Netvigator: Scalable Network Proximity Estimation

Zhichen Xu, Puneet Sharma, Sung-Ju Lee, Sujata Banerjee
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2004-28(R.1)
March 3, 2005*

E-mail: {zhichen,puneet,sjlee,sujata}@hpl.hp.com

network distance, proximity estimation, landmark clustering, Planet-Lab, network measurement

Network proximity 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 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 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.

* Internal Accession Date Only
Approved for External Publication

Netvigator: Scalable Network Proximity Estimation

Zhichen Xu, Puneet Sharma, Sung-Ju Lee, and Sujata Banerjee
Mobile & Media Systems Lab
Hewlett-Packard Laboratories
Palo Alto, CA 94304
{zhichen,puneet,sjlee,sujata}@hpl.hp.com

***Abstract*— Network proximity 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 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 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 closets 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.**

I. Introduction

The Internet provides a platform for various services and applications. An important recent trend is that users are no longer satisfied with receiving services that are targeted at mass audiences. They demand services that are tailored to individual needs. These services can potentially come from hundreds and millions of organizations and individuals that are connected to the Internet. With the proliferation of personalized services, 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 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 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.

Unfortunately, existing Internet proximity estimation tools have shortcomings. Many schemes require considerable amount of infrastructure support, and therefore have deployment issues. Such schemes provide a distance estimate between any two given nodes. They find the node closest to a client from a given set of potential targets by computing the distance to each node and picking the minimum. This process requires the client to obtain a list of all potential hosts providing a specific service from a directory service. In practice, when finding a service/resource, knowledge about 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. The important issue is whether the client can find a server that provides the right service with appropriate quality-of-service guarantee.

Landmark clustering [8], [11] 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. However, current landmark clustering techniques are prone to false clustering where distant nodes are clustered near each other. Further, the estimation quality of current landmark clustering schemes depends on the measurement quality 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 estimation tool. It uses an enhanced landmark clustering technique that is more efficient and accurate than the existing schemes. Our approach employs a small number of *landmarks* and a relatively large number of *milestones*. Each node performs round-trip time (RTT) measurements to landmarks and also records its distances to the milestones that the probing packets encounter en route to the landmarks. The milestones could be the routers encountered in a *traceroute* between the client node and a landmark node.

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. Utilizing distance information from both the landmarks and milestones makes our technique robust to bad measurements.
- We developed a prototype of our scheme and evaluated it on PlanetLab [10] and intranet of a large enterprise (Hewlett-Packard). We also performed a simulation study to evaluate its scalability. To the best of our knowledge, we are the first

to present an experimental report on network proximity estimation on PlanetLab. Our experiments show that with 90% confidence, the real closest node falls in the top two nodes our algorithm identifies.

The remainder of the paper is organized as follows. Section II surveys the related work. Section III describes the details of Netvigator, followed by experimental results in Section IV. Concluding remarks and directions for future work are presented in Section VII.

II. Related Work

Several schemes have been proposed to estimate Internet distances. Internet Distance Maps (IDMaps) [6] 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'.

M-coop [13] 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 [7] takes advantage of the existing DNS architecture and uses the DNS servers as the measurement nodes.

King, M-coop, and IDMaps 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.

There are schemes that use landmark techniques for network distance estimation. Landmark clustering [8], [11] 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. *Landmark ordering* is used in topologically-aware Content Addressable Network (CAN) [11]. With landmark ordering, 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) [8]. 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.

GNP requires 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 [9] 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.

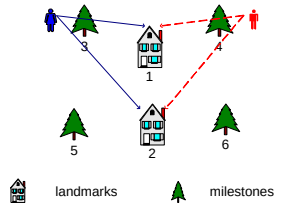


Fig. 1. Example of the enhanced landmark scheme.

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. To solve this problem, [16] for example, uses landmark clustering as a preselection process to identify nodes that are potentially close to a given node, and actual round-trip time measurements are used to identify the closest node. The effectiveness of this approach, unfortunately, is dependent on the topology being tested.

Vivaldi [5] 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.

III. Network Proximity Estimation

Before describing our enhanced landmark clustering technique 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.

The details of the enhanced landmark scheme are described below. 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.

(1) Each node sends probe packets to the landmarks for round-trip time measurement.[1] 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 k and identify a very small number of top candidates.[2]

(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.

Compared with the strategy that blindly increases the size of the landmark set and/or the candidate set for RTT measurements to increase accuracy, our scheme has three advantages. First, each client needs to probe only a small number of landmark nodes. Second, the milestones have accurate information of the local network characteristics. Third, 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 assume that the *triangle inequality* works reasonably well in the network topology. For each node, the k candidates having the smallest clustering metric values are picked as the k closest candidates.

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

[2] Section III-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.

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 node n and its candidate node set C, $min\_sum(n,C)$ is formally defined as

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

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.[3] 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 [15]. 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

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

Our experimental results in the following section will show that the algorithms min_sum and max_diff perform better than the $inner\_product$ algorithm. 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 $inner\_product$.

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

IV. Performance Evaluation

We prototyped our network proximity estimation system, *Netvigator*, based on the algorithms described in the previous section. 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. 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.

We experimented with Netvigator on two different *real* networks: a geographically distributed HP network and a set of *PlanetLab* nodes connected via open Internet [10].

Netvigator computes k closest nodes for each participating node. 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 [8], which has been shown to outperform other schemes such as IDMaps. For GNP, the k closest nodes for each participating node were computed based on the Euclidean distance calculation from the network coordinates GNP 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 both Netvigator and GNP, the end-nodes perform direct `ping` measurements to these k nodes to find the closest target.

A. 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:

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

- **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.

B. Enterprise Network

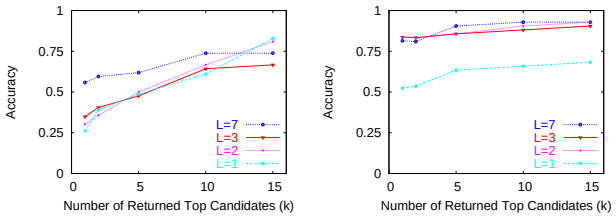
We ran our experiments on a well managed and provisioned enterprise intranet at Hewlett-Packard. We selected 43 globally distributed end-hosts for the experiments. $NetIQ^{TM}$ Chariot [4] performance endpoints were installed on these hosts running Linux or Microsoft windows operating systems. information 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.

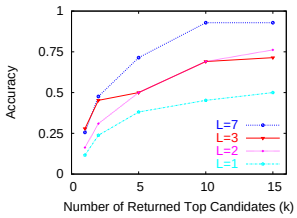
In Figure 2 (a), the mean accuracy is plotted against k for the min_sum ($m0$), as the number of landmarks (L) is varied. For each of the four curves, as k is increased, the mean accuracy improves. The trend is less clear as L is increased, except for very low values of k when having 7 landmark nodes 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 k is increased to 15, the mean accuracy for all four landmark values is around 75%.

In Figure 2 (b), 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 2 (a), 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

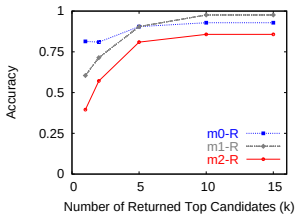


(a) Accuracy of min_sum (m0).

(b) Accuracy of min_sum with routing alias information (m0_R).



(c) Accuracy of GNP.



(d) Accuracy of clustering algorithms.

Fig. 2. Accuracy results from the HP intranet.

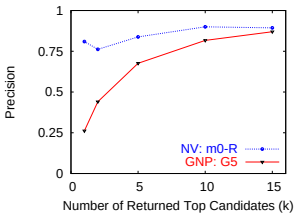
top candidates ($k = 5$). Furthermore, while the single landmark scenario improves only slightly, the performance with L=2, 3, and 7 landmarks improves significantly and are comparable to each other. This is of great benefit as the fewer the number of landmark nodes, the less the measurement overhead.

Figure 2 (c) 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 2 (b) with 2 (c), 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).

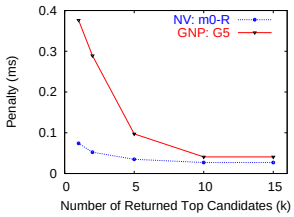
In Figure 2 (d), we compare the different clustering algorithms for the Netvigator. With 7 landmarks and using the router aliasing information, $m0\_R$, $m1\_R$ and $m2\_R$ are compared with each other as k is increased. We find that the $inner\_product$ ($m2\_R$) algorithm performs the worst, while 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 3 (a) and 3 (b), 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 it uses a lot of additional information that is easily available without incurring high 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.

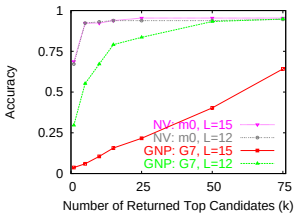


(a) Precision.

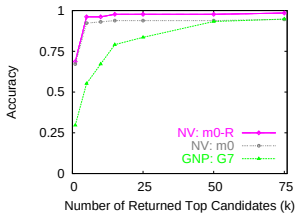


(b) Penalty.

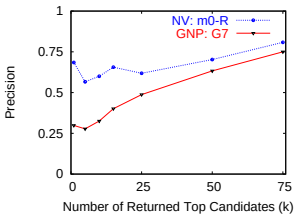
Fig. 3. Precision and penalty results from the HP intranet.



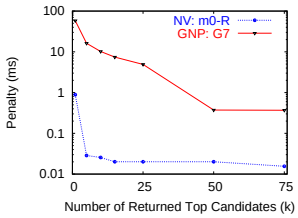
(a) Accuracy of min_sum (m0) and GNP.



(b) Accuracy of min_sum (m0 and m0_R) and GNP.



(c) Precision of min_sum (m0_R) and GNP.



(d) Penalty of min_sum (m0_R) and GNP.

Fig. 4. Measurement results from the planet-lab.

C. PlanetLab

We also conducted experiments on unmanaged, poorly instrumented open Internet using PlanetLab [10]. 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.

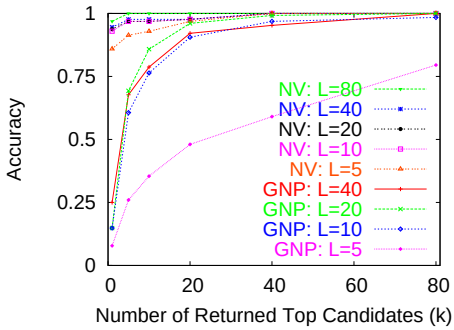
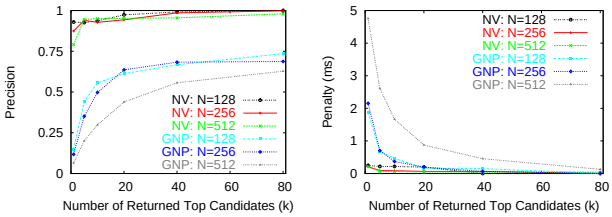


Fig. 5. Accuracy of min_sum (m0) and GNP with N=128.



(a) Precision. (b) Penalty.

Fig. 6. Simulation results with L=20.

In Figure 4 (a), 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 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.

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* [12], 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 4 (b) 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 4 (c) and 4 (d) 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 4 (d) with Figure 3 (b). The penalty for Netvigator is very similar in both experiments. GNP however, shows much higher penalty in PlanetLab experiments.

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 [3] 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 5 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.

Figure 6 shows 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.

V. Case Study

As a case study for our network proximity estimation technique, we demonstrate how Netvigator can be used to efficiently construct a high-quality application-level multicast structure such as Host Multicast Tree Protocol (HMTP) [17] or Service Adaptive Multicast (SAM) [2]. These multicast protocols rely on network proximity information for efficient multicast tree creation. Note that we are applying our scheme to overlay networks for a case study but it can be used for Internet proximity estimation.

We compare the quality of trees constructed using GNP and those constructed using Netvigator. We conduct a simulation study using the two transit-stub topologies as used in Section IV-D. We evaluate the tree quality using the *stretch* metric.

TABLE II
GNP: STRETCH INCREASE (%) COMPARED WITH THE MINIMUM POSSIBLE VALUE.

N	small-transit	large-transit
512	2.36∼12.43	4.7∼19.4
1,024	6.32∼22.6	7.37∼24.58
2,048	9.6∼47.9	10.50∼28.4

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

N	small-transit	min k	large-transit	min k
512	1.8	1	1.7	6
1,024	3.4	2	2.9	5
2,048	5.8	3	4.8	3

Stretch is defined as the ratio of the tree cost (the sum of link delays) to that of a minimal spanning tree.

A. GNP Results

For GNP, we use the k-mean method (described in [8]) to select L landmarks out of the 200 nodes. With only the information of the distances to the L landmarks, we identify the set of k closest candidates. We then perform RTT measurement to select the closest node among the k candidates. The overhead involved in the measurements is therefore on the order of $O(k + L)$.

To show how measurement overhead affects the resulting tree quality, we vary the number of candidate nodes (k) returned k = 10, 20, 30 and the number of landmark nodes (L) L = 20, 50, 100, 150. We consider three overlay network sizes for the small-transit topology, with the number of nodes N equal to 512, 1,024, and 2,048 nodes. As a baseline, we also consider the extreme and impractical case, i.e., $k = N$, where N is the number of overlay nodes. This case is equivalent to an exhaustive search for the closest node, and represents the best possible case.

The results of our evaluation is summarized in Table II, where we provide the range of percentage values with which the stretch may be larger than the minimum possible value. For example, for the 2,048-node overlay in the small-transit topology, depending on the combination of L and k, the stretch varies from 9.6%∼47.9% larger than the minimum possible value.

The number of landmark nodes L and the number of candidates k impact the overall accuracy of the node proximity measure. The clustering mechanism in [8] may not be sufficiently robust for all topologies. It is non-trivial to select the optimal combination of L and k for the best results. Careless selection of the combinations may produce results that are up to 47.9% worse than the best case for the considered examples. An ideal mechanism would work well independent of topology and the size of the overlay, and be relatively insensitive to parameters such as L and k. Moreover, it is preferable to have small values of both L and k to keep the operating overhead low.

B. Netvigator Results

For Netvigator, we use the k-mean method (described in [8]) to select 15 landmarks from 100 nodes. With only the information of the distances to the 15 landmarks, we use GNP [8] to identify the initial candidate set C. The results (reported in Table III) show the percentage increase of the stretch values over the minimum possible value using the enhanced clustering technique. It also shows the minimum number of candidates (k) required to compute the closest node. Comparing this result with Table II, it is evident that the stretch values are dramatically improved using our enhanced clustering mechanism. In all of the cases, 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. Note that all of these results were obtained using only 15 landmark nodes.

VI. SCALABILITY CONSIDERATIONS

A. Distributed Global Information Table

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. In this section we describe a distributed global information service to make Netvigator more robust and scalable. We build global information service on top of a distributed hash table (DHT) based overlay network. A DHT-based overlay provides a hash table abstraction that maps "keys" to "values" on top of an overlay network, and has the advantages of good scalability, fault-tolerance and low management cost [14]. 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. This will result in shorter response time and better network resource utilization. This section outlines our basic approach.

Netvigator requires each node on the DHT to perform RTT measurements to the landmark nodes to obtain a landmark vector. The landmark vectors without distances to the milestones are used as the DHT keys for the infrastructure nodes to join the DHT. When a service/resource node stores its information on the DHT, it computes its landmark vector in the same way as it is done for the DHT nodes and uses its landmark vector as the DHT key to store its information. Similarly, it retrieves information about nodes in network proximity using its landmark vector. This approach has two advantages: (i) information of a node is stored on a DHT node that is close to it; and (ii) information of nodes that are near each other are stored close to each other on the DHT. Therefore, to find information about nodes that are close to a particular node, we first route to the zone using the node's landmark vector as the key and then perform a localized search.

In reality, a landmark vector is usually of a high dimensionality. Requiring the DHT to also have a high dimensionality increases its management cost. We consider an extreme case and use a Chord-like DHT to host the global information table. This is an extreme case since it provides only a one dimensional abstraction. Rather than using the landmark vectors

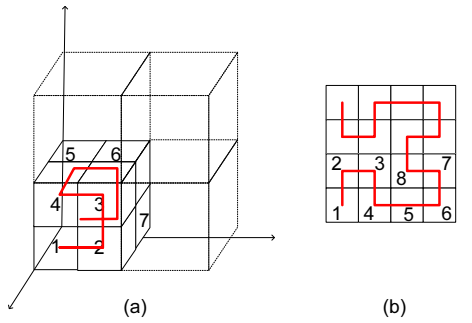


Fig. 7. (a) Fitting a Hilbert curve on a 3-dimensional space. (b) Fitting a Hilbert curve on a 2-dimensional space.

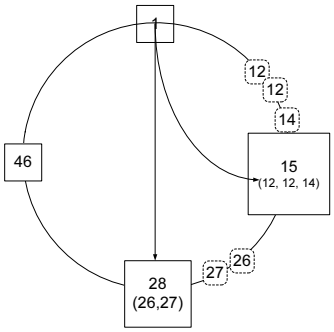


Fig. 8. Storing node information according to network proximity.

as the DHT keys directly, we perform dimensionality reduction of each landmark vector to obtain a scalar number. We call this scalar number the *landmark number* of the node, and use it as the node's DHT key. Dimensionality reduction of landmark vector to landmark number can be done in several different ways, for example, using space-filling curve or Harmonic mean of a node's distances to the landmarks. Figure 7 illustrates how to use a space-filling curve (in this case, Hilbert Curve [1]) to reduce dimensionality of the landmark vectors of a node. Space-filling curves map points in the domain $\Re^d$ into $\Re^1$ so that the closeness relationships among the points are preserved.

Figure 8 shows how landmark numbers can be used to construct topology-aware Chord and are used as keys to store information of nodes onto the DHT. There are four DHT nodes whose landmark numbers are 1, 15, 28, and 46. There are two nodes that have landmark number 12, and one node with landmark number 14. The information of these three nodes are stored on the DHT node with landmark number 15.

We have conducted preliminary simulations by randomly picking 2,048 client nodes from the topology described in Section IV-D. Even with a single Hilbert curve, and no replication, a node can find the closest node on the local node (the node where it is hashed to) with a probability of 83% when each local node stores 256 nodes.

B. 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 9 compare these two smart hop skipping techniques on PlanetLab. The baseline case of $m0\_R$ is also shown.

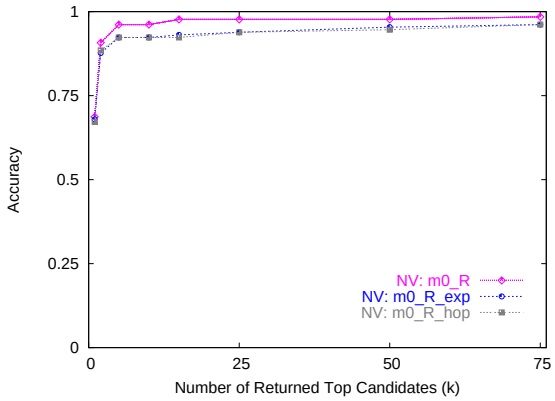


Fig. 9. Accuracy of m0_R, m0_R_exp, and m0_R_hop with L=12 on PlanetLab.

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.

VII. 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 technique uses a small number of landmarks and a large number of milestones that assist the overall process without incurring large overhead. Our clustering algorithms utilize distance information from the landmarks as well as milestones to obtain high accuracy in finding the closest node and is robust 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] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmaier, "Space Filling Curves and Their Use in Geometric Data Structures," *Theoretical Computer Science*, vol. 181, no. 1, pp. 3–15, July 1997.

[2] S. Banerjee, Z. Xu, S.-J. Lee, and C. Tang, "Service adaptive multicast for media distribution networks," in *Proceedings of the IEEE WIAPP 2003*, San Jose, CA, June 2003, pp. 50–60.

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

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

[5] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris, "Practical, distributed network coordinates," in *Proceedings of the ACM HotNets-II*, 2003.

[6] 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.

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

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

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

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

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

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

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

[14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the ACM SIGCOMM 2001*, San Diego, CA, August 2001, pp. 149–160.

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

[16] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," in *Proceedings of the IEEE ICDCS 2003*.

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