issue by searching the set of landmark coordinates that minimizes the total discrepancy *TD* between the measured latencies and the corresponding Euclidean distances. The total discrepancy is measured using the following function:

$$TD(L_1, \ldots, L_k) = \sum_{i \geq 1}^{k} \sum_{j > i}^{k} \varepsilon(d_{L_iL_j}, d^*_{L_iL_j})$$

where $d_{L_iL_j}$ and $d^*_{L_iL_j}$ respectively denote the measured latency between hosts L_i and L_j and the corresponding Euclidean distance, and $\varepsilon(.)$ denotes a classical error function:

$$\varepsilon(d_{L_iL_j}, d^*_{L_iL_j}) = \left(d_{L_iL_j} - d^*_{L_iL_j}\right)^2$$

In this way, GNP reduces the problem of landmark coordinate determination to finding the coordinates of k landmarks that minimize the function TD. GNP solves this multi-variable minimization problem by means of the popular Simplex-downhill algorithm [9].

Once the landmark coordinates are known, any host may determine its own coordinates by simply measuring its latency to each of the k landmarks. Similar to the landmark positioning problem, this operation would be trivial in an ideal case, in which latency measurements are infinitely accurate and always adhere to triangulation, so the set of k distances precisely determines the only possible host coordinates. However, since this assumption is generally wrong, GNP again employs the Simplex-downhill algorithm to approximate the host coordinates. The only difference is that it minimizes the sum of k errors this time to determine the coordinates of only one host H (see Figure 2):

$$F(H) = \sum_{i=1}^{k} \varepsilon(d_{HL_i}, d^*_{HL_i})$$

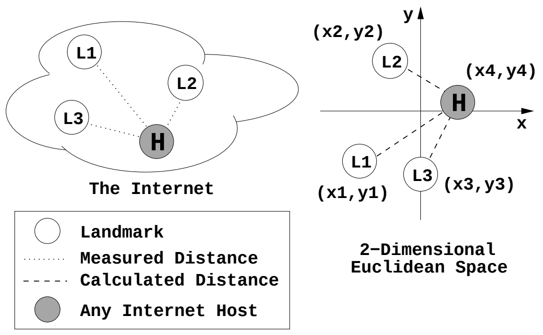

Figure 2. GNP Host Positioning

where $\varepsilon(.)$ denotes the same error function as in the landmark positioning. The GNP authors show that, in 90% of cases, the latency estimations derived from their system are within a relative error ratio of 0.53 compared to the real latency.

2.2 GNP Variants

A number of variants to the original GNP idea have been proposed. In general, they can be classified into two groups: either they optimize the selection of landmarks, or they introduce different positioning schemes.

The first group focuses on improving the landmark choice. The Lighthouses project has shown that hosts can be accurately positioned relative to any previously positioned hosts (acting as "local" landmarks), which eliminates the need for contacting the original landmarks each time a host is positioned [11]. Following the idea, PIC suggested that at least some of the local landmarks should be located close to the positioned host to improve the positioning accuracy [3]. However, using local landmarks does not eliminate the need for a global agreement on the space dimension, a set of global landmarks, and the positioning algorithm.

Some research efforts in the second group replace the Simplex-downhill computation with simpler optimization schemes [8, 15]. Others take a completely different approach and position all hosts simultaneously as a result of a global optimization process [4, 14, 16]. In this case, there is no need to choose landmarks, since every host is in fact considered to be a landmark. Such a global approach is generally faster than its iterative counterpart, which positions hosts one by one. The authors claim that it leads to better accuracy. However, because it operates on all the latencies simultaneously, it can potentially have to be re-run every time new latency measurements are available. Such a rerun is likely to be computationally expensive in large-scale systems, where the number of performed latency measurements is high.

3 SCoLE

3.1 Private Spaces

In a large-scale distributed system, different hosts may have different requirements with respect to the estimation accuracy, the validity period of computed coordinates, or even the choice of hosts that should be used for performing latency measurements because of their reputation, availability, etc. These varying preferences are impossible to accommodate in a large-scale distributed system where all hosts use the same global space parameters.

A crucial observation is that in most cases we are interested only in estimated latencies rather than in host coordinates themselves. We conclude that it is not necessary that all hosts agree on a global space definition. Instead, we propose that each host interested in latency estimation runs an independent instance of the positioning process. It constructs its own private geometric space based on local preferences, triggers latency measurements, and positions other hosts in its private space. Note that, unlike GNP, positioned hosts are no longer supposed to compute their coordinates themselves and then propagate them to other hosts. In our approach, coordinates are only considered to be intermediate data and do not need to be transferred between hosts.

To position a "target" host, a "drafter" host must measure the latencies between the target and a number of "helper" hosts. Helpers are hosts that agree to measure their latencies to given targets upon request from the drafter. The number and choice of helpers are autonomous decisions of each drafter. Note that the distinction between drafter, helper, and target hosts only refers to their roles in a single instance of the positioning process. A given host may at the same time be the drafter of its own space, a helper to some other drafters, and a target for yet other drafters.

Based on measurements provided by helpers, each drafter can construct its own private geometric space. Helpers are natural candidates to become landmarks in this space, as they inform the drafter about their latencies to the targets. To assign landmark coordinates, each drafter instructs its helpers to measure and report latencies among each other. After a drafter has assigned the landmark coordinates, it can position targets in its private space based on the latencies between the landmarks and the targets. It may also decide to re-position them some time later, if it notices that their coordinates are no longer accurate. In both cases, the drafter is the only one to decide whether and how (re)positioning should take place. This decision depends solely on the drafter configuration and can be taken independently from other drafters and from the targets themselves.

A possible scheme for positioning new nodes is depicted in Figure 3. A new target to be positioned is discovered, for

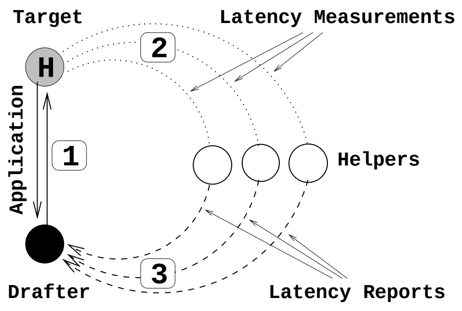

Figure 3. The Architecture of SCoLE

example, when it opens some application-level connection to the drafter (1). The drafter then triggers latency measurements between its helpers and the target (2). For example, it can directly request each helper to perform an active latency measurement to the target. Alternatively, it can instruct the target to contact each of the helpers, thus indirectly asking the latter to perform passive latency measurements. Finally, the helpers report measurement results back to the drafter, which uses them to determine the target coordinates inside its private space (3).

The above positioning scheme can be inefficient with several respects. For example, targets that contact multiple drafters will be positioned by each drafter independently, which leads to redundant latency measurements. Also, even though the latencies to co-located targets are likely to be similar, they will be measured independently, again leading to redundant measurements. We address these issues in Section 5, when discussing various practical aspects of a SCoLE implementation. To simplify the discussion, however, we will first consider the positioning scheme in its most elementary form.

3.2 Discussion

By allowing each drafter to independently construct its private geometric space and to position targets in that space, we enable each drafter to adjust the latency estimation process to its own requirements. Adjustments may include the selection of a space dimension, a positioning algorithm, and a set of helpers.

Another consequence of using private spaces is complete decentralization. SCoLE eliminates the need for global agreement and configuration, since there is no single configuration aspect on which all the hosts must agree or to which they must obey. Although running individual instances of SCoLE requires that helpers cooperate with their drafters, these cooperating groups are small and independent from each other. This property of SCoLE makes it particularly suitable for large-scale distributed systems, where neither global knowledge nor coordination is possible.

Positioning hosts in private spaces poses a difficulty when drafters need to send host coordinates to each other. Such coordinate transfers can be useful in certain situations, for example to request information about hosts located within a certain radius from a given location. We address this problem by representing a target's location not as a set of absolute coordinates, but as a set of latencies between that target and a (potentially large) number of reference hosts. Provided that the receiver has already positioned a significant subset of the reference hosts, it will be able to derive the coordinates of the transferred location within its own private space. Because the selection of the reference hosts can be negotiated, and because the latencies can be in fact latency estimates, any two drafters are still able to pass coordinates to each other.

4 Evaluation

SCoLE allows drafters to estimate latencies independently from each other. However, if multiple drafters jointly run the same application, it is desirable that latencies estimated by different drafters are highly correlated. For example, if an application requires each drafter to build a minimal host-spanning tree based on latency estimations, then estimation inconsistencies can result in each drafter building a different tree, which can lead to a malfunctioning application. On the other hand, if the correlation coefficient is high, then the application may rely on local estimations, and consider the rare cases when estimations are inconsistent as exceptions.

In this section, we show that latencies estimated in different private spaces are consistent with each other. We start with describing the datasets we use in our experiments, and discussing our experience with selecting the most appropriate space dimension. Then, we evaluate the correlation of spaces using the same positioning algorithm. Finally, we investigate the impact of using different positioning algorithms on estimation accuracy, and measure the correlation of spaces using different positioning algorithms.

4.1 Dataset Description

We evaluate SCoLE on three independent datasets, each containing a snapshot of latencies measured among a set of machines in the Internet. The first dataset was collected on June 25, 2003, using King [6]. This tool allows to estimate latencies between any pair of DNS servers, provided that they support recursive DNS queries. We chose 100 DNS servers such that they were diversified in terms of both geographical location and IP address prefix. In particular, each of them belonged to a different Autonomous System. We measured the latencies between each pair of the DNS servers. As it turned out, 37 of them did not support recursive DNS queries, which left us with a snapshot of latencies among the remaining 63 DNS servers. Note that fair distribution of the DNS servers over the entire Internet may result in an abnormally low occurrence of short latencies.

The second dataset was collected on October 1, 2003, using the RIPE-NCC's Test-Traffic Measurements Service [5]. The service infrastructure consists of a number of probing hosts deployed on the backbones of Internet Service Providers. We selected 40 of the probing hosts such that the round-trip time between any pair of them was at least 1 millisecond. The resulting set contained 33 probing hosts in Europe, 5 in the USA, 1 in Asia, and 1 in Australia. Since most of the probing hosts are located in the same continent, this trace may be biased toward short latencies.

The third dataset was collected on November 19, 2003 on 60 PlanetLab nodes [1]. PlanetLab is a distributed infrastructure of Linux hosts deployed in academic and research institutions. The hosts we used were located mainly in the USA (46), but also in Europe (7), Asia (5), Australia (1), and South America (1). We measured round-trip times between each pair of hosts using the SYN/ACK-ACK method [2]. This method evaluates the RTT between two hosts by opening a TCP connection between them. The RTT is measured by the server host during the TCP handshake as the delay between sending the SYN/ACK packet and receiving the corresponding ACK packet.

4.2 Space Dimension vs. Estimation Accuracy

Before evaluating SCoLE itself, we address the question: which space dimension offers the most accurate latency estimations? The answer to this question is the base for our further experiments.

For each space dimension N, we randomly select $N+1$ hosts to act as landmarks and use the Simplex-downhill method to compute the landmark coordinates. We position the remaining hosts relative to these landmarks using only the measured latencies between the positioned hosts and the landmarks. Note that this positioning process does not use any of the measured target-to-target latencies.

For each pair of targets, we compare the estimated latency between them to the measured one. We calculate relative errors $\varepsilon(.)$ for latencies between all pairs of hosts similar to GNP [10]:

$$\varepsilon(d_{AB}, d^*_{AB}) = \left|\frac{d^*_{AB} - d_{AB}}{min(d_{AB}, d^*_{AB})}\right|$$

where d_{AB} and d^*_{AB} respectively denote the measured and estimated latencies between hosts A and B. The smaller the value of ε, the more accurate the estimation. We performed this experiment repeatedly 1000 times for each value of N between 2 and 12.

The aggregated results are depicted in Figure 4. Figure 4a presents the cumulative distribution function (CDF) of relative errors, whereas Figure 4d shows the percentage of all estimations for which $\varepsilon \leq E$, for E equal to 0.25, 0.5, and 1.0, all for the King dataset. The Figures 4b, 4e and 4c, 4f present the corresponding results for the RIPE and PlanetLab datasets, respectively.

In all datasets, the accuracy increases between dimensions 2 and 6, then remains stable between dimensions 6 and 9, to finally decrease between dimensions 9 and 12. These results confirm the findings from [15]. Note that other researchers observed accuracy convergence for dimensions 12 and higher [3]. However, since in our experiments dimension 6 is the lowest to give the best results, we decided to use it in all further experiments.

4.3 Helper Selection vs. Space Correlation

SCoLE allows different drafters to construct their private spaces independently based on different sets of measurements. In this experiment, we investigate to what extent the latency estimations derived from different private spaces are consistent with each other. To do this, we create two private spaces based on disjoint sets of landmarks. All other hosts are positioned relative to the landmarks. For each pair of hosts, we compare the two latencies estimated within the two private spaces.

Figures 5a, 5b, and 5c plot the latencies estimated within two private spaces chosen at random and constructed from the King, RIPE, and PlanetLab datasets, respectively. Each point (x, y) corresponds to the latencies x and y estimated in the two private spaces for the same pair of hosts. As can be observed, most points are close to the ideal line $y = x$.

However, the correlation of two private spaces obviously depends on the choice of these spaces. To obtain more meaningful results, we calculated a correlation coefficient C_{cor} for each possible pair of 100 random spaces. As it turned out, there were pairs of spaces for which C_{cor} was close to 0. Detailed analysis of these pairs revealed that they contained at least one space based on a landmark set with high total discrepancy *TD*. Since using such self-inconsistent landmark sets degrades the space correlation, we decided to eliminate them by restricting the maximum *TD* of landmark sets that can be used as bases for private spaces. In a real-life system, landmark sets producing self-inconsistent spaces can be easily identified, in which case a system should choose a different landmark set instead.

Figures 5d, 5e, and 5f depict the cumulative distributions of correlation coefficients for TD_{max} equal to 0.5 and ∞ calculated for the King, RIPE, and PlanetLab datasets, respectively. For each TD_{max}, we used the same set of 100 random spaces, but we filtered out spaces where $TD > TD_{max}$. Then, we calculated correlation coefficients for all

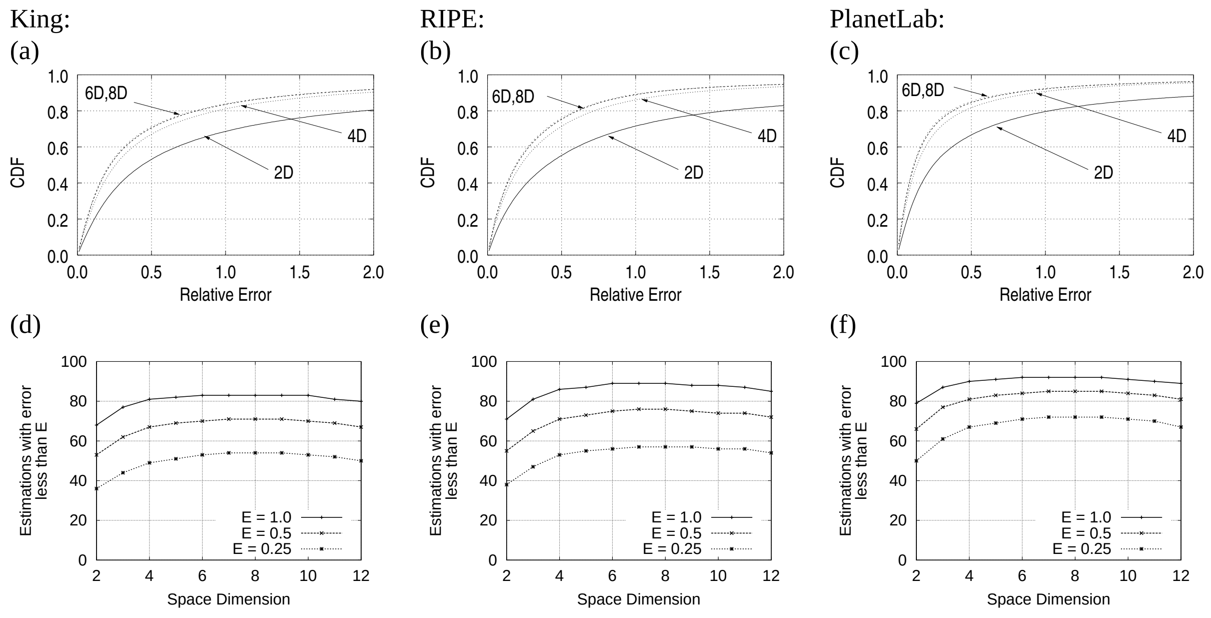

Figure 4. Space Dimension vs. Accuracy: Error Distribution for 1000 Random SCoLE Spaces

Maximum *TD*		∞	3	2	1	0.5	0.25	0.1
King	Eliminated spaces (%)	0	3	4	10	21	39	72
	Average C_{cor}	0.85	0.89	0.89	0.89	0.90	0.91	0.92
RIPE	Eliminated spaces (%)	0	2	3	6	14	27	57
	Average C_{cor}	0.93	0.95	0.95	0.95	0.96	0.96	0.96
PlanetLab	Eliminated spaces (%)	0	1	1	2	5	14	35
	Average C_{cor}	0.94	0.94	0.94	0.94	0.94	0.95	0.95

Table 1. Improving Space Correlation by Restricting the *TD* Value

pairs of the remaining spaces. The statistics for this experiment are presented in Table 1, which also includes results for TD_{max} equal to 3, 2, 1, 0.25, and 0.1. As can be observed, using $TD_{max} = 0.5$ eliminates 5% to 21% of pairs (depending on the dataset), but leads to average C_{cor} of at least 0.9. Interestingly, although limiting *TD* seems to be important for inter-space correlation, it only slightly improves the estimation accuracy in general. We plan to investigate this phenomenon in the future.

4.4 Algorithm Selection vs. Estimation Accuracy

In SCoLE, each drafter positions its targets based on the measured latencies between the targets and its chosen helpers. In addition, latencies must also be measured between each pair of helpers to assign coordinates to each helper in the space construction phase. Still, however, the latency estimations are derived only from a small subset of all potential measurements that could (theoretically) be performed for each pair of hosts in the system. An interesting question is whether and to what extent the latency estimations would improve if each drafter measured the latencies between each target pair, and then positioned all the targets based on a full set of measurements between them, using some global optimization algorithm. In other words, we compare the accuracy of SCoLE to the theoretical best accuracy among latency estimation techniques based on host positioning.

To calculate the maximal accuracy, we applied the global optimization scheme proposed by Vivaldi [4] to each of our datasets, every time using *all* the latencies they contain. Then, we compared the thus-obtained optimal accuracy to that of pure SCoLE evaluated in the previous experiment.

The accuracy comparison is presented in Figure 6 (lines labeled "Pure SCoLE" and "SCoLE/Vivaldi Full"). As can be observed, running SCoLE/Vivaldi Full on the King dataset did not result in any improvement compared to the results obtained with Pure SCoLE. On the other hand, the

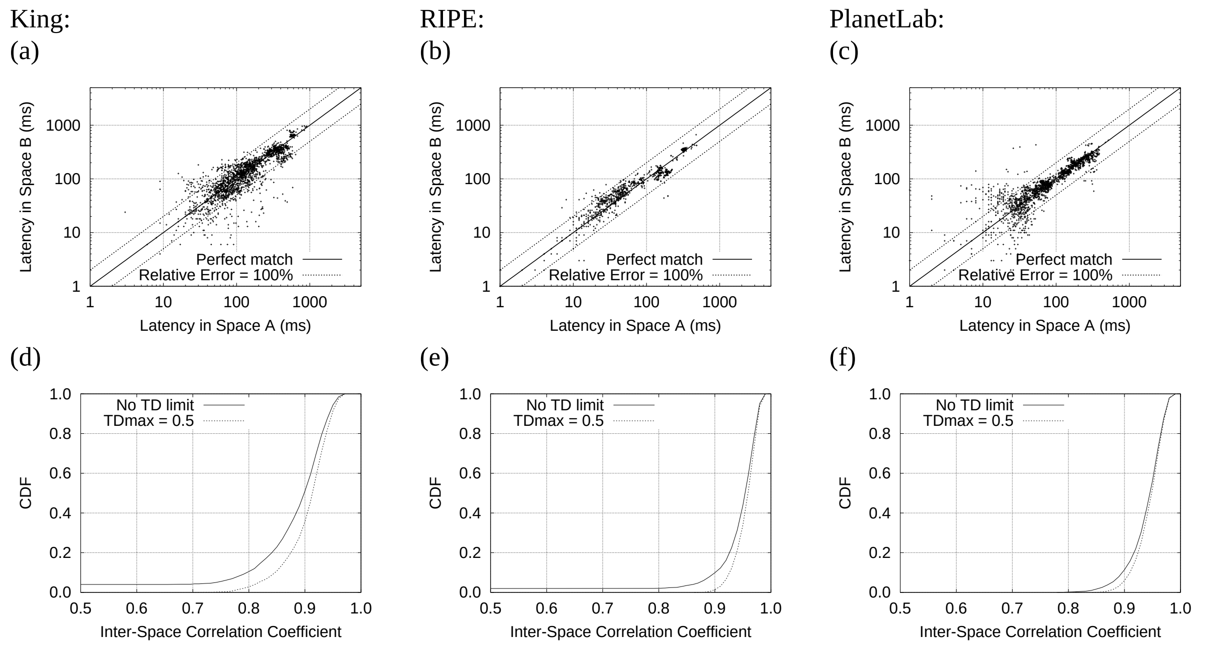

Figure 5. Helper Selection vs. Correlation: 2 Independent Spaces and 1000-space Pair CDFs

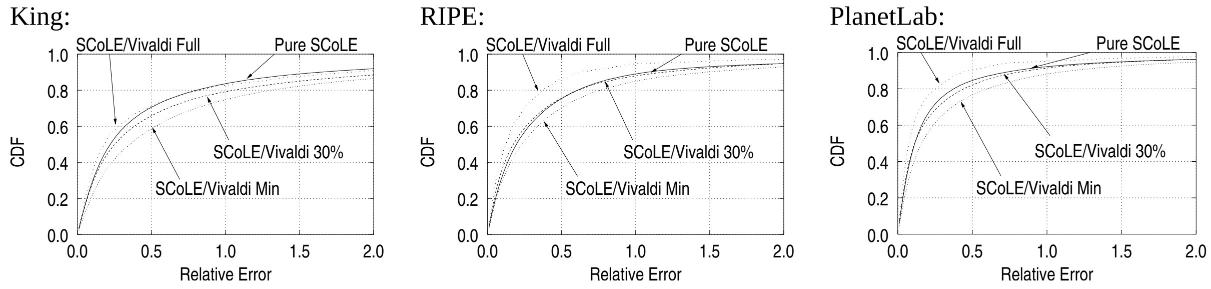

Figure 6. Algorithm Selection vs. Accuracy: Error Distribution

accuracy increased for the other two datasets. Interestingly, the improvement was clearly higher for the RIPE dataset than for that collected on PlanetLab. Since Pure SCoLE and SCoLE/Vivaldi Full produce similar (relative) host coordinates (as we show in Section 4.5), we believe that the difference in estimation accuracy is caused by *slight* differences between relative coordinates themselves. These differences have almost no impact on the estimation of long latencies constituting the main part of the King dataset, but they do affect the estimations of short latencies enclosed in the RIPE dataset. This phenomenon also explains why the diversity of latencies in the PlanetLab dataset causes SCoLE/Vivaldi Full to only moderately improve the estimation accuracy.

Because collecting a full set of latencies is usually not possible, a question arises how good would the global optimization approach perform on a minimal data subset required by Pure SCoLE. Such a minimal subset consists of all latencies between each of $N+1$ helpers and all the other hosts. To answer this question, we applied Vivaldi to 100 minimal data subsets, each including a random selection of helpers.

The results are presented in Figure 6 (lines labeled “Pure SCoLE” and “SCoLE/Vivaldi Min”). For all three datasets, SCoLE/Vivaldi Min performed worse than Pure SCoLE. This is not surprising, as the nature of Vivaldi assumes fair distribution of measurements across all the hosts, and not their concentration on a small number of helpers.

Since SCoLE/Vivaldi Min performed worse than Pure SCoLE, we wondered how many helpers are necessary for a drafter running Vivaldi to achieve the same performance as drafters using the Pure SCoLE scheme. In order to find out, we again applied Vivaldi to 100 random data subsets. This time, however, instead of containing the minimal number of $N+1$ helpers, they included from 20% to 100% of the hosts as helpers.

The aggregated results for this experiment are shown in Figure 6. Except for the King dataset, the accuracy of SCoLE/Vivaldi is similar to that of Pure SCoLE, if the number of helpers included in a data subset is about 30%. This result could not be observed for the King dataset, as it did not outperform Pure SCoLE even when the entire dataset was used. We conclude that to benefit from running Vivaldi, a drafter would have to use more hosts as its helpers. Although it is unlikely for a drafter to have a large number of helpers, there are methods of obtaining measurements from non-helper hosts, as we discuss in Section 5. Also, note that the accuracy achieved by Pure SCoLE using minimal datasets is not much worse than the theoretical optimum achieved by SCoLE/Vivaldi Full. This indicates that gathering more data than really necessary may turn out be expensive compared to the expected gain in estimation accuracy.

4.5 Algorithm Selection vs. Space Correlation

Except for choosing its own landmark set, each host in SCoLE can decide on which positioning algorithm to use. In this section, we discuss how this decision affects the correlation of latency estimations by comparing the estimations made by hosts that use different positioning algorithms.

We applied both SCoLE/Vivaldi and Pure SCoLE to 100 random data subsets of each of our datasets. We ensured that both achieve similar *accuracy* by using minimal data subsets for Pure SCoLE, and 30% data subsets for SCoLE/Vivaldi (see Section 4.4). Then, we calculated the correlation coefficients for all space pairs, where one is made with Pure SCoLE, and the other with SCoLE/Vivaldi 30%.

The results are presented in Figure 7. In all datasets, the spaces generated by Pure SCoLE are highly correlated with these generated by SCoLE/Vivaldi 30%. However, since there is some difference in accuracy achieved by SCoLE/Vivaldi 30% and SCoLE/Vivaldi Full, we decided to compare the correlation of Pure SCoLE spaces against the latter as well.

The results are presented in Figure 8. Also in this case, estimations made by Pure SCoLE using minimal data subsets are highly correlated with the theoretical optimum achieved by SCoLE/Vivaldi Full.

These three experiments prove that, even if drafters use different algorithms to construct their private spaces, their latency estimates remain consistent. Moreover, the estimates are also consistent with the theoretical optimum. This property allows drafters to perform their estimations independently, even if they jointly run some distributed application that requires consistent estimates to operate correctly.

5 Practical Issues

In this section, we discuss how the elementary scheme proposed in Section 3 can be improved to allow a real-world system implementation.

5.1 Latency Measurements

The only non-local operations in SCoLE are latency measurements. Although there are many methods how they can be performed, some of these methods are more attractive than the others. In a large-scale system, it is often desirable that no additional traffic is generated while the latencies are being measured. This goal is achieved by passive schemes, which merely monitor the traffic that already exists in the system, and measure latencies as a side effect of that traffic.

An example of a passive scheme is the SYN/ACK-ACK method, proposed in Webmapper [2]. This method allows a (server) host to measure its latency to any other (client) host, provided that the client opens a TCP connection to the server. The round-trip time between two hosts is measured by the server during the three-way handshake as the delay between sending the SYN/ACK packet and receiving the corresponding ACK.

The SYN/ACK-ACK method is particularly attractive for SCoLE, because it requires that the measuring software is deployed only on helpers, and not on targets. The only requirement is that drafters can make their targets open TCP connections to the helpers. This can be easily achieved at the application level. For example, in a Content Delivery Network (CDN), a (drafter) Web server can make its (target) Web client open HTTP connections to several other (helper) Web servers by embedding a special list of object references in the returned Web documents. The drafter may want to ensure that each referenced object (i) is located at a different helper, (ii) is small (to avoid communication overhead), and (iii) does not affect the appearance of the Web document in which it is embedded. An example of such an object is a transparent 1x1-pixel image. Since, by default, Web clients follow all the embedded object references, the target will open HTTP connections to all the helpers, thus allowing each helper to measure the latency using the SYN/ACK-ACK method.

We have implemented a prototype of SCoLE based on the SYN/ACK-ACK method. The drafter is a simple C program that monitors the state of its helpers and maintains a file containing a list of HTML references to transparent 1x1-pixel GIF images served by the alive helpers. The reference

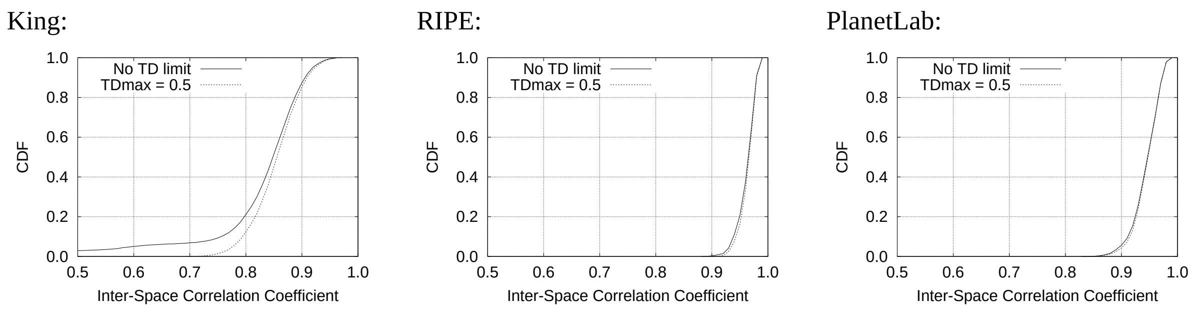

Figure 7. The Correlation between Pure SCoLE Spaces and SCoLE/Vivaldi 30% Spaces

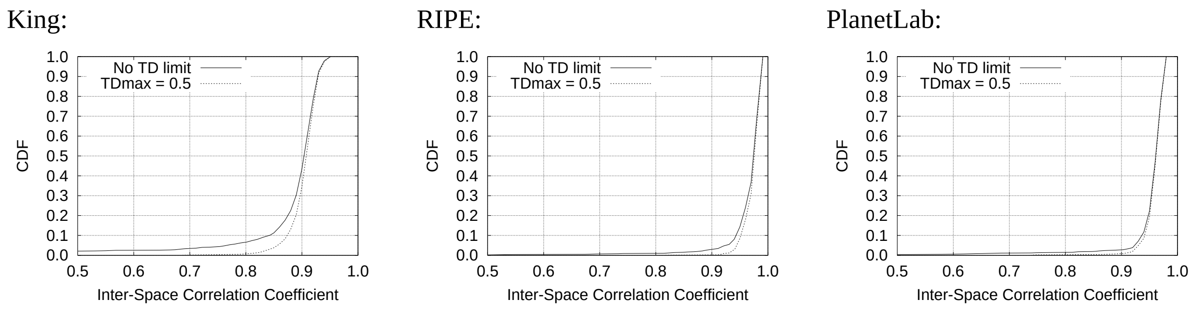

Figure 8. The Correlation between Pure SCoLE Spaces and the SCoLE/Vivaldi Full Space

list is dynamically included in selected Web pages of our departmental Web server. As a consequence, all Web clients retrieving these pages also retrieve small GIF images from the helpers alive at the moment. This allows the helpers to passively measure their latencies to the Web clients. Also, to discover the latencies between the helpers, the drafter periodically instructs each helper to act as a Web client and download images from all the other helpers.

We deployed the prototype on our departmental Web server, with helpers running on 14 PlanetLab nodes. During 7 weeks of continuous operation, the prototype collected a set of latencies that suffice to position 11,405 Web clients.

5.2 Target Clustering

Even though latency measurements can be performed inexpensively, their results still need to be stored and processed by drafters. In a large-scale system, drafters may need to position very large numbers of targets, which requires large storage capacities. Moreover, even though co-located targets are likely to have similar latencies to the same helpers (and therefore similar coordinates), drafters position these targets independently, which leads to redundant latency measurements.

We propose to reduce the necessary storage capacity and the number of measurements by grouping co-located targets into clusters. Since the coordinates of clustered targets are likely to be similar, drafters can position entire clusters at once, instead of positioning each target independently.

An efficient clustering scheme is called “network-aware clustering,” which forms clusters out of BGP prefixes [7]. Prefixes describe (sets of) networks in BGP, which is a routing protocol used for communication among different Autonomous Systems [13]. The authors of the network-aware clustering scheme claim that it clusters co-located hosts in 99.99% of cases.

We have implemented network-aware clustering and evaluated its impact by analyzing the trace collected by our SCoLE prototype. The original 11,405 unique IP addresses were grouped into 3,760 BGP prefixes, which would reduce the storage capacity and the number of measurements by a factor of 3. We also investigated the impact of clustering on a longer trace that contained 1,816,686 unique IP addresses of the Web clients of our departmental Web server. In this case, the IP addresses were grouped into 34,511 BGP prefixes, which would reduce the required number of measurements and storage capacity by a factor of 52.7.

5.3 Data Sharing

In a large-scale system, it is likely that some targets contact more than one drafter during the application lifetime.

In this case, several drafters would be interested in latencies to such targets. However, since in the elementary SCoLE version drafters trigger their measurements independently from each other, redundant measurements to these targets may be performed. This overhead can be reduced by allowing drafters to share the measurement data they collect.

There are several techniques enabling efficient data sharing in large-scale systems, such as lazy dissemination and publish-subscribe systems. In general, they ensure that the data sent by one host are eventually propagated to other hosts interested in the data. Note that both these properties meet the SCoLE requirements, as it requires neither immediate data propagation, nor the guarantees that all drafters receive the data.

Allowing SCoLE drafters to share the measurement data with each other eliminates redundant measurements. It does not mean, however, that drafters lose the flexibility of choosing which measurements they use to position their targets. Since the measurement results can be signed by the helpers that produce them, each drafter can still identify the measurement origin and decide whether it should be trusted or not. Also, because drafters share only raw inter-host latencies, and not host coordinates, each drafter can still run its preferred positioning algorithm, just as if the received latencies were measured by its own helpers.

6 Conclusion

We have presented SCoLE, a scalable cooperative latency estimation system. SCoLE allows each participating host to construct its "private" space and position other hosts in that space. Given any two hosts, SCoLE estimates the latency between them as the Euclidean distance between their associated coordinates.

In SCoLE, the private space parameters as well as the positioning algorithm can be adjusted on a per-host basis, which improves system flexibility to an extent that could not be achieved by previous systems using a single global space. More importantly, since private spaces are mutually independent, SCoLE is more scalable than its predecessors, whose scalability is limited by the necessity of global negotiation of space parameters and by the global knowledge of these parameters.

We have shown that latency estimates performed in different private spaces are highly correlated with each other, while providing reasonable estimation accuracy. Also, we have addressed a number of practical issues that need to be taken into account when implementing SCoLE in a real-world system. We implemented and deployed SCoLE prototype using 14 PlanetLab nodes as helpers, and calculated the coordinates of 11,405 Web clients that accessed our departmental Web server during the 7 weeks of continuous prototype operation.

We plan to develop the prototype further and use it to locate Web clients in Globule, a peer-to-peer Content Delivery Network that our group is developing [12]. We believe that SCoLE will help us determine optimal replica locations, as well as efficiently select replicas on a per-client basis.

Acknowledgment

We would like to thank Henk Uijterwaal from RIPE NCC in Amsterdam for providing us with the data collected by the Test-Traffic Measurement Service.

References

[1] The PlanetLab Project. `http://www.planet-lab.org/`.

[2] M. Andrews, B. Shepherd, A. Srinivasan, P. Winkler, and F. Zane. Clustering and Server Selection Using Passive Monitoring. In *21st IEEE INFOCOM*, June 2002.

[3] M. Castro, M. Costa, P. Key, and A. Rowstron. PIC: Practical Internet Coordinates for Distance Estimation. Technical Report MSR-TR-2003-53, Microsoft Research, Sept. 2003.

[4] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Practical, Distributed Network Coordinates. In *2nd ACM Workshop on Hot Topics in Networks (HotNets-II)*, Nov. 2003.

[5] F. Georgatos, F. Gruber, D. Karrenberg, M. Santcroos, A. Susanj, H. Uijterwaal, and R. Wilhelm. Providing Active Measurements as a Regular Service for ISP's. In *Passive and Active Measurements Workshop*, Apr. 2001.

[6] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *2nd ACM Internet Measurement Workshop*, Nov. 2002.

[7] B. Krishnamurthy and J. Wang. On Network-Aware Clustering of Web Clients. In *ACM SIGCOMM*, Aug. 2000.

[8] H. Lim, J. Hou, and C.-H. Choi. Constructing Internet Coordinate System Based on Delay Measurement. In *ACM Internet Measurement Conference*, Oct. 2003.

[9] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 4(7), 1965.

[10] T. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *21st IEEE INFOCOM*, June 2002.

[11] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. Lighthouses for Scalable Distributed Location. In *2nd International Workshop on Peer-to-Peer Systems*, Feb. 2003.

[12] G. Pierre and M. van Steen. Design and Implementation of a User-Centered Content Delivery Network. In *The 3rd IEEE Workshop on Internet Applications*, June 2003.

[13] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771, Mar. 1995.

[14] Y. Shavitt and T. Tankel. Big-Bang Simulation for Embedding Network Distances in Euclidean Space. In *22nd IEEE INFOCOM*, Apr. 2003.

[15] L. Tang and M. Crovella. Virtual Landmarks for the Internet. In *ACM Internet Measurement Conference*, Oct. 2003.

[16] M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. In *1st ACM Workshop on Hot Topics in Networks (HotNets-I)*, Oct. 2002.