Estimating Network Proximity and Latency

Puneet Sharma, Zhichen Xu, Sujata Banerjee, and Sung-Ju Lee
Mobile & Media Systems Lab
Hewlett-Packard Laboratories
Palo Alto, CA 94304

***Abstract*— Network proximity and latency estimation is an important component in discovering and locating services and applications. With the growing number of services and service providers in the large-scale Internet, accurately estimating network proximity/latency with minimal probing overhead becomes essential for scalable deployment. Although there exist a number of network distance estimation schemes, they either rely on extensive infrastructure support, require the IP address of the potential targets, falsely cluster distant nodes, or perform poorly with few measurement errors. We propose *Netvigator*, a scalable network proximity and latency estimation tool that uses information obtained from probing a small number of landmark nodes and intermediate routers (termed milestones) that are discovered en route to the landmarks, to identify the closest nodes. With very little additional probing overhead, Netvigator uses distance information to the milestones to accurately locate the closest nodes. We developed a Netvigator prototype and report our performance evaluation on Planet-Lab and in the intranet of a large enterprise. We also test its scalability using simulations. The performance of Netvigator is compared with Vivaldi and GNP.**

I. Introduction

Proliferation of Internet access has not only whetted consumers' appetite for context-sensitive personalized delivery of services, but has also put the tools for service creation in their own hands [2]. We believe that the distinction between service consumer and provider will blur further. Our vision differs from current hosting of services by large providers because we believe there will be a vast number of small providers in the new Internet economy. Such an environment will cause exponential growth in service composition and distribution on the Internet. Building a scalable and adaptive Internet service infrastructure is a key enabler for our vision. An important challenge facing future network infrastructure is to balance the tradeoffs between providing individualized service to each client and making efficient use of the networked resources. Efficient resource utilization enables the same infrastructure to accommodate more services and clients and respond better to flash crowds.

A key issue in effectively utilizing network resources and services is efficiently and quickly locating the desired resources or services in specific network locations. These kinds of location services allow a service provider to construct efficient service overlay networks [9], [14], [25], [32], which for example could be used to distribute rich media content, enable a client to identify the closest cache/proxy that has the desired data or service, enable a client to quickly locate a well provisioned nearby server for participating in a massive multiple-user online game, or to quickly construct a proximity-aware peer-to-peer (P2P) overlay for applications such as content sharing. Hence, techniques that accurately and efficiently estimate locality of resources/services and compute network distances have become important.

Currently proximity estimation is based on pairwise distance estimate between any two given nodes. Such schemes find the node closest to a client from a given set of potential targets by estimating the distance to each candidate and picking the minimum. This process requires a priori knowledge about the set of all the candidates. Furthermore in some systems, such as P2P systems, the set of service providing nodes changes dynamically, necessitating fast recomputation of proximity. In practice, when finding a service/resource, knowledge about the distance to the complete set of potential servers is usually not of interest, and ideally, the proximity estimation tool should be able to answer client queries such as *"Which is the closest node that can provide a media transcoding service?"* without requiring information about all potential nodes. Most of the distance estimation techniques build a globally consistent network model designed to optimize the estimation error averaged globally. For proximity estimation it is more important to lower the distance estimation error between closeby nodes. The global optimization schemes perform poorly for proximity estimation. Their proximity accuracy can be greatly improved by relaxing the global consistency requirement.

Landmark clustering [16], [20] is a popular scheme used for network distance estimation that uses a node's distances to a set of landmark nodes to estimate the node position in a Cartesian space. However, current landmark clustering techniques can be prone to false clustering where distant nodes are positioned near each other. Further, the estimation quality of current landmark clustering schemes depends on the quality of measurement data and can be significantly inferior to the optimal when there is bad measurement data.

In this paper, we describe *Netvigator* (Network Navigator), a new network proximity and latency estimation tool that is more efficient and accurate than the existing schemes.[1] Netvigator primarily focuses on proximity estimation (i.e. to rank nodes according to proximity to any given node) as well as distance estimation. In Netvigator, each node measures distances to a given set of landmarks, similar to other landmark clustering techniques. Netvigator additionally leverages topology information by recording the distances to the *milestones* that are encountered while probing the landmarks. It must be noted

Zhichen Xu is now with Yahoo! Inc.

[1] In this paper we use the terms latency and distance interchangeably.

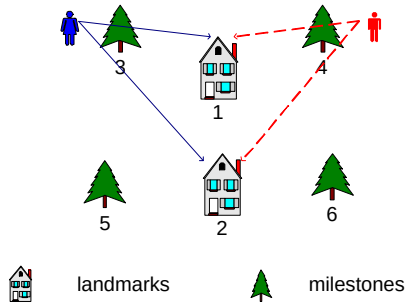

Fig. 1. Example of the enhanced landmark scheme.

that our scheme does not require deployment of milestones, instead they are discovered during the probing process, e.g., the intermediate routers encountered during a traceroute. The milestones have the capability to intercept measurement packets and respond to the client nodes. Instead of attempting to embed all the nodes in a global Cartesian-space based on measurements, Netvigator performs local clustering for proximity estimation.

This paper makes the following contributions:

- We propose three novel clustering algorithms that utilize the distance information from the landmarks as well as the milestones to obtain higher accuracy in finding the closest node, with relatively small measurement overhead. Utilizing distance information from both the landmarks and milestones not only improves the accuracy but also makes our technique robust to bad measurements. In addition to proximity, one of the clustering techniques also produces a highly accurate distance estimate.
- We developed a prototype of our scheme and evaluated it on PlanetLab [19] and the intranet of a large enterprise (Hewlett-Packard) in addition to a large scale simulation study. Our experiments show that with 90% confidence, the real closest node falls in the top two nodes our algorithm identifies. Further, NetVigator's distance estimation outperforms that of GNP and Vivaldi.

The remainder of the paper is organized as follows. Section II describes the details of Netvigator, followed by experimental results in Section III. We discuss deployment experiences, efficiency and scalability issues in Section IV. Section V surveys the related work. Concluding remarks and directions for future work are presented in Section VI.

II. NETVIGATOR

Before describing Netvigator techniques we look at how network proximity estimation methods can be used in general. One of the primary use is to find the closest node providing a particular service such as caching, transcoding, etc. Proximity estimation can also be used to narrow down choices in a multi-attribute constraint scenario. In this case, k closest nodes are first computed using proximity estimation and then the constraint matching is performed on the smaller set (k nodes).

In Netvigator, each node measures distances to a given set of landmarks, similar to other landmark clustering techniques. Netvigator additionally records the distances to the milestones that are encountered while probing the landmarks. The idea behind this approach is illustrated in Figure 1 using a simple analogy. Two people estimate their physical location by measuring their distances to the two houses (i.e., the landmarks). Each of them also records their distances to the trees (i.e., milestones) that are on the way to the two houses. Without using the distances to the trees 3 and 4, a naïve landmark clustering approach would conclude that the two people are close to each other because they have similar distances to the two landmarks. By accounting for the distances to the milestones, false clustering can be avoided, thus increasing the accuracy of the proximity estimation.

In Netvigator, a small number of landmarks are used for bootstrapping and a large number of milestones are used for refinement. The milestones are discovered during the probing process, e.g., the intermediate routers encountered during a `traceroute`. The milestones have the capability to intercept measurement packets and respond to the client nodes. It must be noted that our scheme **does not require** deploying milestones as part of the infrastructure. In Netvigator, given N client nodes and L global landmark nodes, each client conducts a traceroute to each landmark node. Thus a total of N*L traceroutes are conducted asynchronously in each measurement period. The details of our proximity estimation scheme are described below.

(1) Each node sends probe packets to the landmarks for round-trip time measurement.[2] These packets may encounter milestones en route to the landmarks. When a milestone node receives a probe packet, it sends an acknowledgment packet back to the node that originated the probe packet.

(2) After a node receives all acknowledgment packets from the landmarks and milestones (if any), it constructs a landmark vector that includes the distances to all the landmarks as well as the milestones the measurement packets have encountered.

(3) Each node submits its landmark vector to a repository called global information table.

(4) Upon receiving a query from a node to find the k closest nodes, the infrastructure carries out the following steps:

(4.1) With the landmark vectors of all the candidate nodes stored in the global information table and the landmark vector of the querying node, apply the clustering algorithm to reduce the size of the candidate set to identify k top candidates.[3]

(4.2) Send the information of these identified candidates to the client node.

(5) The client node performs RTT measurements to the identified top k candidates.

The better performance of Netvigator as opposed to other landmark based techniques can be attributed to two reasons. First, our use of milestones improves the information about the local network characteristics. This reduces false clustering without increasing the number of landmarks. Second, Netvigator leverages local clustering techniques instead of globally mapping all the nodes to a Cartesian space. Due to increased accuracy, each client needs to probe only a small number of landmark

[2]We assume the nodes have landmark information through some announcement and discovery mechanisms.

[3]Section II-A describes the clustering algorithms.

TABLE I
NOTATIONS.

Notation	Description
n	a node that wants to find the nearest service node.
C	the set of candidate nodes the global information table identifies by examining only the distances to the landmarks.
$L(n,c)$ and $c \in C$	the common set of nodes (landmark and milestone) that c and n have measured to.
$dist(a,b)$	the distance(latency) between nodes a and b.

nodes. After performing clustering, a client needs to perform RTT measurements to only a small number of top candidate nodes the clustering algorithm identifies. Note the only additional overhead of our scheme is the ACK packet transmissions from the milestones. A node does not send additional messages to locate the milestones, as they are encountered by probe packets on their way to at least one landmark.

A. Clustering Algorithms

Before we describe the clustering algorithms, we first define notations in Table I. The three clustering algorithms we present are: min_sum, max_diff, and $inner\_product$. The clustering algorithms min_sum and max_diff are based on the *triangle inequality*, while $inner\_product$ is motivated by information retrieval literature.[4] For each node, the k candidates having the smallest clustering metric values are picked as the k closest candidates. While all three clustering algorithms have been designed with the goal of providing proximity information, the min_sum clustering metric can also be used for distance estimation.

1) min_sum: The intuition behind min_sum is that if there are sufficient number of landmark nodes that two nodes n and c measure against, it is very likely one of the landmark nodes is located on the shortest path between the two nodes. Suppose this landmark node is l. The sum of $dist(n,l)$ and $dist(c,l)$ should be minimal if the triangle inequality holds. For a given node n and its candidate node set C, nodes are ranked in the increasing order of the min_sum clustering metric as follows.

For each node $c \in C$,

$$min\_sum(n,c) = min_{\forall l \in L(n,c)}\{dist(n,l) + dist(c,l)\}$$

Note from the above definition, $min\_sum(n,c)$ is the shortest distance among all overlay paths between nodes n and c constructed using each milestone/landmark common to nodes n and c. Thus $min\_sum(n,c)$ provides an upper bound on latency between nodes n and c and can be used as an estimate of the latency between them. In other words, $\widehat{dist}(n,c) = min\_sum(n,c)$, where the estimate of $dist(n,c)$ is denoted by $\widehat{dist}(n,c)$.

For a given node n, when all nodes $c \in C$ have been ranked using the above metric, the k closest nodes to node n are identified as the first k nodes in this ranked list, i.e., the k nodes that have the smallest values of the min_sum metric. Thus the closest node (say node x) to node n is the one that satisfies the following:

$$min\_sum(n,x) = min_{\forall c \in C: l \in L(n,c)}(dist(n,l) + dist(c,l)).$$

2) max_diff: Similar to min_sum, the idea behind max_diff is that if there are sufficient number of landmark nodes that both n and c measure against, then there is a large likelihood that there exists a landmark node l such that c is on the shortest path from n to l, or n is on the shortest path between c and l. In that case, $ABS(dist(n,l) - dist(c,l))$ is the maximum when the triangle inequality holds.[5] The formula $max\_diff(n,C)$ is defined as

$$max_{\forall c \in C: l \in L(n,c)} ABS(dist(n,l) - dist(c,l)).$$

3) $inner\_product$: The algorithm $inner\_product$ is motivated by document ranking algorithm used in information retrieval [26]. The semantic information of a document is represented as a term vector. The weight of each term is set to be proportional to the frequency of the term in the document and inversely proportional to the total number of documents that contain this term in the entire corpus. The intuition is that the more frequent a term in a document, the more likely the term uniquely identifies the document. However, if the term also appears in a lot of other documents, the importance of the term should be diminished. The similarity between a document and a query is determined using the inner product of the vector representation of the two. Applying this concept, $inner\_product$ expects that if a landmark node is close to node n, then it can give a better indication of the physical network position of n. $inner\_product(n,C)$ is defined as

$$max_{\forall c \in C} \Sigma_{l \in L(n,c)}((1.0/(dist(n,l)^2)) \times (1.0/(dist(c,l)^2))).$$

Our experimental results in the following section will show that the algorithms min_sum and max_diff perform better than the $inner\_product$ algorithm. In addition, only min_sum provides a reasonable distance estimate between any two nodes and thus holds the most promise for deployment. For $inner\_product$, the choice of square in $1.0/dist(n,l)$ is somewhat arbitrary with a goal to favor a close landmark node than a far away one. In the future, we will explore possible improvements to $inner\_product$.

B. Prototype Implementation

We prototyped Netvigator, our network proximity and latency estimation system. Netvigator comprises of the following three modules:

- **Path Probing Module**: The path probing module resides on all participating nodes. It sends probes towards the landmarks to collect the path information. Our system uses the *traceroute* tool. Each client periodically collects the path information and sends landmark vectors to the information data module.
- **Information Data Module**: The information data module collects the path information from the participating nodes.

[4] Violation of triangular inequality in the Internet has been reported in [1], [34]. However, those results do not apply in this context as they do not include the intermediate routers as points of the triangle.

[5] The function $ABS(x)$ returns the absolute value of x.

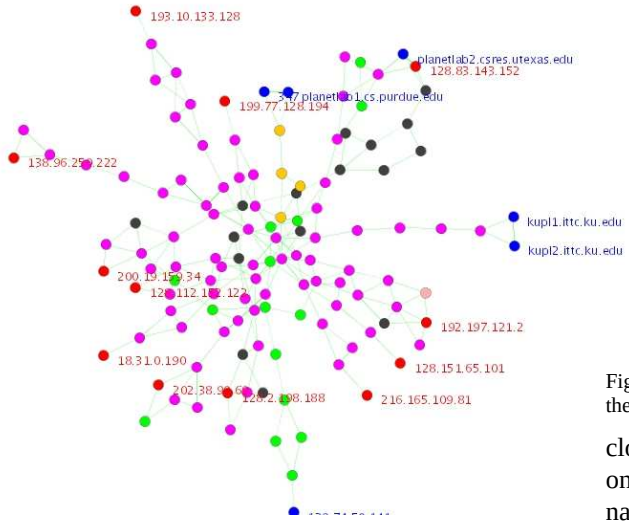

Fig. 2. Visualization of the partial topology used in clustering.

The data is filtered to remove any specious information. If available, it also consults a *router-alias* database to further enhance the data. The router-alias database contains a list of interface IP addresses assigned to the same physical router. This enables the identification of multiple interface IP addresses to a single router, rather than identifying each interface IP address as a separate router.

- **Clustering Module**: From the partial topology information, the clustering module performs the proximity computation for different nodes.

In the current Netvigator implementation, the information data module and the clustering module are centralized and reside on a single node.

III. Performance Evaluation

We experimented with Netvigator prototype on two different *real* networks: a geographically distributed HP enterprise network and a set of *PlanetLab* nodes connected via open Internet [19].

To give the readers a flavor of how our tool performs before diving into a detailed evaluation, Figure 2 visualizes the partial topology information our tool collects and uses to derive the results. Figure 3 shows the visualization of the top 5 closest nodes (the dark red nodes) identified by our tool for the node "kupl2.ittc.ku.edu" (the node in the center). In the figure, the length of the edges between the nodes is proportional to the actual measured network distances. We observe from the figure that our technique is quite effective in proximity estimation. The visualizations were generated using Zoomgraph [35].

Netvigator computes the k closest nodes for each participating node. We also compute the latency estimate for all client node pairs and evaluate its accuracy. While there are several techniques that estimate latency between node-pairs, there are no other available methods that directly output the k closest nodes. Thus for comparison purposes, we need to use the output of a distance estimator indirectly. We compare our results with GNP [16], which has been shown to outperform other schemes such as IDMaps. We also compare the performance of Netvigator with Vivaldi [8], a more recent scheme, for latency estimation in Section III-E. For both GNP and Vivaldi, the k closest nodes for each participating node were computed based on the Euclidean distance calculation from the network coordinates GNP or Vivaldi respectively generates. For the dimension parameter of the coordinate space in the GNP method, we use values between 5 and 7, which were shown by its authors to give good performances. For Vivaldi, we use [13], which embeds the client nodes in a 3-dimensional Euclidean space to obtain the end-to-end latency estimate.[6] For a fair comparison, the same set of N*L measurements (ping for GNP and Vivaldi and traceroute for Netvigator) are provided as input to the three schemes. To stabilize the Vivaldi coordinates, same set of measurements are fed to the Vivaldi system 100 times. The end-nodes are expected to perform direct `ping` measurements to these k nodes to find the closest target or use the distance estimates generated.

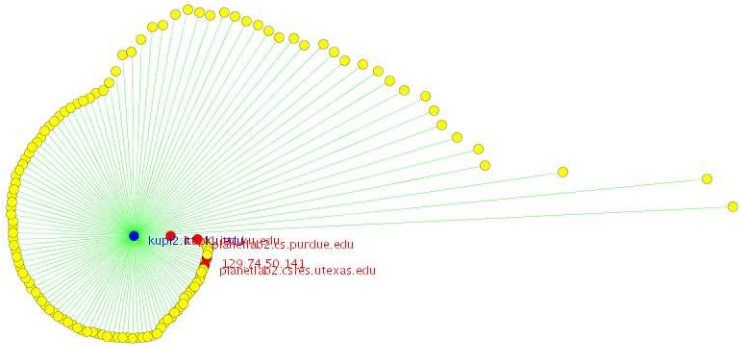

Fig. 3. Visualization of the clustering result: finding the top 5 closest nodes to the node "kupl2.ittc.ku.edu".

A. Evaluation Metrics

We define three metrics for comparing the performance of Netvigator with other proximity estimation schemes. For each node i of the N participating nodes, let $S^i_{e,k}$ denote the set of k closest nodes *estimated* by the scheme. We also collected the N^2 `ping` measurements to find the actual closest nodes. These ping measurements are for verification purposes only to comparatively evaluate the performance of the algorithms. Let $S^i_{a,k}$ denote the corresponding set of k *actual* closest nodes. Let c_i denote the actual closest node to node i.

- **Accuracy**: It measures whether the actual closest node was returned in the proximity set $S^i_{e,k}$.

$$a(i) = \begin{cases} 1 & \text{if } c_i \in S^i_{e,k} \\ 0 & \text{o.w.} \end{cases}$$

Mean accuracy is represented as:

$$acc_k = \sum_{i=1}^{N} a(i)/N$$

- **Precision**: It measures the overlap between the k actual closest nodes and the k closest nodes computed by the proximity scheme. Mean precision averaged over all N nodes for a given k is defined as:

$$prec_k = \frac{\sum_{i=1}^{N} \frac{|(S^i_{a,k} \cap S^i_{e,k})|}{k}}{N}$$

[6]We thank Jonathan Ledlie for providing us with the Vivaldi implementation.

- **Penalty**: It evaluates the potential cost due to inaccurate proximity estimation. Relative penalty for node i can be represented as:

$$penalty_{k,i} = \frac{(min_{\forall s \in S^i_{e,k}} dist(s,i)) - dist(c_i,i)}{dist(c_i,i)}$$

The numerator in the above equations is the absolute penalty. The average penalty is computed over all N nodes. The penalty metric depends on the topology as well as the location of the client nodes.

As we mentioned earlier, min_sum can also be used for estimating latency between different nodes. For min_sum, we also provide distance estimation results and compare with those obtained from GNP and Vivaldi. The quality of the latency estimation is evaluated using the two metrics given below.

- **Directional Relative Error**: This is the same metric that was used in [16]. The directional relative error for the distance between nodes n and c, $DRE(n,c)$ is given by

$$DRE(n,c) = \frac{\widehat{dist}(n,c) - dist(n,c)}{min(\widehat{dist}(n,c), dist(n,c))}$$

The ideal value of DRE(n, c) is 0, which indicates that the estimation is perfect with no estimation error. A DRE value of $(-)1$ indicates that the estimated latency is (smaller) bigger by a factor of 2. Also, we use the metric Relative Error, $RE(n,c)$, which is just the absolute value of the $DRE(n,c)$.

- **Mean Absolute Distance Estimation Error**: This metric computes the mean absolute distance estimation error over all N*(N-1) node pairs. The estimation error for the distance between nodes n and c is given by

$$err(n,c) = \widehat{dist}(n,c) - dist(n,c)$$

The mean absolute distance estimation error:

$$\overline{E} = \frac{\sum_{\forall n,c \in C, n \neq c} |err(n,c)|}{N(N-1)}$$

We also compute the standard deviation of E.

With N end nodes, the brute force method to obtain 100% accuracy and precision would be to conduct $N*(N-1)$ direct `ping` measurements. Each node conducts a `ping` measurement to every other $(N-1)$ nodes. In Netvigator, each node conducts `traceroute` measurements to L landmarks plus k `ping` measurements (where k is the number of candidate nodes estimated by the scheme), with a total of $N*(L+k)$ measurements. It is $N*(N-L-k-1)$ less than $N*(N-1)$ ping measurements. In GNP, each node conducts $(L+k)$ `ping` measurements, with a similar net savings of $N*(N-L-k-1)$ measurements. Though the message overhead at the end node for a ping probe is constant, the message overhead for a traceroute probe is proportional to length of path from probing node to the landmark. If H is the average number of hops between end nodes and landmark nodes, the total number of probe messages sent from all end nodes is $N*(L*H+k)$.

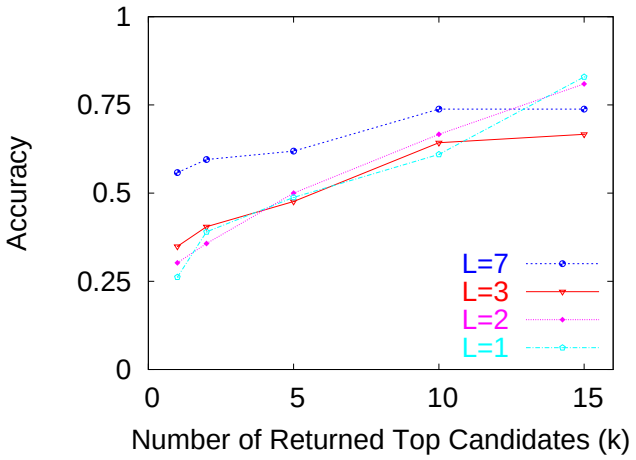

Fig. 4. Accuracy of min_sum (m0) in HP intranet.

In the next two sections, we present our proximity estimation results on an enterprise network and PlanetLab respectively. For a more controlled environment, we also conduct large scale simulations and these results are presented in Section III-D. For the sake of brevity, the results of our distance estimation are condensed and presented in Section III-E.

B. Enterprise Network

We ran our experiments on a well managed and provisioned enterprise intranet at Hewlett-Packard. We selected 43 globally distributed end-hosts for the experiments. $NetIQ^{TM}$ Chariot [5] performance endpoints were installed on these hosts running Linux or Microsoft windows operating systems. The `traceroute` capability of the endpoint was used to collect path information when remotely triggered via the Chariot console. Complete router-aliasing information about the enterprise network was available from the management servers. The endpoints were also used for conducting 43^2(=1849) ping measurements required for algorithm verification.

We assigned up to 7 landmark nodes among 43 end hosts in this set of experiments. The landmark nodes were selected manually with some coarse geographical input. We varied the number of landmark nodes (1, 2, 3, and 7 nodes) and computed the metrics for the 43 end nodes as a function of k, the number of top closest candidates returned by the proximity estimation algorithm.

We use our three clustering algorithms— min_sum, max_diff and $inner\_product$ for Netvigator. We denote these as $m0$, $m1$, and $m2$ in the plots. When the router aliasing information is incorporated in the solution, we denote the schemes as $m0\_R$, $m1\_R$, and $m2\_R$ respectively.

In Figure 4, the mean accuracy is plotted against k for the min_sum ($m0$), as the number of landmarks (L) is varied. For each of the four curves, as k is increased, the mean accuracy improves. The trend is less clear as L is increased, except for very low values of k when having 7 landmark nodes is clearly better than other cases. In fact, with $k = 1$, i.e., when the Netvigator returns just one candidate, the scheme returns the actual closest node in over 50% of the cases when $L = 7$. As

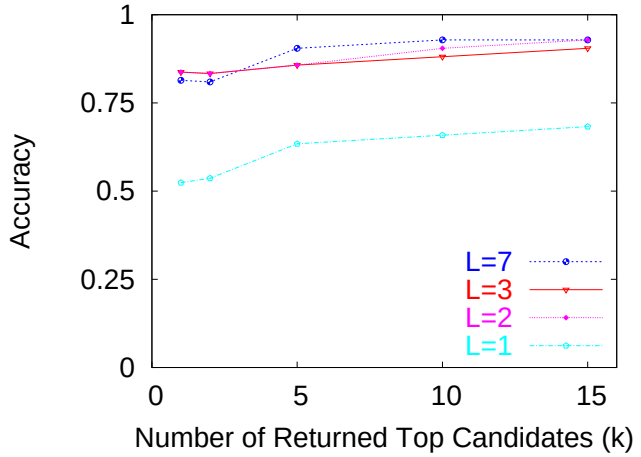

Fig. 5. Accuracy of min_sum with routing alias information (m0_R) in HP intranet.

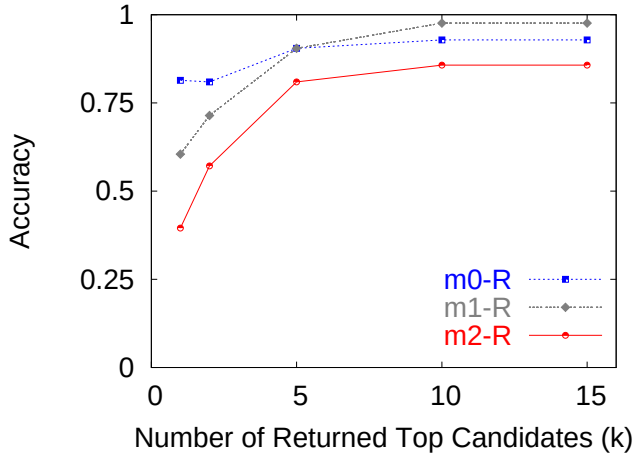

Fig. 7. Accuracy of min_sum (m0_R), max_diff (m1_R), and $inner\_product$ (m2_R) in HP intranet.

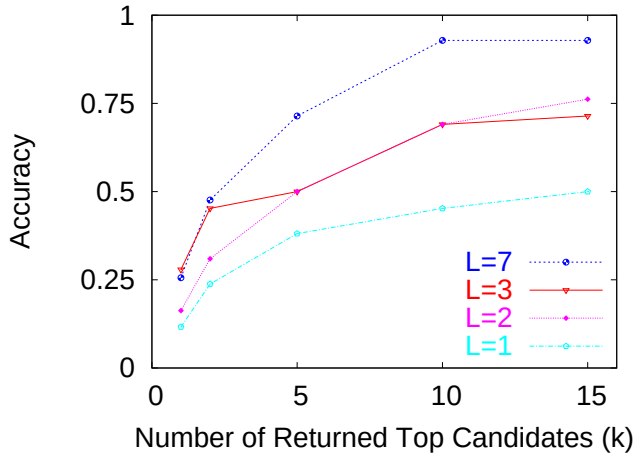

Fig. 6. Accuracy of GNP in HP intranet.

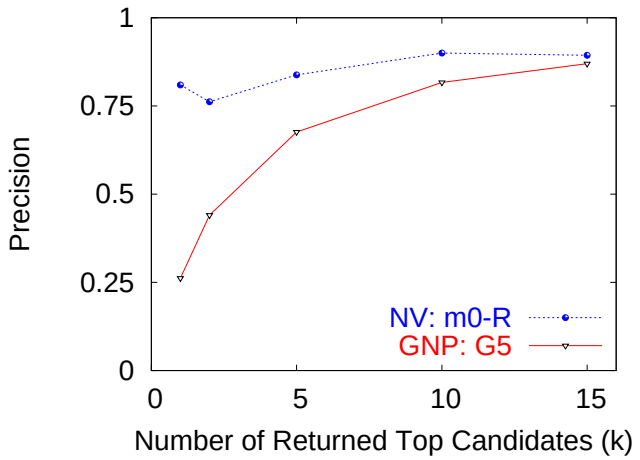

Fig. 8. Precision of min_sum (m0_R) and GNP in HP intranet.

k is increased to 15, the mean accuracy for all four landmark values is around 75%.

In Figure 5, we explore the effect of using the router aliasing information for the min_sum clustering ($m0\_R$) as the number of landmark nodes is varied. Compared with Figure 4, the mean accuracy is significantly improved when the router aliasing information is incorporated in the clustering algorithm. With $L = 7$, over 90% accuracy is achieved with just 5 top candidates ($k = 5$). Furthermore, while the single landmark scenario improves only slightly, the performance with L=2, 3, and 7 landmarks improves significantly and are comparable to each other. This is of great benefit as the fewer the number of landmark nodes, the less the measurement overhead.

Figure 6 plots the mean accuracy for the GNP method as L and k are varied. Two observations from the result: having more landmark nodes shows clear advantage and the accuracy is very poor when k is small. These results indicate that for GNP to have the same accuracy as Netvigator, the overhead has to be significantly higher than Netvigator. Comparing Figures 5 with 6, with 5 candidates, Netvigator has an accuracy of 90.7% (L=7) and 86.05% (L=2,3), while GNP has an accuracy of only 72.09% (L=7).

In Figure 7, we compare the different clustering algorithms for the Netvigator. With 7 landmarks and using the router aliasing information, $m0\_R$, $m1\_R$ and $m2\_R$ are compared with each other as k is increased. We find that the $inner\_product$ ($m2\_R$) algorithm performs the worst, while the performance of $m0\_R$ and $m1\_R$ are similar. Although the accuracy of $m1\_R$ is the highest when k is large, $m0\_R$ has the advantage of providing a higher accuracy at low values of k (which reduces the total overhead). Thus in the remainder of the paper, we only show the results for the min_sum ($m0$) algorithm.

In Figures 8 and 9, the mean precision and mean penalty incurred for Netvigator $m0\_R$ and GNP (with dimension 5: G5) are compared for the 7 landmarks case. The absolute penalty numbers are in milliseconds. As expected, the precision values increase with k, while the penalty numbers decrease with k. We find that Netvigator outperforms GNP in both metrics.

In summary, although the accuracy and precision of Netvigator are not 100%, with L=7, it incurs a significantly less measurement overhead of about 15% (of what the brute force 100% accurate method would take), and provides over 90% accuracy, with a low penalty. The reason for the superior performance of the Netvigator over existing schemes stems from the fact that

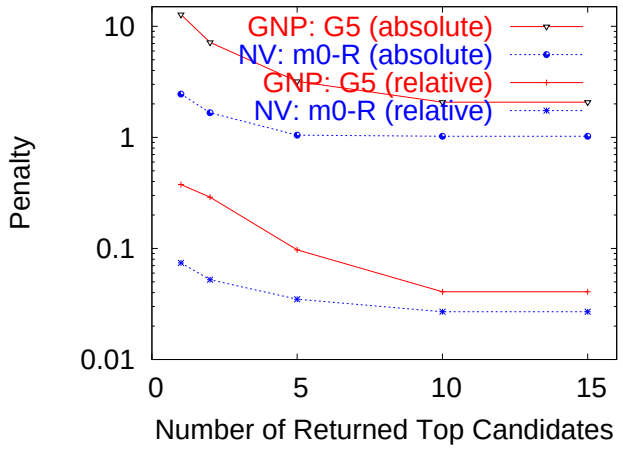

Fig. 9. Penalty of min_sum (m0_R) and GNP in HP intranet.

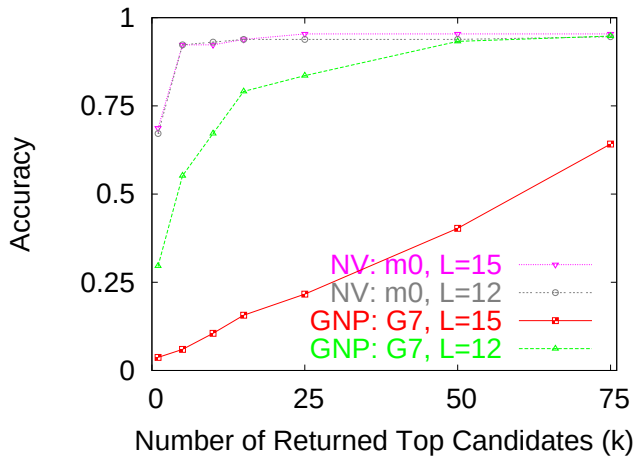

Fig. 10. Accuracy of min_sum (m0) and GNP in PlanetLab.

it uses a lot of additional information that is easily available without incurring high extra overhead. Utilizing the additional topology information provides robustness to incomplete measurements or measurement errors and also provides high accuracy, high precision, and low penalty. These features of Netvigator become even more apparent in the next section when we discuss an environment that is less predictable than the enterprise intranet.

C. PlanetLab

We also conducted experiments on unmanaged, poorly instrumented open Internet using PlanetLab [19]. A set of 131 Linux nodes with 15 landmark nodes was selected for the PlanetLab experiments. The 15 landmark nodes were selected with approximate global geographical representation to the extent possible.

In Figure 10, the mean accuracy is plotted against the number of candidates returned for Netvigator ($m0$) and GNP (dimension 7, denoted as G7). With L=15 landmark nodes, Netvigator performs very well, achieving over 90% accuracy with just 5 candidates. However, GNP performs very poorly with L=15. We found that of the 15 landmark nodes, 3 landmarks had partially missing measurement data and this affected the GNP results adversely while Netvigator was not affected. When we reran the experiment with only L=12 landmarks which had more complete measurement data, the performance of GNP improved dramatically, although it was still comparatively poor to Netvigator especially when k is small. Our experiments show that a small number of bad measurement data can cause the estimation quality of GNP to degrade more than 300%. It is interesting to note that the performance of Netvigator with either number of landmarks was approximately the same. This demonstrates the strong robustness of Netvigator to bad measurements. Realistically on the Internet, it is difficult to get a good and consistent set of measurement data at all times and hence proximity estimation algorithm needs to work well in spite of these measurement issues. We expect Vivaldi and GNP to perform approximately the same with respect to the proximity estimation metrics. Figure 11 compares the proximity estimation accuracy of Netvigator with GNP and Vivaldi. GNP and Vivaldi perform approximately the same, with GNP performing a little better than Vivaldi. Thus, to keep the graphs clearer, we do not report the proximity metrics for Vivaldi in the remaining graphs.

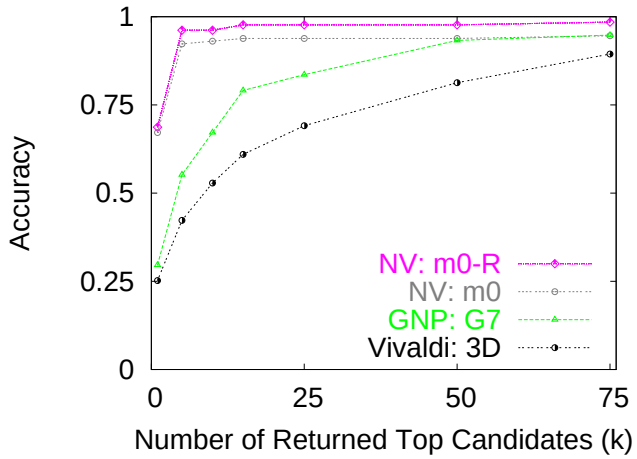

Fig. 11. Accuracy of min_sum (m0 and m0_R) and GNP in PlanetLab.

As shown from the results in the enterprise network, the router aliasing information improves clustering accuracy. However, unlike the HP enterprise network where we were able to get access to a complete router aliasing database, on the open Internet, this information is hard to obtain. Using the *scriptroute sr-ally* [21], we attempted to get as much information as possible. Due to the large number of router interfaces encountered, the process was very slow and because of non-responsive routers, the resulting aliasing information was incomplete. Nevertheless, we used what aliasing information we had to run the $m0\_R$. Figure 11 shows that with the L=12 landmarks, using the router aliasing information does give a boost to the mean accuracy, having over 90% accuracy with just 2 candidates.

Figures 12 and 13 show the precision and relative penalty for Netvigator ($m0\_R$) and GNP (dimension 7). Netvigator outperforms GNP in both metrics. It is interesting to compare Figure 13 with Figure 9. The penalty for Netvigator is very similar in both experiments. GNP however, shows much higher penalty in PlanetLab experiments.

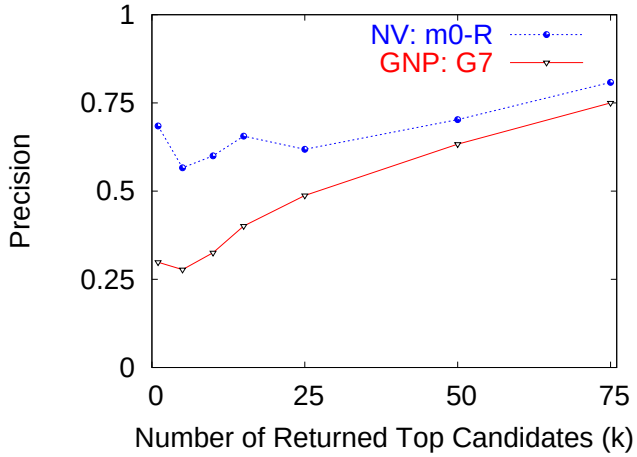

Fig. 12. Precision of min_sum (m0_R) and GNP in PlanetLab.

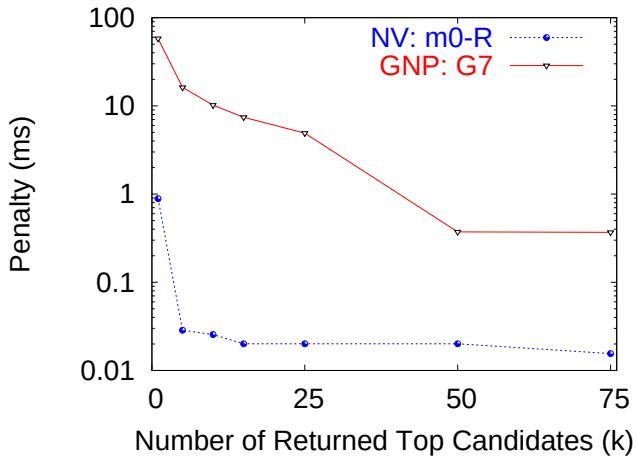

Fig. 13. Penalty of min_sum (m0_R) and GNP in PlanetLab.

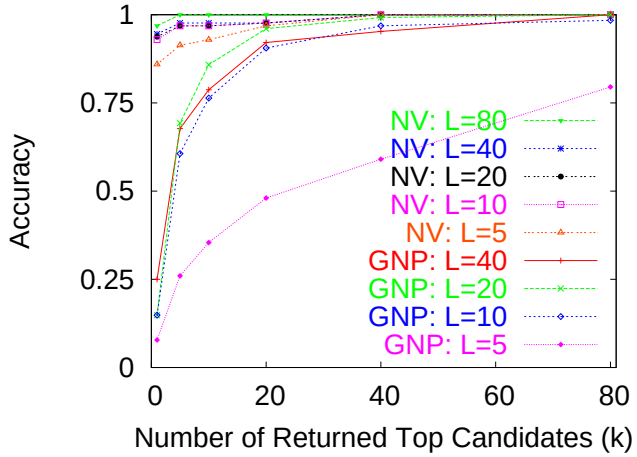

Fig. 14. Accuracy of min_sum (m0) and GNP with N=128.

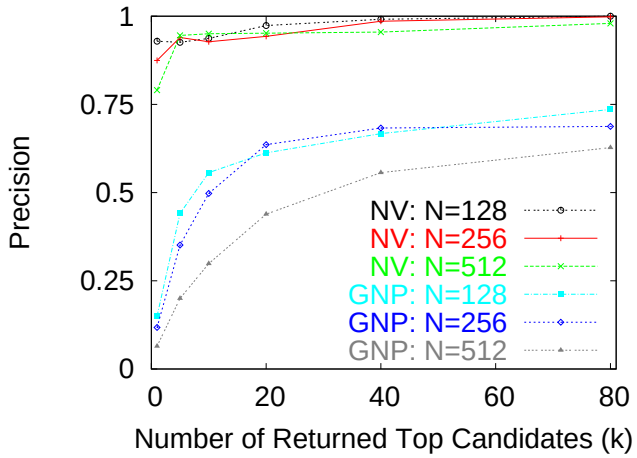

Fig. 15. Precision of min_sum (m0) and GNP with L=20.

Due to `traceroute` problems described above, we encountered cases where no milestone was found on the paths to the landmarks. This problem can be addressed by active probing from the milestones, though this can cause a higher overhead. Each of the milestones also measures its distances to the landmarks to obtain its own landmark vector. The milestones can use the landmark vector as the DHT key to report themselves to the global information table. The global information table then instructs nodes to measure to a set of milestones that the global information table has identified to be close to each of the nodes, using a landmark clustering scheme such as GNP.

D. Simulation

In addition to the experiments on the operational networks, we conducted large scale simulations to test the effectiveness of Netvigator and compare it with GNP. The simulation enables us to consider larger number of nodes (10,000 node topology), but it is important to realize that the simulation also represents some unrealistic ideal settings (e.g., complete measurements, no router aliasing problem, etc.).

We conduct a simulation study on a transit-stub topology produced using GT-ITM [4] with approximately 10,000 nodes. This topology has 25 transit domains, 5 transit nodes per transit domain, 4 stub domains attached to each transit node, and 20 nodes in each stub domain.

We considered 128, 256, and 512 end nodes (randomly chosen) whose k closest nodes are to be determined. We randomly picked L=5, 10, 20, 40, 80 landmark nodes. We ran Netvigator (with min_sum) and GNP (with dimension 7) and computed the performance metrics. Due to space constraints, we show only subset of results.

Figure 14 contains the mean accuracy for Netvigator and GNP. For Netvigator, the larger the number of returned candidates, the better the accuracy, reaching 100%. The number of landmark nodes has little effect on the accuracy. Even with only 10 landmark nodes and just single top candidate identified, the accuracy is 92.97%. On the other hand, the GNP results vary widely as k and L are increased, thus demonstrating the high sensitivity to these parameters. At low values of k, the accuracy is significantly worse than that obtained using Netvigator.

Figures 15 and 16 show the precision and relative penalty in L=20 landmark nodes and 128, 256, and 512 end nodes for Netvigator and GNP. The precision values are very high, close to 100% for Netvigator, while GNP only achieves about 70% at best. The penalty values for Netvigator are lower than that of GNP, although at high values of k, the values get similar.

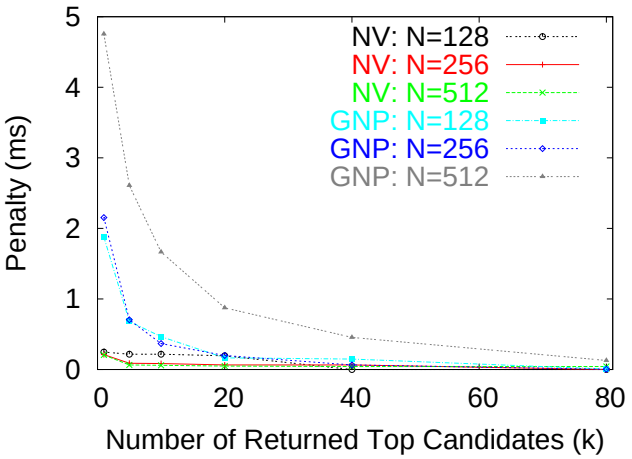

Fig. 16. Penalty of min_sum (m0) and GNP with L=20.

TABLE II

NETVIGATOR: STRETCH INCREASE (%) COMPARED WITH THE MINIMUM POSSIBLE VALUE.

N	stretch increase (%)	min k
512	1.8	1
1,024	3.4	2
2,048	5.8	3

We also performed simulations to demonstrate how Netvigator can be used to efficiently construct a high-quality application-level multicast structure such as Host Multicast Tree Protocol (HMTP) [33] and Service Adaptive Multicast (SAM) [3]. These multicast protocols rely on network proximity information for efficient multicast tree creation.

The results reported in Table II show the percentage increase of the stretch values over the minimum possible value using Netvigator.[7] It also shows the minimum number of candidates (k) required to compute the closest node. We observe the stretch values are within 6% of the optimal value. In most cases, only one RTT measurement was needed to locate the nearest node as the DHT infrastructure performs accurate computation, and no cases required more than 10 measurements. All of these results were obtained using only 15 landmark nodes.

E. Latency Estimation Results

In this section, we present a subset of our results on latency estimation on PlanetLab. The results for the enterprise network are similar and are omitted for the sake of brevity. We use the same datasets on which proximity estimation was tested. As mentioned before, only the min_sum clustering technique produces a distance estimate in Netvigator. We compare the accuracy of the distance estimation of Netvigator (L=12 landmarks) with that of GNP (L=12 landmarks, G=7 dimensions) and Vivaldi (3-dimensions) on the same PlanetLab measurement dataset. As mentioned before, in the real world, obtaining a complete set of measurements on a platform such as PlanetLab is not guaranteed. Thus, for a small fraction of node pairs

[7] *Stretch* is the ratio of the tree cost (the sum of link delays) to that of a minimal spanning tree.

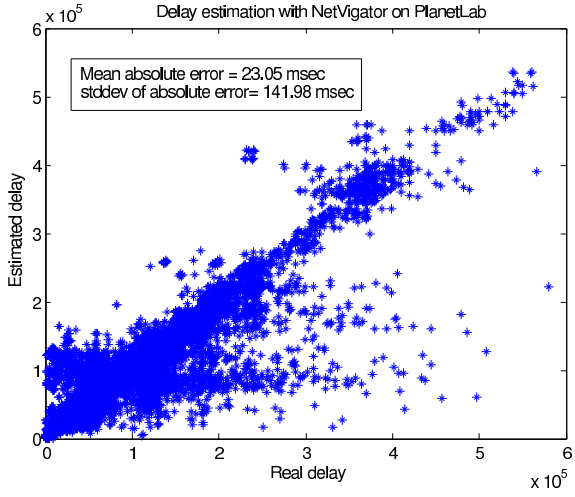

Fig. 17. Estimated vs. actual delay on PlanetLab using Netvigator, L=12

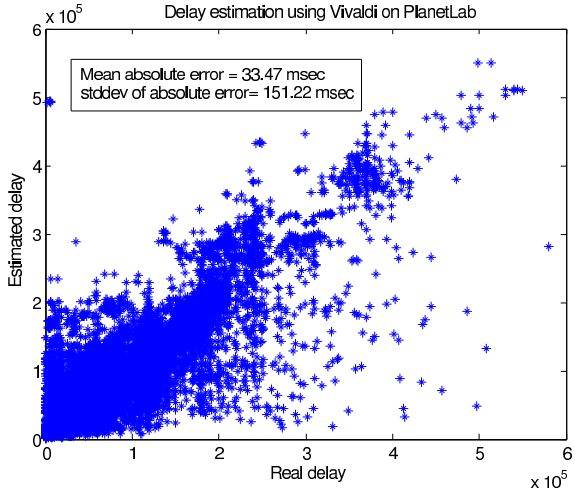

Fig. 18. Estimated vs. actual delay on PlanetLab using Vivaldi

(less than 12%) we do not have the actual measured latency and these pairs are omitted from the evaluation of distance estimation accuracy.

Figures 17, 18 and 19 contain plots of the estimated delay versus the actual measured delay for Netvigator, Vivaldi and GNP respectively. The units for the axes in these plots are in microseconds. In these scatter plots, the closer the points plotted are to the diagonal, the better is the estimation. In each case, the mean and standard deviation of the absolute error (E) as defined earlier is listed with the corresponding plot. The estimation with Netvigator is the best, with a mean absolute estimation error of 23 msec, followed by Vivaldi and GNP in this order.

The directional relative errors (DRE) for Netvigator, Vivaldi and GNP are plotted against the actual measured delay in Figures 20, 21 and 22. The mean and variance of the relative error (RE) computed over all node pairs with available measurements is also listed with each plot. For Netvigator, the DRE ranges between -30 and 60, whereas for both Vivaldi and GNP, the DRE range is much larger. For the sake of easy visual comparison, all three plots are shown with the same range of the axes as used in the Netvigator plot. Again, it is clear that Netvigatorexhibits much better latency estimation performance than the other two

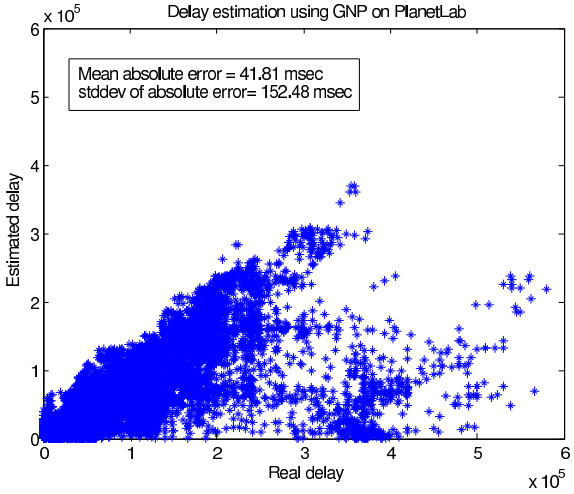

Fig. 19. Estimated vs. actual delay on PlanetLab using GNP, L=12, G=7

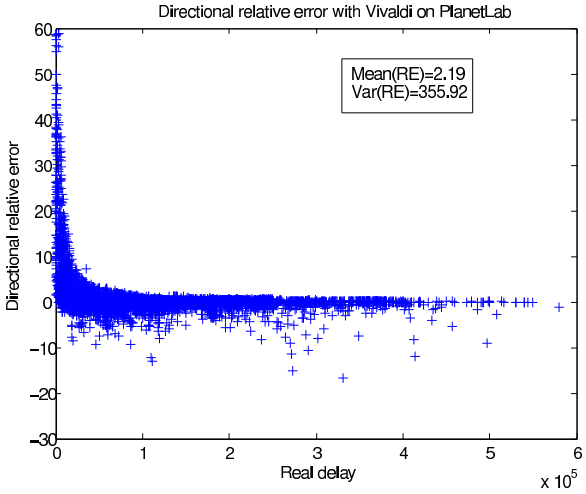

Fig. 21. Directional relative error on PlanetLab using Vivaldi

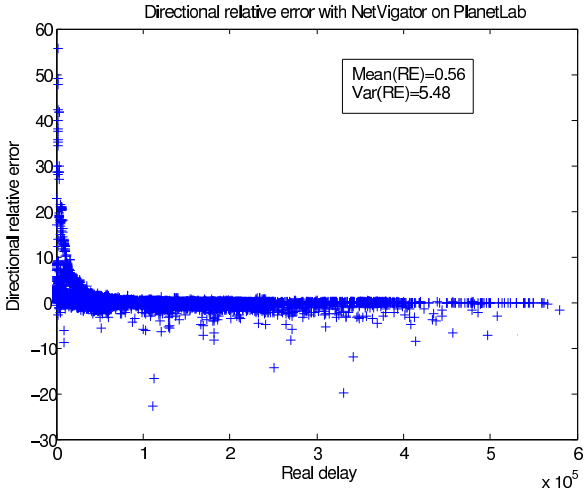

Fig. 20. Directional relative error on PlanetLab using Netvigator, L=12

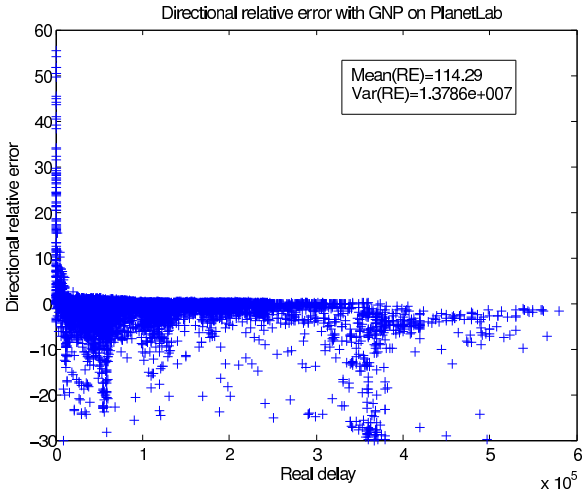

Fig. 22. Directional relative error on PlanetLab using GNP, L=12, G=7

schemes with an average RE of just 0.56. Vivaldi is the next best performer with an average RE of 2.19. For this dataset, GNP performs very poorly with an average RE of 114.29.

IV. Discussion

A. Reducing Measurement Overhead

As discussed earlier, each `traceroute` probe incurs message overhead proportional to the length of the probed path. To reduce the measurement overhead we propose "smart-traceroute," a smarter version of `traceroute` that does not probe every hop. Instead of incrementing the TTL of the ICMP probe packet by 1, smart-traceroute intelligently skips intermediate hops. It exploits the fact that fine grained delay information is primarily needed for the milestones closer to the node. We used two heuristics *exp* and *hop* as skipping patterns for smart-traceroute. The *exp* exponentially increases `traceroute` TTL and probes only hops 1, 2, 4, 8, etc., whereas the *hop* mimics the hierarchical network structure and uses slowly increasing probing TTL (1, 2, 3, 6, 9, 12,. . .). If the largest path is 32, the message overhead for each smart-traceroute probe is less than 6 for *exp* and 12 for *hop*. Figure 23 compare these two smart hop skipping techniques on PlanetLab. The baseline case of $m0\_R$ is also shown.

We observe from the graph that with less measurement overhead, both *exp* and *hop* perform similarly and are only slightly less accurate than the baseline $m0\_R$. This implies that the information not used in *exp* and *hop* was not adding much value.

We plan to explore this further to find how the skipping patterns for smart-traceroute can be adapted to given topologies. In addition, we will enable smart-traceroute to probe multiple targets simultaneously. This smart version will have the following features: (i) Using heuristics, it avoids probing the common paths multiple times. (ii) It reports data incrementally as it becomes available. (iii) It groups consecutive routers that are close to each other into super-routers to increase the probability of capturing common sub-path among different clients.

B. Blocking of ICMP Probe Messages

The widespread deployment and availability of traceroute made it the measurement tool of choice for our experiments. At the same time, we observed that traceroute is not a perfect path probing tool. Some network administrators disable the ICMP responses originating from their routers. This is more common in the open Internet to shield the routers from denial-of-service attacks. Hence, it was not always possible to get complete path information between the client nodes and the landmark nodes.

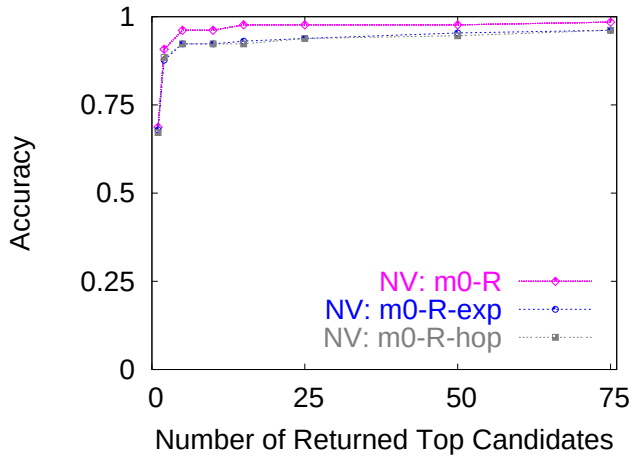

Fig. 23. Accuracy of m0_R, m0_R_exp, and m0_R_hop with L=12 on PlanetLab.

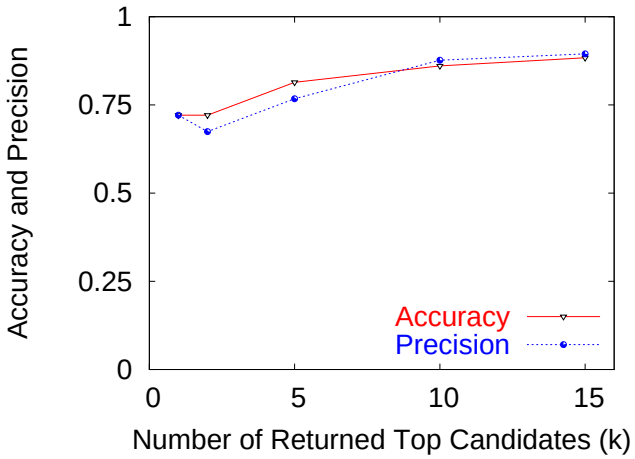

Fig. 24. Precision and Accuracy of Netvigator on HP Intranet using traceroutes initiated at landmark nodes

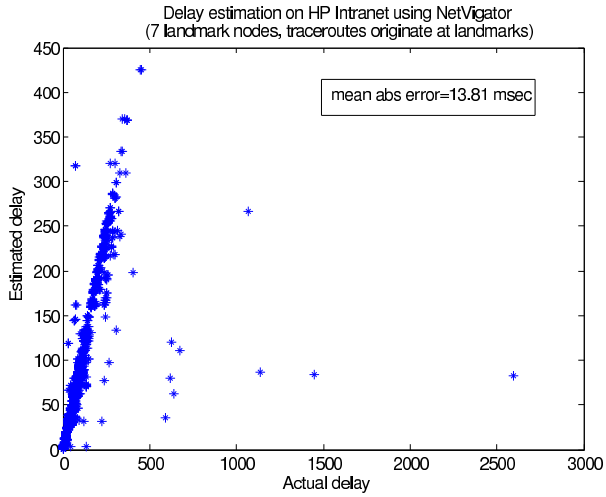

Fig. 25. Estimated vs. actual delay on HP Intranet using Netvigator with traceroutes initiated at landmark nodes

Some routers can also be configured to filter or rate control the ICMP responses to reduce the load on the routers. These configuration and policy changes result in specious data such as the case where the delay to the next hop in the path does not increase monotonically. Also, several routers do not process ICMP message in the fast path, leading to incorrect hop latencies. Some of the routers do not correctly decrement time-to-live field of the packet. Duplicate hops in the path are the result of such behavior. We also found that several routers in the enterprise intranet were configured to block ICMP packets. This is especially a problem for traceroute tool for the Windows^{TM} operating system that uses ICMP request packets for probing the path instead of UDP packets.

C. Measurements Initiated at the Landmark Nodes

The current Netvigator assumes that the traceroutes are initiated by the end client nodes to the landmark nodes. This is a good approach in a loosely managed heterogeneous/P2P environment where the identity of all nodes in the system is not known. Thus the end nodes have the onus of deploying the measurement scripts and running them periodically to probe the advertised well known landmark nodes. However, in a well managed enterprise network environment, from a deployment standpoint it is easier to give the identity of the end clients to a few infrastructure management nodes that can act as landmarks and carry out periodic traceroutes to the end clients. Netvigator can work well in the enterprise paradigm.

For the enterprise network, we ran experiments with the traceroutes being initiated at the landmark nodes instead of at the clients. Figures 24 and 25 contain results of the proximity as well as distance estimation using 7 landmark nodes and the traceroutes initiated at the landmark nodes. It is clear from these figures that initiating traceroutes at the landmark nodes does not cause deterioration of either the proximity estimation or the distance estimation. We plan on exploring this thread of work in more detail in the future.

D. Path Symmetry

In network coordinate-based systems (such as GNP and Vivaldi) that compute Euclidean distance metrics, the distances computed between any two nodes n and c is the same in either direction. However, path asymmetry exists in the real world. Netvigator does not a priori assume asymmetric paths and can provide different distance estimates in the two directions. This is because the common closest landmark/milestone may be different when computing from node n or from node c. However, Netvigator can be modified to take directional input, provided traceroutes to/from landmark nodes are available. Thus instead of N*L traceroutes, 2*N*L traceroutes would need to be conducted. This aspect of our work is currently ongoing.

E. Distributed Querying

Using a centralized server to store all information is not scalable as it requires all nodes to report and query a central unit. This can cause a concentration of network traffic and single point of failure. We are working on a distributed Internet distance information service to make Netvigator more robust and scalable. Our solution takes advantage of the locality of interest since nodes usually are interested in finding resources or services that are close by in network proximity. We propose to distribute the position data among a set of infrastructure nodes,

and have used a heuristics based approach for partitioning and querying this data [10]. Simulation results show that distributed approach is not only accurate but also results in shorter response time and better network resource utilization.

V. Related Work

Several schemes have been proposed to estimate Internet distances. Internet Distance Maps (IDMaps) [11] places tracers at key locations in the Internet. These tracers measure the latency among themselves and advertise the measured information to the clients. The distance between two clients A and B is estimated as the sum of the distance between A and its closest tracer A', the distance between B and its closest tracer B', and the distance between the tracers A' and B'. An algebraic approach can be used complimentary to IDMaps to compute distances between intermediate hops without extra probing [22]. The accuracy of IDMaps improves as the number of tracers increases. One problem with IDMaps is that a client node has to measure distances to all tracers to identify the closest tracer. To alleviate this problem, Dynamic Distance Maps (DDM) [28] is similar to IDMaps. DDM organizes the tracers into a hierarchy, and a client node traverses the hierarchy top-down to locate the closest tracer.

M-coop [24] utilizes a network of nodes linked in a way that mimics the autonomous system (AS) graph extracted from BGP reports. Each node measures distances to a small set of peers. When an estimate between two IP addresses is required, several measurements are composed recursively to provide an estimate. King [12] is similar in spirit to IDMaps and M-coop. It takes advantage of the existing DNS architecture and uses the DNS servers as the measurement nodes.

King, M-coop, IDMaps, and DDM all require that the IP addresses of both the source and the destination are known at the time of measurement. Therefore, they cannot be used when the IP address of the target node is unknown. In scenarios where a client attempts to locate close-by Internet services, the target IP addresses of these local service nodes are usually not known in advance, and must be discovered. Furthermore, storing and retrieving the measurement information is a non-trivial task.

There are schemes that use landmark techniques for network distance estimation. Landmark clustering [16], [20] uses a node's distances to a common set of landmark nodes to estimate the node's physical position. The intuition behind this technique is that if two nodes have similar latencies to the landmark nodes, they are likely to be close to each other. There are several variations of landmark clustering. In landmark ordering [20], a node measures its round-trip time to a set of landmarks and sorts the landmark nodes in the order of increasing round-trip time (RTT). Therefore, each node has an associated order of landmarks. Nodes with the same (similar) landmark order(s) are considered to be close to each other. This technique however, cannot differentiate between nodes with the same landmark orders.

Another variation is GNP (Global Network Positioning) [16] and its sequel NPS (Network Positioning Systems) [17]. In this scheme, landmark nodes measure RTTs among themselves and use this information to compute the coordinates in a Cartesian space for each landmark node. These coordinates are then distributed to the clients. The client nodes measure RTTs to the landmark nodes and compute the coordinates for itself, based on the RTT measurements and the coordinates of the landmark nodes it receives. The Euclidean distance between nodes in the Cartesian space is directly used as an estimation of the network distance. Internet Iso-bar [6] also requires each node to measure distance to landmarks. It uses the distance information to form clusters. For each cluster, it assigns a monitoring agent for the center node. Each monitor measures the distance to all other cluster-heads and this latency is used to estimate distances between two hosts in different clusters.

GNP and Internet-Iso-bar require that all client nodes contact the same set of landmarks nodes, and the scheme may fail when some landmark nodes are not available at a given instant of time. To address this problem, Lighthouse [18] allows a new node wishing to join the network to use any subset of nodes that is already in the system (i.e., *lighthouses*) as landmarks to compute a global network coordinate based on measurements to these lighthouses. Mithos [30] and PCoord [29] are similar to lighthouse in that they do not require each node to measure distances to all the predetermined landmarks. PIC (Practical Internet Coordinates) [7] has been proposed to use a security test based on triangular inequality to filter out wrong coordinates/measurements from the malicious landmarks.

Virtual landmarks [27] and ICS (Internet Coordinate System) [15] are also landmark-based schemes. They however, use the PCA (Principal Component Analysis) of Lipschitz embedding, instead of Euclidean embedding, to obtain network positions. Compared with GNP, these schemes are computationally more efficient and require less dimensionality. BBS (Big-Bang Simulation) [23] on the other hand reduces Euclidean embedding error by iteratively simulating the error as potential force fields.

Despite the variations, current landmark clustering techniques share one major problem. It causes *false clustering* where nodes that have similar landmark vectors but are far away in network distance are clustered near each other.

Vivaldi [8] is another scheme that assigns coordinate space for each host, but it does not require any landmarks. Instead of using probing packets to measure latencies, it relies on piggybacking when two hosts communicate with each other. With the information obtained from passively monitoring packets (e.g., RPC packets), each node adjusts its coordinates to minimize the difference between estimates and actual delay. Although Vivaldi is fully distributed, it takes time to converge, requires applications to sample all nodes at relatively same rate to ensure accuracy, and packets need to add Vivaldi-specific fields.

Meridian [31], network location service, is similar in intent to Netvigator and is also focused on proximity estimation. Meridian forms a loosely-structured overlay network of concentric rings based on direct measurements. A scalable gossip protocol is used for exchanging information amongst the nodes. The query matching is performed by forwarding it towards the most appropriate node. As it was very recently proposed we have not been able to evaluate the performance of Meridian. We plan to compare the accuracy and overhead of Meridian with Netvigator.

VI. Conclusions

We presented *Netvigator*, a network proximity estimation tool that uses a novel enhanced landmark clustering technique to accurately locate the closest node to a given node.

Our clustering algorithms for finding the closest nodes utilize distance information from the landmarks as well as milestones encountered en route. This approach provides high accuracy as well as robustness to bad measurements. We developed a prototype of our scheme and evaluated it in the real world including on planet-lab as well as HP intranet. We also performed simulation for scalability testing. Our experiments show that with 90% confidence, our algorithm identifies the actual closest node.

Our future work will extend the capability of our tool to estimate network distance directly and further reduce measurement overhead. Another aspect of our future work is, with a distributed information table based on DHT, to apply our techniques to real applications such as constructing efficient service overlay networks. We also plan to install Netvigator as a running service on PlanetLab.

References

[1] S. Banerjee, M. Pias, and T. Griffin, "The interdomain connectivity of planetlab nodes," in *Proceedings of the PAM*, Sophia-Antipolis, France, April 2004.

[2] S. Banerjee, J. Brassil, A. Dalal, S.-J. Lee, E. Perry, P. Sharma, and A. Thomas, "Rich media from the masses," HP Laboratories, Tech. Rep. HPL-2002-63R1, March 2002.

[3] S. Banerjee, Z. Xu, S.-J. Lee, and C. Tang, "Service adaptive multicast for media distribution networks," in *Proceedings of the IEEE WIAPP*, San Jose, CA, June 2003.

[4] K. Calvert, M. Doar, and E. W. Zegura, "Modeling Internet topology," *IEEE Commun. Mag.*, vol. 35, no. 6, pp. 160–163, June 1997.

[5] Chariot, http://www.netiq.com/products/chr/default.asp.

[6] Y. Chen, K. H. Lim, R. H. Katz, and C. Overton, "On the stability of network distance estimation," *ACM Performance Evaluation Review*, vol. 30, no. 2, pp. 21–30, September 2002.

[7] M. Costa, M. Castro, A. Rowstron, and P. Key, "PIC: Practical internet coordinates for distance estimation," in *Proceedings of the IEEE ICDCS 2004*, Tokyo, Japan, March 2004.

[8] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proceedings of the ACM SIGCOMM*, Portland, OR, August 2004.

[9] Z. Duan, Z.-L. Zhang, and Y. T. Hou, "Service overlay networks: SLAs, QoS and bandwidth provisioning," *IEEE/ACM Trans. Networking*, vol. 11, no. 6, pp. 870–883, December 2003.

[10] R. Fonseca, P. Sharma, S. Banerjee, S.-J. Lee, and S. Basu, "Distributed querying of internet distance information," in *Proceedings of the 8th IEEE Global Internet Symposium*, Miami, FL, March 2005.

[11] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A global internet host distance estimation service," *IEEE/ACM Trans. Networking*, vol. 9, no. 5, pp. 525–540, October 2001.

[12] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *Proceedings of the ACM IMW*, Marseille, France, November 2002.

[13] J. Ledlie and M. Seltzer, "Stable and accurate network coordinates," Department of EECS, Harvard University, Tech. Rep., July 2005.

[14] Z. Li and P. Mohapatra, "QRON: QoS-aware routing in overlay networks," *IEEE J. Select. Areas Commun.*, vol. 22, no. 1, pp. 29–40, January 2004.

[15] H. Lim, J. C. Hou, and C.-H. Choi, "Constructing internet coordinate system based on delay measurement," in *Proceedings of the ACM IMC*, Miami Beach, FL, October 2003.

[16] T. S. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proceedings of the IEEE INFOCOM*, New York, NY, June 2002.

[17] ——, "A network positioning system for the internet," in *Proceedings of USENIX Annual Technical Conference*, Boston, MA, June 2004.

[18] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti, "Lighthouses for scalable distributed location," in *Proceedings of the IPTPS*, Berkeley, CA, February 2003.

[19] PlanetLab, http://www.planet-lab.org.

[20] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proceedings of the IEEE INFOCOM*, New York, NY, June 2002.

[21] Scriptroute, http://www.cs.washington.edu/research/networking/scriptroute.

[22] Y. Shavitt, X. Sun, A. Wool, and B. Yener, "Computing the unmeasured: An algebraic approach to internet mapping," *IEEE J. Select. Areas Commun.*, vol. 22, no. 1, pp. 67–78, January 2004.

[23] Y. Shavitt and T. Tankel, "Big-bang simulation for embedding network distance in euclidean space," in *Proceedings of the IEEE INFOCOM*, San Francisco, CA, April 2003.

[24] S. Srinivasan and E. Zegura, "M-coop: A scalable infrastructure for network measurement," in *Proceedings of the IEEE WIAPP 2003*.

[25] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proceedings of the ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[26] C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *Proceedings of the ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[27] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Proceedings of the ACM IMC*, Miami Beach, FL, October 2003.

[28] W. Theilmann and K. Rothermel, "Dynamic distance maps of the Internet," in *Proceedings of the IEEE INFOCOM*, Tel Aviv, Israel, March 2000, pp. 275–284.

[29] L. w. Lehman and S. Lerman, "PCoord: Network position estimation using peer-to-peer measurements," in *Proceedings of the IEEE NCA*, Cambridge, MA, August 2004.

[30] M. Waldvogel and R. Rinaldi, "Efficient topology-aware overlay network," in *Proceedings of the ACM HotNets-I*, Princeton, NJ, October 2002.

[31] B. Wong, A. Slivkins, and E. Sirer, "Meridian: A lightweight network location service without virtual coordinates," in *Proceedings of the ACM SIGCOMM*, Philadelphia, PA, August 2005.

[32] D. Xu and K. Nahrstedt, "Finding service paths in a media service proxy network," in *Proceedings of the SPIE MMCN*, San Jose, CA, January 2002.

[33] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," in *Proceedings of the IEEE INFOCOM*, New York, NY, June 2002.

[34] H. Zheng, E. K. Lua, M. Pias, and T. Griffin, "Internet routing policies and round-trip-times," in *Proceedings of the PAM*, Boston, MA, April 2005.

[35] Zoomgraph, http://www.hpl.hp.com/shl/projects/graphs.