IDMaps: A Global Internet Host Distance Estimation Service

P. Francis S. Jamin C. Jin Y. Jin D. Raz Y. Shavitt L. Zhang

Abstract

There is an increasing need to quickly and efficiently learn network distances, in terms of metrics such as latency or bandwidth, between Internet hosts. For example, Internet content providers often place data and server mirrors throughout the Internet to improve access latency for clients, and it is necessary to direct clients to the nearst mirrors based on some distance metric in order to realize the benefit of mirrors. We suggest a scalable Internet-wide architecture, called IDMaps, which measures and disseminates distance information on the global Internet. Higher-level services can collect such distance information to build a virtual distance map of the Internet and estimate the distance between any pair of IP addresses. We present our solutions to the measurement server placement and distance map construction problems in IDMaps. We show that IDMaps can indeed provide useful distance estimations to applications such as nearest mirror selection.[1]

Keywords: network service, distributed algorithms, scalability, modeling.

1 Introduction

It is increasingly the case that a given service request from a client can be fulfilled by one of several Internet servers. Examples range from short-lived interactions such as a single Web page access to a Web server, to the long-term peering relationship between two news (NNTP) servers [3]. In all such interactions, all other things being equal, it is advantageous to access the "nearest" server with low latency or high bandwidth. Even when all other things are not equal, for instance, when different Web servers have different response times, it is still useful to include the distance to each candidate host as a factor in making a selection [4].

One method to obtain distance information is for the initiating host to measure it itself, using either unicast (`ping`, `traceroute` [5]) or multicast (expanding ring search [6]) tools. While these tools are easy to use, their utility is generally limited by their overhead. For instance, the latency of running a single `traceroute` can exceed the latency of a Web page access itself. More important still, a large number of hosts making independent and frequent measurements could have a severe impact on the Internet. Ideally, measurements made by one system (host or router) should be made available, with low overhead, to other hosts.

A useful general service for the Internet should enable a host to quickly and efficiently learn the distance between any two hosts. To be widely useful, such a service should provide an answer with a delay and overhead less than the gain achieved by using the service. A simple protocol for such a service, SONAR, was discussed in the IETF as early as February 1996 [7], and in April 1997 as a more general service called HOPS (Host Proximity Service) [8]. Both of these efforts proposed lightweight client-server query/reply protocols similar to a DNS query/reply. The specifications also required each server to produce an answer in a very short time—preferably, though not necessarily, by using information already stored locally. As

[1]Parts of this paper were previously published as [1] and [2].

stated, both services need some underlying measurement infrastructure to provide the distance measurements.

In this paper, we propose a global architecture for Internet host distance estimation and distribution which we call "IDMaps" (Internet Distance Map Service). We intend to have IDMaps be the underlying service that provides the distance information used by SONAR/HOPS. We discuss the basic IDMaps architecture and show, through Internet experiments and simulation, that our approach can indeed provide useful distance information.

In Section 2 we give an overview of our design goals, problem motivations, and the context of related works in which IDMaps is part of. In Section 3 we describe the basic IDMaps architecture: its components, distance measurement, and dissemination process. In Sections 3.2 and 3.3 we present several fundamental issues pertaining to IDMaps' deployment. In Section 4 we present results from our performance evaluation of the IDMaps architecture.

2 Overview of IDMaps

2.1 IDMaps Goals

A distance estimation service could be called upon to support a wide range of applications, from a client's accessing a single page from a Web server once, to Network Time Protocol (NTP) servers establishing long term peering relationships with each another. Each application that can potentially find a distance estimation service useful will have its own set of requirements. IDMaps is not designed to satisfy all conceivable requirements for distance estimation service. For instance, due to technology contraints and the need for global scalability of the service, we cannot hope for a general IDMaps service to provide near-instantaneous information about current delays and bandwidth seen between two Internet hosts, even though such information could be very useful to some applications.

Rather, we have taken the opposite tack—we determined roughly the best service we may be able to provide given technology constraints and the need for global scalability of the service, and then considered whether there are applications for which this level of service would be useful. We now turn to a discussion of the resulting goals.

Separation of Functions: We envision IDMaps as an underlying measurement infrastructure to support a distance information query/reply service such as SONAR. The full separation of IDMaps and the query/reply service is necessary because the different functionalities place different constraints on the two systems. The requirements for IDMaps are relative accuracy of distance measurements with low measurement overheads, while the requirements for the query/reply service are low query latency, high aggregate query throughput, and reasonable storage requirements.

By decoupling the different functionalities, we can streamline the design of IDMaps to perform measurements with low overheads and allow the query/reply service to make flexible uses of the measured distances. IDMaps' design must be robust to provide continued, though maybe less accurate, services even in the face of component failures. Furthermore, given the continued growth of the Internet, the system must scale effortlessly, utilizing new IDMaps servers whenever and wherever they appear. This calls for a design that allows IDMaps servers to be added or removed by any authorized party, at any time, at any location. IDMaps servers must be autonomous entities which monitor and measure their changing environments with minimal human intervention.

Distance Metrics: Our goal is to provide distance information in terms of latency (e.g., round-trip delay) and, where possible, bandwidth. Latency is the easiest sort of information to provide, and luckily the most generally useful. There are two reasons latency information is easy to provide. First, it is easy to measure. A small number of packets can produce a good coarse-grain estimate. Second, two different paths may have the same latency such that while our probe packets may not travel the same path as the path actually taken by the users' data packet, the reported latency would still be useful. Bandwidth is also clearly important for many applications, but compared to latency, bandwidth is more difficult to provide. It is more expensive to measure, and it is also more sensitive to the exact path—a single low-bandwidth link dictates the bandwidth for the whole path. It may also be that there is a rough correspondence between latency and bandwidth—that is, the lower the latency between two hosts, the higher the bandwidth is likely to be, thus allowing applications that need high-bandwidth to minimize latency as a first approximation.

Accuracy of the Distance Information: We believe highly accurate distance estimates (say, within 5% of the distance measured by the end-host itself) are impossible to achieve efficiently for a large scale Internet service. While we may be able to achieve this level of accuracy for each path measured, an estimate based on triangulation of such the measurements will see an accumulation of the error terms. Instead, our goal is to obtain accuracy within a factor of 2 with very high probability and often better than that. We expect this level of accuracy to be adequate for SONAR and HOPS servers. It would, for instance, allow an application to select the nearest Web server among ones that are reported to be 20 ms, 100 ms, and 500 ms away. The (reported) 20 ms server, which might actually be as far as 40 ms, would still be better than the (reported) 100 ms server, which might be as close as 50 ms. Being able to distinguish systems that are very close, very far, or somewhere in between is useful for a wide range of applications. For those that require more accurate measurements, they may at least use this coarse-grained information as a hint to server selection.

Timeliness of the Distance Information: We must consider two kinds of distance information—load sensitive and "raw" (distances obtained assuming no load on the network, which generally can be approximated by saving the minimum of a number of measurements). In the interest of scalability, we plan to provide the raw distance information with an update frequency on the order of days, or if necessary, hours. In other words, the distance information will not reflect transient network conditions, and will only adjust to "permanent" topology changes. Instantaneous or near-instantaneous (within 15 or 20 seconds) load information is both impossible to distribute globally and of diminishing importance to future applications: as the Internet moves to higher and higher speed, the propagation delay will become the dominant factor in distance measurements.[2]

Scope of the Distance Information: We assume that the distance information applies only to the "public" portion of the Internet—the backbone networks, BGP information, and possibly the public side of firewalls and border routers of private networks. Even if distance information of private networks were obtainable, it may be desirable not to include it for scalability reasons. This is not to suggest that distance information inside private networks is not important. We believe the architecture presented in this proposal can be replicated within a private network, but otherwise do not address distance measurement within private networks.

2.2 Alternative Architectures and Related Works

The primary motivation of IDMaps is to provide an estimate of the distance between any two valid IP addresses on the Internet. It is important to discuss this motivation because it significantly differentiates IDMaps from other services that also provide distance information, e.g., the SPAND, Remos, IPMA, NIMI, and MINC projects [9, 10, 11, 12, 13]. These projects serve various purposes, such as general Internet characterization, maintenance, fault-isolation, and end-system distance estimation; IDMaps, on the other

[2] While propagation delay is lower bounded by geographic distance, it is determined by topological distance. Given the dynamic nature of Internet topology, changes to topological distances can be scalably tracked only by an automatic system such as IDMaps.

hand, aspires to be a public infrastructure that provides distance information between any two arbitrary points on the Internet, where the querying entity is not one of the two end points.

An alternative to providing distance information between arbitrary pairs of hosts on the Internet would be a localized service that provides only distance information between hosts close to a distance server and remote hosts on the Internet. Such a service is simpler to provide because the amount of information each distance server has to work with scales at the order of number of possible destinations (N), instead of the N^2 order of IDMaps'—all possible pairs of Internet destinations. SONAR and SPAND propose this type of service. Localized distance service, however, works well when only a few networks conduct distance measurements. With the increase in applications that require distance information, such as peer-to-peer file sharing and client-hosted multiplayer gaming, the aggregated load of localized distance service will scale on the order of N^2. The amount of measurement traffic under IDMaps will likely be much smaller than the N^2 order because of the global sharing of distance information and as a result of our application of graph compression techniques such as t-spanners (see Sections 3.3.1 and 4.4). The administrative cost of setting up and maintaining IDMaps service is also fixed.

It has been argued in [9] that passive measurement has an advantage over active measurement in not sending additional traffic and perturbing actual Internet traffic. Although the non-intrusive nature of passive measurement is very appealing, passive measurement has its limitations:

1. Passive monitoring can only measure the regions of the Internet that application traffic has traversed. A service that relies solely on passive monitoring may not be able to discover the distance to unknown networks/hosts. For example, a client trying to choose the nearest among multiple copies (or mirrors) of a Web server will require distance information between the client and all such mirrors. A passive monitoring system can only provide distance information to mirrors that have been previously accessed, either by the client or others within its proximity.

2. Passive monitoring has only one view of the Internet because only end-to-end path data is collected. When Internet topology changes, for example, due to an upgrade of some provider's backbone network, passive monitoring may be forced to re-collect most if not all of its distance information. Although IDMaps also uses end-to-end measurements, the end-to-end distances in IDMaps are collected from multiple, intermediate, points on the Internet, this allows the distance database to locate any topological change and update only those actually effected.

3. The responsibility of deploying passive monitoring based distance service rests on the administrator of each individual network and requires certain expertise and resources. With IDMaps, network administrators only need to install a querying system, which can be standardized similar to DNS (Domain Name System).

4. Finally, passive measurement typically requires some kind of monitoring or snooping of network traffic, which may raise privacy and security concerns.

Another alternative to providing distance information on the Internet is by charting the physical connectivities between nodes (hosts and routers) on the Internet and computing a spanning tree on the resulting connectivity map. Distances between pairs of nodes can then be estimated by their distances on the spanning tree. We call this alternative the hop-by-hop approach. The projects described in [14, 15, 16, 17] provide snapshots of the Internet topology at the hop-by-hop level. This approach largely relies on sending ICMP (Internet Control Message Protocol) packets to chart the Internet. To minimize perturbation to the network, each snapshot of the topology is usually taken over a period of weeks, hence the result does not adapt well to topological changes. More seriously however, due to the recent increase in security awareness on the Internet, such measurement probes are often mistaken for intrusion attempts.

To motivate the service provided by IDMaps, we present some results on how it can be used in the nearest mirror selection problem. The problem of mirror selection and mirror ranking has been studied extensively in the literature, for example by the authors of [18, 19, 20, 21, 22, 23, 24, 25, 26, 9]. Previous works on mirror selection can be classified into three categories: network-layer, application-layer, and measurement-based. In network-layer selection, the nearest mirror in terms of number of hops (AS hops or router hops) [27] or in terms of network latency [20] is selected. In application-layer selection, some application performance metrics (e.g., server load and available bandwidth) were taken into account when selecting a mirror [18, 9]. In measurement-based selection, measurements (e.g., actual download of data objects from different mirrors) are conducted to rank different mirrors [23]. In [23], the authors measured nine clients scattered throughout the United States retrieving documents from forty-seven Web servers, which mirrored three different Web sites. They presented findings that revealed good stability of mirror rankings according to download time. In [28], the authors presented a selection technique that involves periodic measurements from clients to all of the mirrors. Finally, as already mentioned, the authors of [9] proposed a selection scheme based on shared passive end-to-end performance measurements performed by clients in the same network and collected at a central repository accesible to all local clients. The effectiveness of client-side mirror selection and provider-side mirror selection were studied in [21] and [29, 26] respectively. The authors of [18, 21] studied different distance metrics and concluded that measured round-trip-time is the best metric to base selection decisions upon, for different data object sizes. We emphasize that the nearest mirror selection problem is used only as a motivational example in this paper. We do not intend to study nearest mirror selection problem exhaustively in this paper.

3 IDMaps Architecture

This section outlines the IDMaps architecture. Specifically, we address the following three questions:

1. What form does the distance information take?
2. What are IDMaps' components?
3. How should the distance information be disseminated?

3.1 Various Forms of Distance Information

The conceptually simplest and most accurate form of distance information IDMaps can measure consists of distances between any pair of globally reachable IP addresses[3] (as shown in Figure 1). The distance from one IP address to another is then determined by simply indexing the list to the appropriate entry (using a hashing algorithm) and reading the number. The large scale of this information (on the order of (H^2), where H, number of hosts, could be hundreds of millions) makes this simple form of distance infeasible, as does the task of finding all such hosts in an ever-changing Internet in the first place.

The next simplest would be to measure the distances from every globally reachable Address Prefix (AP) on the Internet to every other (Figure 1). An AP is a consecutive address range of IP addresses within which all hosts with assigned addresses are equidistant (with some tolerance) to the rest of the Internet. Determining the distance from one IP address to another is only slightly more complicated than the first approach—each IP address is first mapped into its AP, and the AP is then indexed in the list. Unlike determining the global set of IP addresses, determining the set of APs, while non-trivial, seems feasible (see Section 3.4). The scale of the information, however, is still prohibitive. The number of assigned CIDR blocks [30] is around 100,000 as of March 2001 and growing; there are probably several times as many distinct APs as there are CIDR blocks. Probing, disseminating, and storing the full list of (P^2) pairs of AP-AP distances (easily a terabyte, given 200,000 APs, assuming on average 2 AP per CIDR block, and 25 bytes per list entry) is probably out of the question.

[3]Understanding here that different IP addresses may be reachable at different times, given technologies like NAT and dial-up Internet access.

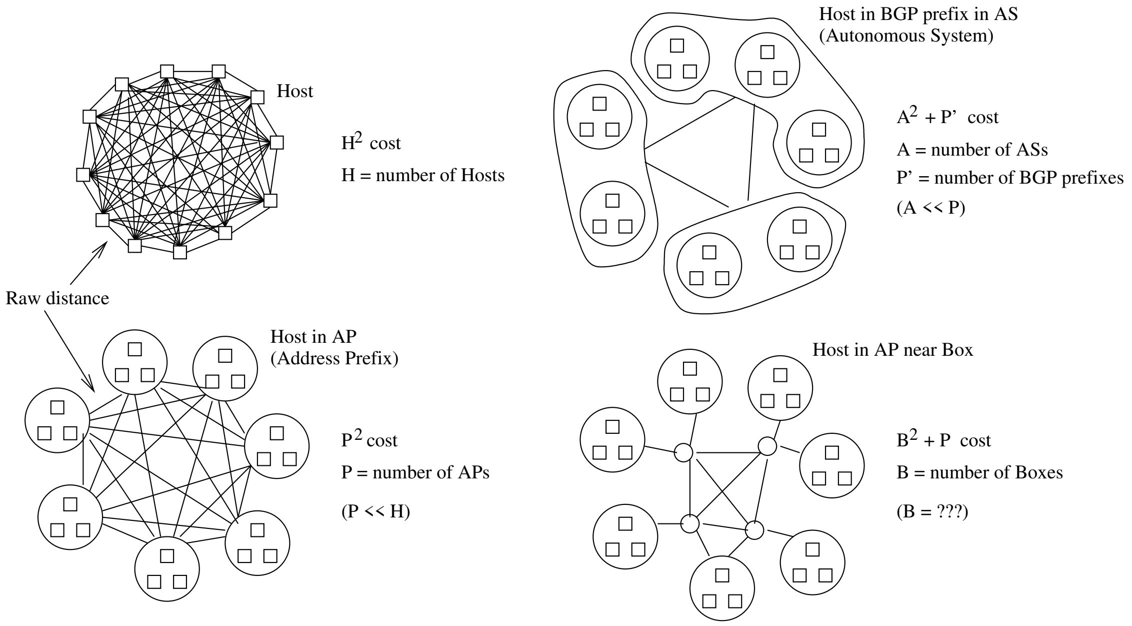


Figure 1: Various Forms of Distance Information

Clearly some way of further compressing this information is needed. One way is to keep a list of distances from every Autonomous System (AS) to every other. The AS is the unit of path information carried by the BGP inter-domain routing protocol. BGP also maps blocks of IP addresses into their ASs [31]. This shrinks the size of the information to $A^2 + P'$, where A ($A \ll P$) is the number of ASs and P' the number of BGP-advertised IP address blocks (not an AP by the above definition, but of the same order of magnitude in size). While still a large list, maintaining it is certainly feasible (there are about 10,000 ASs as of March 2001). The resulting accuracy of the estimated distances, however, is highly suspect. Many ASs are global in scope, and multiple ASs can cover the same geographic area. It is often the case that some IP hosts are very close to each other (both in geographical and latency terms) yet belong to different ASs, while other IP hosts that are very far apart belong to the same AS.[4]

Yet another form of distance information includes some clustering of APs, but at a smaller unit than the AS. We can select certain systems, which we will call Tracers, to be distributed around the Internet, so that every AP is relatively close to one or more Tracers. The distances between these Tracers are then measured, and so are the distances between APs and their nearest Tracer(s). The distance between any two APs can then be calculated as the sum of the distance from each AP to its nearest Tracer, and the distance between the two Tracers. The resulting accuracy depends on the number of Tracers and where they are located. Assuming that we can manipulate the number and location of Tracers, we have a tuning knob for increasing accuracy at the expense of measuring more raw distances.

This approach scales as $B^2 + P$, where B is the number of Tracers. Assuming that P, the number of APs, is a manageable number (no more than several hundred thousand), the question then becomes, how big should we make B? If B is on the order of 10,000, then the size of the list is quite large. If, however, B is on the order of 500, then the B^2 component is roughly the same as the P component and, at least in terms of simple storage and lookup, definitely manageable.

[4]If the internal topology of each AS is known, e.g. discovered using the *Mercator* tool [32], a more accurate distance can be computed as the shortest path across the internal topologies of transit ASs between two hosts. This approach, which we call the Hop-by-Hop model, has limitations and drawbacks of its own, as we documented in [1].

Of the four forms of distance information mentioned above, the last one appears to have the best scalability with reasonable estimation accuracy. We decided to use this form of distance information in IDMaps. There are thus three main components of IDMaps: APs, Tracers, and the raw distances, which we call "Virtual Links" (VLs). We further differentiate VLs into two types: those between Tracers[5] (Tracer-Tracer VLs) and those between Tracers and APs (Tracer-AP VLs). Before we examine each component in greater detail, we first evaluate the basic assumption that we can estimate the distance between two points as the sum of distances between intermediate points. In analytic terms, this assumption relates to whether the triangle inequality holds.

Triangulation on the Internet. Given a graph G with the set of vertices V, a cost function C is said to satisfy the triangle inequality if for all vertices a, b, c in a graph, $C(a,c) \leq C(a,b) + C(b,c)$ [33] (in the remainder of the paper, we use the notation (a,b) interchangeably with $C(a,b)$). If distances (a,b) and (b,c) are known, then from the triangle inequality we have that (a,c) is bounded above by $(a,b)+(b,c)$, and below by $|(a,b) - (b,c)|$. If either of the two distances is small relative to the other, the bound is tight and the estimate accurate. Deriving a distance estimate from this bound has been referred to as "triangulation" [34, 35].

A key point to keep in mind is that any time we estimate a distance from a to c based on distances to an intermediary b, we are to some degree relying on what we will term efficient routing: that Internet routing does indeed strive to find low-latency paths, and that the routes used by two nearby hosts will not be drastically different from each other. This assumption can be violated due to policy-based routing, and also by the use of large layer-2 "clouds" by ISPs that are invisible at the network layer, and hence contain significantly complex topology that are completely hidden from network-layer-only viewpoints such as available to IDMaps. If efficient routing is violated, it can render the triangle inequality incorrect: (a,c) might be much higher than $(a,b) + (b,c)$ or much lower than $|(a,b) - (b,c)|$. of Internet path data to assess how often an alternate route In this section, we present results from rudimentary experiments done on the Internet to explore the feasibility of using triangulation to estimate distances. Our intention here is not to test whether the triangle inequality holds over all parts of the Internet (indeed it does not [36]), but only whether using triangulation to estimate distances on the Internet, independent of IDMaps, is at least feasible.

We analyze end-to-end `traceroute` measurements collected using the network probe daemon (NPD) tool described in [37]. A number of sites on the Internet were recruited to run NPDs. At random intervals, each NPD measured the route to another NPD site using `traceroute`. In this paper we analyze the `traceroute` data collected by the NPDs in two experiments: the Nov. 3 to Dec. 21, 1995 experiment ($D1$) and the Sep. 4, 1996 to Jan. 24, 1997 experiment ($D2$). We split experiment $D1$ into 6 data sets, and experiment $D2$ into 3 data sets. The data sets are non-overlapping in time. Thirty-three hosts distributed around the globe participated in experiment $D1$, 48 in experiment $D2$. A description of the measurement process, such as inter-measurement interval, number of measurements per period, etc., and data cleansing done on the collected data are available in [37] for the $D1$ data set and in [38] for the $D2$ data set.

In addition, we recently collected two new data sets, $D3.1$ and $D3.2$, using the Multiple Traceroute Gateways [39] during Jan 2000. The Multiple Traceroute Gateways (MTG) are a collection of voluntary Web sites, each of which can run `traceroute` from itself to a specified remote address. Data set $D3.1$ contains `traceroutes` from 33 Traceroute Gateways measuring distances to all the others continuously in a round-robin fashion over a period of five days. Each round took 20 to 40 minutes. Data set $D3.2$ contains the `traceroutes` from 74 Traceroute Gateways measuring distances to all the others exactly once.

[5]The actual distances used would not include the legs from the Tracers to their backbone routers, since this part of the path is not used by other hosts. For the sake of readability, however, we refer to the distance between one Tracer's router and another Tracer's router simply as the distance between the two Tracers.

Data Set	Hosts	Total Paths	Shortest Paths
$D1.1$	26	12,192	590
$D1.2$	31	17,448	804
$D1.3$	30	15,050	733
$D1.4$	33	16,878	830
$D1.5$	33	27,366	980
$D1.6$	32	26,936	960
$D2.1$	45	128,244	2,663
$D2.2$	48	138,468	2,790
$D2.3$	39	95,925	2,229
$D3.1$	33	31,955	1,047
$D3.2$	73	341,674	5,138

Table 1: Number of triangles obtained from each data set.

For each data set, we estimate the latency between every `traceroute` source, s, and every `traceroute` destination, d, as the minimum of the end-to-end round-trip-times reported across all of the `traceroutes` from s to d. From each data set we then compute a set of triangles, each involving three such minimum latency `traceroutes`: from a host a to another host b, from host b to a third host c, and from host a to host c. Column 2 of Table 1 lists the number of triangles we obtained from each of the 11 data sets. Column 3 lists the number of shortest-path triangles computed from each data set. Given all the b's that can potentially be used to estimate the distance (a, c) as $(a, b) + (b, c)$, we call the path that uses the b that provides the smallest $(a, b) + (b, c)$ the shortest-path triangle.

We use only the additive form of triangulation (estimating (a, c) using $(a, b) + (b, c)$) in this paper because concatenation of distances involving multiple intermediary points is much simpler for the additive form and, as we show below, the resulting estimates are acceptable for our purposes. We emphasize that computing distances on the Internet is not a straight forward process and there is future work needed on distance measurement and estimation; however we also caution against interpreting the triangulation results presented here as an indication of how well IDMaps will perform. We expect IDMaps to perform better than the results presented here due to the more deliberate placement of Tracers under IDMaps. For example, we expect the additive form of triangulation to hold more prevalently when Tracers are placed strategically and addresses are aggregated into APs (see Sections 3.2 and 4.4).

Figure 2 shows the ratios of $\frac{(a,c)}{(a,b)+(b,c)}$ for all shortest-path triangles in our data sets. The closer this ratio is to 1, the smaller the triangulation error. Without differentiating which curves belong to which data set, we observe that that between 75% and 90% of triangulation estimates fall within a factor of 2 of the real distances. We reported similar results involving only the $D1$ and $D2$ data sets in an earlier version of this work ([1], though an analysis error in that paper misreports the figures), and this was also shown by [36]. Studying the extreme cases at both ends of the distributions, we found that (a, c) being orders of magnitude smaller than $(a, b) + (b, c)$ is mostly caused by a and c being co-located. On the other extreme, (a, c) being much larger than $(a, b) + (b, c)$ is mostly caused by large (a, c). We were not able to track down why these paths have very long distances in general (see, however, [36]).

For comparative purposes, we also show in Figure 3 the cumulative distribution function of the triangulation error from triangles involving all potential b's. The figure shows, for example, that the actual distance between a and c in about 40% of the triangles formed is shorter than half the sum of (a, b) and (b, c), which gives good indication that triangulating shortest paths will more closely approximate the actual distances. Note that for (a, c) to be 100 times longer than $(a, b) + (b, c)$ is as bad as for it to be 100 times shorter.

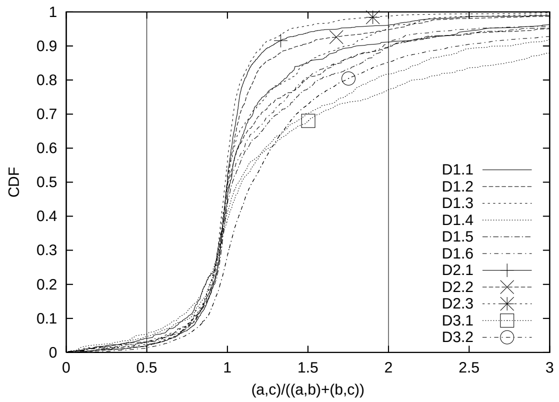


Figure 2: Cumulative Distribution Function (CDF) of the ratio of $(a,c)/((a,b)+(b,c))$ for shortest-path triangles.

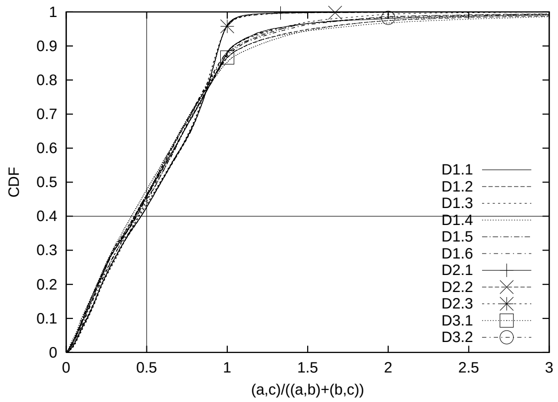


Figure 3: Cumulative Distribution Function (CDF) of the ratio of $(a,c)/((a,b)+(b,c))$ for all triangles.

In summary, while we cannot as yet make any claim as to the potential accuracy of triangulation in IDMaps, results presented in this section suffice to argue that the use of the additive form of triangulation as a method to estimate distances on the Internet is feasible.

3.2 Tracer Placement

As mentioned above, the resulting accuracy of IDMaps distances depends on the number of Tracers and where they are located. Ideally, Tracers should be placed where they are able to obtain accurate raw distance information. In this section, we briefly review two graph theoretic approaches we apply to determine the number and placement of Tracers, namely the k-HST and the minimum K-center algorithms. These algorithms have been used to determine placement of fire stations, ambulance placement, etc. [40]. More formal descriptions of these algorithms are available in [41, 42]. The assumption we make in applying these algorithms to the Tracer placement problem is that the most accurate distance information can be obtained by minimizing the maximum distance between an AP and its nearest Tracer. Given a graph, these algorithms partition it into subgraphs satisfying certain conditions. Since IDMaps cannot assume prior knowledge of Internet topology, these algorithms are mostly useful in informing and evaluating our placement heuristics. We describe three placement heuristics in subsection 3.2.3.

We use the generic term "center" in place of "Tracer" in the following descriptions. We present two variants of the center placement problem: in the first case, the maximal center-node distance is given, and one is required to find the minimal number of centers needed to satisfy this constrain; in the second case, the number of centers is given, and one needs to decide the locations of these centers such that the maximum distance between a node and the nearest center is minimized. Each of the two algorithms described below can be used to solve both of these problems.

Number of Centers. Given a network G with N nodes (that is, the topology is a priori known), and a bound $\mathcal{D}$, one has to find a smallest set of centers S_C such that the distance between any node i and its nearest center $C_i \in S_C$ is bounded by $\mathcal{D}$. The performance metric (P_{diam}) is the size of this set ($|S_C|$). More formally, find the minimal K such that there is a set $S_C \subset V$ with $|S_C| = K$ and $\forall v \in V : d(v, C_v) \leq \mathcal{D}$, where C_v is the nearest center to v.

Center Placement. For the placement of a given number of centers, one could consider the following metric (P_{minK}): given a network G with N nodes, and a number K, find a set of centers S_C of size K that minimizes the maximum distance between a node and the nearest center. This problem is known as the minimum K-center problem.

While our Tracer placement problem is similar in spirit to the center placement problem articulated above, for Tracer placement we have to consider other practical deployment issues, primarily that we do not know the Internet topology a priori, and that the Internet topology changes dynamically. Furthermore, we must consider the willingness of network owners to host Tracers, and the managerial and financial constraints on the number of Tracers we can afford to deploy and maintain. Hence our goal is not to determine the minimum number of Tracers required to provide distance estimates at a given precision, but rather to evaluate the effectiveness of various Tracer placement and number of Tracers. In Section 4.4.4 we present results from experiments with different number of Tracers. We use the graph theoretic results only as yard sticks to evaluate the performance of our placement heuristics presented in Section 3.2.3, we do not intend to use these graph theoretic algorithms in actual deployment of Tracers for reasons cited above.

3.2.1 k-HST

We present in this subsection a placement algorithm based on k-hierarchically well-separated trees (k-HST) [41]. Intuitively, think of the algorithm that generates a k-HST as a top-down graph partitioning algorithm that transforms a graph into a tree of partitions by recursively dividing far-apart nodes in each partition into several smaller child partitions. The diameter of a partition is defined to be the furthest distance between two nodes in the partition. More formally, the k-HST algorithm consists of two phases. In the first phase, the graph is recursively partitioned as follows: A node is arbitrarily selected from the current (parent) partition, and all the nodes that are within a random radius from this node form a new (child) partition. The value of the radius of the child partition is a factor of k smaller than the diameter of the parent partition. This process recurses for each partition, until each node is in a partition of its own. We then obtain a tree of partitions with the root node being the entire network and leaf nodes being individual nodes in the network. In the second phase, a virtual node is assigned to each of the partitions on each level. Each virtual node in a parent partition becomes the parent of the virtual nodes of the child partitions. The length of the links from a virtual node to its children is half the partition diameter. We embed the virtual nodes in the original graph based on a technique developed by Awerbuch and Shavitt [43]. Together, the virtual nodes also form a tree.

The randomization of a partition radius is done so that the probability of a short link being cut by partitioning decreases exponentially as one climbs up the tree. Hence nodes close together are more likely to be partitioned lower down the tree. We take advantage of this characteristic of the resulting k-HST tree to devise the following greedy algorithm to find the number of centers needed when the maximum center-node distance is bounded by $\mathcal{D}$. Let node r be the root of the partition tree, $\mathcal{N}_i$ be the children of node i on the partition tree, and $\mathcal{L}$ be a list of partitions sorted in the decreasing order of the partition diameter at all times. $max(\mathcal{L})$ denotes the partition at the head of the list, and $diam(max(\mathcal{L}))$ its diameter. Fig. 4 presents our greedy algorithm on the k-HST tree (see [43] for a more formal presentation of the algorithm). The algorithm pushes the centers down the tree until it discovers a partition with diameter $\leq \mathcal{D}$. The number of partitions, $|\mathcal{L}|$, is the minimum number of centers required to satisfy the performance metric P_{diam}. To select the actual centers, we can simply set the virtual nodes of these partitions in $\mathcal{L}$ to be the centers.

The k-HST-based greedy placement algorithm presented above tells us the number of centers needed to satisfy the performance metric P_{diam}. For any given budget of centers, the algorithm above can also be used to determine their placement. For example, to place K centers, we simply change line 3 in Fig. 4 with "while ($|\mathcal{L}| < K$)". Obviously, the performance metric P_{diam} may no longer be satisfied for K below a certain number.

Algorithm 1 (Greedy placement)

1. $\mathcal{L} \leftarrow \mathcal{N}_r$
2. $h \leftarrow max(\mathcal{L})$
3. while $(diam(h) > \mathcal{D})$
4. $\quad \mathcal{L} \leftarrow \mathcal{L} - h$
5. $\quad \mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{N}_h$
6. $\quad h \leftarrow max(\mathcal{L})$

Figure 4: Greedy placement of centers on an k-HST tree.

Algorithm 2 (2-approximate minimum K-center [42])

1. Construct $G_1^2, G_2^2, \ldots, G_m^2$
2. Compute M_i for each G_i^2
3. Find smallest i such that $|M_i| \leq K$, say j
4. M_j is the set of K centers

Figure 5: Two-approximate algorithm for the minimum K-center problem.

3.2.2 Minimum K-Center

The placement of a given number of centers such that the maximum distance from a node to the nearest center is minimized, known as the minimum K-center problem, is NP-complete [44]. However, if we are willing to tolerate inaccuracies within a factor of 2 (2-approximate), i.e. the maximum distance between a node and the nearest center being no worse than twice the maximum in the optimal case, the problem is solvable in O$(N|E|)$ [42]. In contrast to the k-HST algorithm, one can intuitively think of the minimum K-center algorithm as a bottom-up approach to graph partitioning: it collects nearby nodes into clusters.

More formally, the minimum K-center algorithm receives as input a graph $G = (V, E)$ where V is the set of nodes, $E = V \times V$, and the cost of an edge $e = (v_1, v_2) \in E$, $c(e)$, is the cost of the shortest path between v_1 and v_2. All the graph edges are arranged in non-decreasing order by cost, c: $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$, let $G_i = (V, E_i)$, where $E_i = \{e_1, e_2, \ldots, e_i\}$. A square graph of G_i, G_i^2 is the graph containing V and edges (u, v) wherever there is a path between u and v in G_i of at most two hops, $u \neq v$. An independent set of a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that, for all $u, v \in V'$, the edge (u, v) is not in E. An independent set of G_i^2 is thus a set of nodes in G_i that are at least three hops apart in G_i. We also define a maximal independent set M as an independent set V' such that all nodes in $V - V'$ are at most one hop away from nodes in V'.

The outline of the minimum K-center algorithm from [42] is shown in Fig. 5. The basic observation is that the cost of the optimal solution to the K-center problem is the cost of e_i, where i is the smallest index such that G_i has a dominating set[6] of size at most K. This is true since the set of center nodes is a dominating set, and if G_i has a dominating set of size K, then choosing this set to be the centers guarantees that the distance from a node to the nearest center is bounded by e_i. The second observation is that a star topology in G_i, transfers into a clique (full-mesh) in G_i^2. Thus, a maximal independent set of size K in G_i^2

[6] A dominating set is a set of D nodes such that every $v \in V$ is either in D or has a neighbor in D.

implies that there exists a set of K stars in G, such that the cost of each edge in it is bounded by $2e_i$: the smaller the i, the larger the K. The solution to the minimum K-center problem is the G_i^2 with K stars. Note that this approximation does not always yield a unique solution.

The 2-approximate minimum K-center algorithm can also be used to determine the number of centers needed to satisfy the performance metric P_{diam} by picking an index k such that $c(e_k) \leq \mathcal{D}/2$. The maximum distance between a node and the nearest center in G_k is then at most $\mathcal{D}$, and the number of centers needed is $|M_k|$.

3.2.3 Tracer Heuristics

The graph theoretic approaches described above assume known network topologies. However, the topology of the Internet may not be known to all parties at any one time. Furthermore, the Internet topology changes continuously, from physical and algorithmic causes. In this paper, results from the graph theoretic algorithms are used as yard sticks to evaluate the performance of our Tracer placement heuristics.

Given a number of Tracers and an unknown topology, we devise the following heuristics for Tracer placement:

Stub-AS: Tracers are placed only on stub Autonomous Systems (ASs). This would most likely reflect the initial deployment of Tracers on the Internet, when Tracers would be run from end hosts.

Transit-AS: Tracers are placed only on transit ASs, i.e. ASs that are connected to several neighboring ASs and are willing to carry traffic from one of its neighbors to another. This reflects deployment of IDMaps on ISP backbones. As IDMaps becomes more popular, we hope that there will be enough incentives for network providers and institutions with private networks to deploy IDMaps. For networks that do not have IDMaps deployed, Tracers could still be run from end hosts.

Mixed: Tracers are randomly, with uniform distribution, placed on the network. This is the simplest placement method and does not assume any knowledge of network characteristics. It means Tracers are placed on both stub and transit ASs (hence the name "Mixed"). In terms of deployment, this placement reflects IDMaps being partially deployed on some ISPs.

The choice of these very simple placement heuristics reflects our intention to delimit how well IDMaps can be expected to perform. While the graph-theoretic approaches described in the previous section assume full knowledge of the underlying network topology, our placement heuristics intentionally assume minimal knowledge of the underlying network topology. Knowing the performance of these boundary cases, we can evaluate the benefits of further refinements to the algorithm, for example by iteratively using the distance map collected to compute better placement. As we show in Section 4.4.1 even these most rudimentary placement heuristics can give very good results.

Aside from the placement of the Tracers, we also need to address the question of how many Tracers are needed on the Internet. We could apply the P_{diam} performance metric and require that each AP is within a certain distance of the nearest Tracer, or require that the distances between APs and Tracers be minimized (P_{minK} performance metric). Unfortunately, these performance metrics may be arbitrarily difficult to achieve scalably. In the worst case, in a network where the minimum distance between any two nodes is no less than the performance metric, we would have to put a Tracer at every node to satisfy the performance metric. In addition, the topology of the network also affects the number of required Tracers. For example, the number of Tracers needed on a tree topology will be different from that on a full-mesh.

3.3 Virtual Links

Once Tracers are placed on the Internet, they start tracing each other and APs (defined in Section 3.1). The resulting distance information are advertised to IDMaps' clients. Clients of IDMaps, such as SONAR or

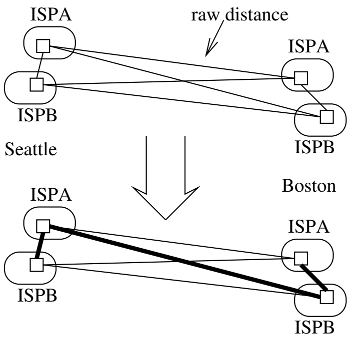


Figure 6: Distance Measurement Reduction

Algorithm 3 (t-spanner [45])

1. sort E by cost c in non-decreasing order
2. $G' \leftarrow (V, E'), E' \leftarrow \emptyset$
3. for each edge (u, v) in E do
4.   if $(t * c((u, v)) < d_{G'}(u, v))$
5. $E' \leftarrow (u, v) \cup E'$

Figure 7: The t-spanner algorithm.

HOPS servers, collect the advertised distance information and construct distance maps. In this section, we first discuss Tracer-to-Tracer part of the distance map; then we discuss Tracer-to-AP virtual links.

3.3.1 Tracer-Tracer Virtual Links

As of March 2001, there are close to 100,000 routing address prefixes in the Internet [30]. Assuming we have 5% as many Tracers (see section 4.4.4 for the effect of having more or less Tracers) and each Tracer traces to every other Tracer, there will be millions of VLs to be continually traced and advertised. Where efficient routing and triangle inequality hold (see Section 3.1), it is not necessary to list all B^2 Tracer-Tracer distances to achieve good accuracy. For example, given a number of Tracers in the Seattle and Boston areas, it would almost certainly not be very useful to know all of the distances between them.[7] Knowing the distance of one Tracer from each area would likely allow a sufficient distance approximation between hosts in Seattle and hosts in Boston (Figure 6).

We now generalize the above observations by applying the t-spanner algorithm [45] to distance map construction. A t-spanner of a graph is a subgraph where the distance between any pair of nodes is at most t times larger than the distance in the original graph [46, 47]. Formally, given a graph, $G(V, E)$, a t-spanner is a subgraph $G'(V, E')$, $E' \subseteq E$ such that $d_{G'}(u, v) \leq t \cdot d_G(u, v)$, $\forall u, v \in V$. The number of edges required to build a 5-spanner, for example, on a graph with N nodes is bounded by O($N^{3/2}$). For $t = \log N$, the bound on the number of edges required is O(N). We examine the effect of using different t values on the performance metric P_{app} in Section 4.4.

[7] We recognize that geographical distance does not directly relate to network distance (though often the two are related), for instance because of multi-point traffic exchange between global ISPs. We use geographical locations here to simplify the discussion.

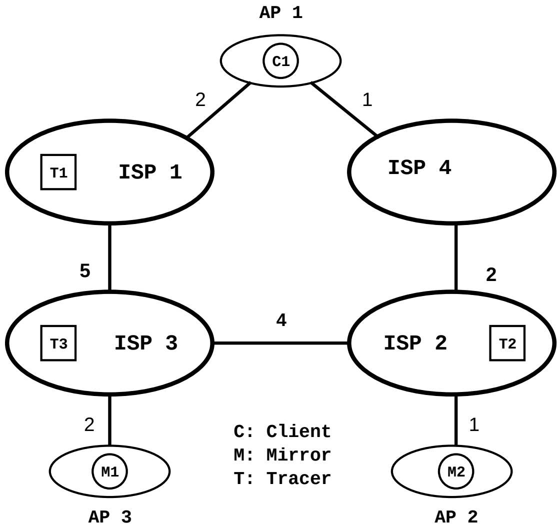


Figure 8: Network with multiple connections to the Internet.

Cai [48] showed that the minimum t-spanner (a t-spanner with the minimum number of edges) is an NP-complete problem. However, asymptotically, the algorithm of Althöfer et al. generates, from a graph $G(V, E)$, a t-spanners whose edge count is in the same order of magnitude as the optimal t-spanner [45]. Fig. 7 presents the t-spanner algorithm of Althöfer et al. [45]. It first sorts, in increasing order, all the edges in G by the edge cost. The edges are examined starting with the shortest. An edge (u, v) is added to the spanner G' if it improves the distance between u and v by at least a factor of t.

To apply the t-spanner algorithm described above would require IDMaps clients to first collect and store all B^2 VLs advertised by the B Tracers. It also assumes that once a t-spanner is computed, it will remain static. In reality, we do not expect all IDMaps clients to be able to store B^2 VLs. As the underlying Internet topology changes, we expect the set of VLs that makes up the t-spanner to change from time to time. To keep track of topological changes, Tracers continually trace and advertise all B^2 VLs—albeit at different frequencies, with higher frequencies for those used by the t-spanner and those that are less stable; accordingly, IDMaps clients must continually examine each new advertisement of a VL and continually update their t-spanners.

3.3.2 Tracer-AP Virtual Links

Recall that an AP (Address Prefix) as a consecutive address range within which all assigned addresses are equidistant (with some hysteresis) to the rest of the network. Unless an AP is pre-configured into a dedicated Tracer (see Section 3.4), only the Tracers nearest to the AP itself can discover and subsequently advertise the Tracer-AP distance. As a result, when a Tracer first discovers an AP, it assumes itself to be the nearest Tracer and advertises its distance to the AP as the Tracer-AP distance. Thereafter, however, other Tracers should probe the AP to determine if they may be closer. If one is, then it advertises its closer distance. Upon hearing this, the Tracer with the longer distance can stop advertising.

In this paper, we study whether it is sufficient for each AP to be traced by a single Tracer. If an AP has more than one path to the rest of the Internet, having a single Tracer tracing to that AP could result in inaccurate distance estimates between this AP and hosts that are not sharing paths with the Tracer. Fig. 8 shows a network of four ISPs and three APs. One Tracer each is placed in ISP1, ISP2, and ISP3, i.e., T1, T2, and T3, respectively. The label on each link denotes the distance of the link. Consider the following scenario. Mirrors M1 and M2 of a service are placed in AP3 and AP2, respectively. Assume that Tracer T1 traces to AP1, T2 traces to AP2, and T3 traces to AP3. Client C1 in AP1 will then be directed to mirror M1 in AP3 instead of M2 in AP2. Had Tracer T2 also traced to AP1, the client would have been directed to M2. We investigate the effect of having more than one Tracer tracing to each AP in Section 4.4.6.

3.4 Discovering APs

APs in IDMaps are the end-points of distance information. The difficulty in grouping Internet hosts into APs is that the address blocks advertised by ISPs in BGP do not necessarily represent a group of addresses in a single Internet "location." Inside an ISP, a BGP-advertised block may be further partitioned into many sub-blocks (i.e., APs) that are topologically far away from each other. The only direct way the address ranges of these sub-blocks can be learned is by querying the ISPs' routers using SNMP, or by listening to their routing protocols. Where ISPs themselves have setup Tracers, these methods can be used. Ideally, a large number of Tracers should be installed by each ISP to provide accurate distance information for each site. These "dedicated" Tracers can easily be configured with site AP information.

For APs not covered by either of the above, "general purpose" Tracers will have to discover the address boundaries of APs. One basic approach is to let a few of these Tracers obtain address blocks from BGP and advertise them to all other Tracers. These address blocks serve to "seed" the Tracers. Tracers will periodically select an address block, and probe a few specific IP addresses scattered throughout the block, including addresses at the high and low boundaries of the block. If the probe reveals that all these addresses reside in the same location, then the block can be considered to be a single AP. Here location is determined using the latency between a Tracer and hosts in the block. If probes to two IP addresses produce "similar" latencies, say within 5% of each other, they are assumed to be in the same AP. If the addresses are from different locations, then a kind of binary search over the block can be executed in search of the boundaries of the sub-blocks. In such a search, each new probe would select an address midway between two previously probed addresses that were found to be in different APs.

3.5 Distance Information Dissemination

In this section we explore how distance information produced by IDMaps can be collected by higher-level services in the context of a complete distance map service. Figure 9 illustrates a three-tier model of a distance map service. At the bottom are the Tracers (T) that measure and advertise raw Internet distances (VLs). In the middle layer, we have IDMaps Clients (iC), or simply Clients that collect the raw distances and build a virtual distance map of the Internet. SONAR and HOPS servers are examples of potential IDMaps Clients. These Clients use the distance maps computed to answer queries from their clients (C), which are user applications such as a web browser or a napster client. IDMaps itself is concerned only with the infrastructure at the bottom tier that collects and advertises raw distances.

IDMaps Tracers continuously send probe packets to "explore" the Internet to measure distances. Measured distances are then advertised to Clients. Upon receiving such advertisements, each Client independently determines the usefulness of the advertised information and handle it appropriately. A Client running t-spanner, for example, would consider the virtual links that form the t-spanner more useful than the others. The design of IDMaps allows its Clients to provide feedback to Tracers on which virtual links form their t-spanners (see Section 3.3.1) and should be continually traced. To capture topological changes, instead of completely disregarding virtual links not currently used in building distance maps, Tracers will simply reduce the frequencies at which they trace and advertise these links.

When an Internet host is interested in learning the distance between two hosts, it queries a Client. The Client then runs a shortest path algorithm to determine the end-to-end distance of the two hosts in its distance map. The result of the computation is sent back as a reply to the host.

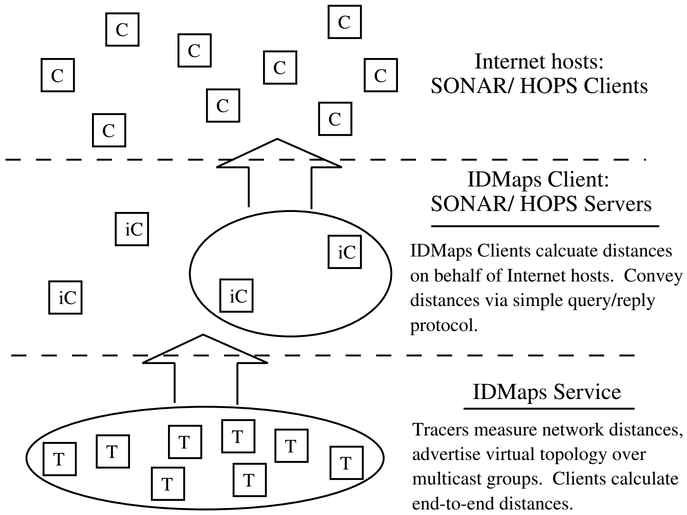


Figure 9: Basic Model: Two Tiers of Functionality

One ideal mechanism to disseminate distance information is IP multicast. Each Tracer periodically (with random skew in the period, to prevent inadvertent synchronizations among Tracers) transmits its raw distances onto the appropriate multicast group(s). A Client receiving on the group(s) will, over time, receive all of the raw distances. This mechanism allows each Tracer to transmit its portion of the distance map without coordination with other Tracers. The Tracers never need to know who are the Clients, and the Clients do not need to know about the Tracers in advance (though they do learn of the Tracers through reception of the Tracers' transmissions). Since the distance map does not have to be error-free or synchronized among Clients, the non-guaranteed delivery characteristic of IP multicast is acceptable. Where IP multicast is not available, a host-level multicast protocol such as Yallcast or Narada [49, 50] may be used.

4 Performance Evaluation

To study the various algorithms presented in this paper prior to the deployment of IDMaps on the Internet, we conduct some simulations on generated network topologies. In this section, we give an overview of the three topology generation processes used in this study. Then we describe how we "deploy" IDMaps on the generated topologies. Finally, we describe how the performance metric, P_{app}, is computed.

4.1 Topology Generation

We use three models to generate network topologies: the Waxman model [51], Tiers [52], and a model based on AS-connectivity ("Inet") observed from data collected on the Internet. All three models work by first placing a given number of nodes, N, on a plane of dimensions s distance units by s distance units. In this paper, we always choose s that is one order of magnitude larger than N. The cost of each edge in the generated network is the Euclidean distance between the two end nodes, the cost of a path between two nodes is the sum of the edge costs. The three models differ in their determination of (1) how nodes are placed on the plane, (2) how many other nodes each node should directly connect to (a.k.a. node connectivity degree, or simply, node degree), and (3) which edges are selected to form the network. These three models produce networks with very different characteristics, including the node degree distribution,

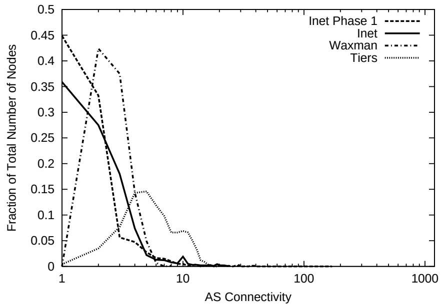


Figure 10: Distribution of node degree.

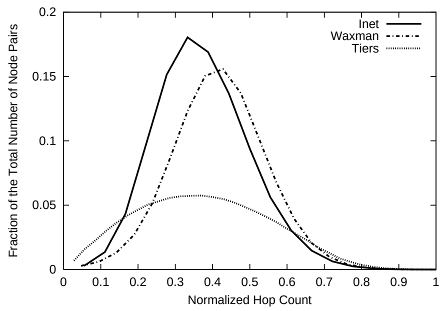


Figure 11: Distribution of hop count.

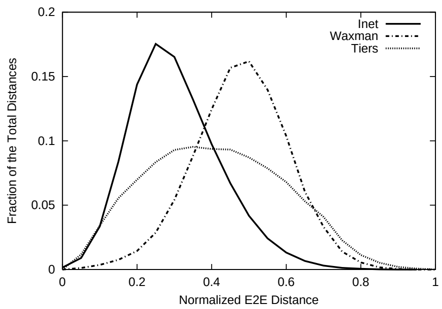


Figure 12: Distribution of end-to-end distance.

the end-to-end delay distribution, and the network diameter. We next briefly describe the three topology generation processes and the parameters used in this paper; a more detailed description is given in [2].

Waxman. Under the Waxman model, the probability that an edge is formed between nodes u and v is given by $P(u,v) = \beta e^{(-d(u,v)/L\alpha)}$, where $d(u,v)$ is the Euclidean distance between u and v and L is the maximum possible distance between any two nodes on the generated network. A random number between 0 and 1 is generated. The edge is added if this number is smaller than the computed $P(u,v)$. We use $\alpha = .6$ and $\beta = .0015$, which give us a mean node degree of around 2.2. Finally, a spanning tree is computed, adding edges where necessary, such that the final network is a connected graph.

Tiers. Tiers places the given N nodes on the $s \times s$ plane ensuring that nodes are not within a radius of $.5s/\sqrt{N}$ of each other. Edges are added by first computing a minimum spanning tree connecting the nodes. Each node is then connected to as many other nodes as specified by its node degree. In selecting a node's neighbors, the ones nearest to the node that have not yet met their node degrees are selected first. The connectivity degree of each node is randomly distributed with a maximum node degree (WAN redundancy, in Tiers parlance) of r. Fig. 10 shows the distribution of the resulting node degree on our 1,000 node network for $r = 20$.

Inet. Motivated by the recent findings on the power-law relationship between node (Autonomous System) vertex degree and the frequency distribution of the vertex degrees [53], we designed a topology generator to produce graphs with such power-law relationship. We call our generator Inet and described it more fully in [2].[8] Briefly, we first connect the $\tau (= 5)$ nodes with the highest vertex degrees into a full-mesh. Then to simulate the geographically dispersed nature observed of large ISPs' networks, we expand the most connected κ nodes into networks with η randomly placed nodes each. In this paper, we use $\kappa = 10$ for $N = 1,000$, $\kappa = 35$ for $N = 4,200$, and $\eta = 20$ in all cases.

For each of the three models, we generate thirty-one networks with 1,000 nodes and thirty-one networks with 4,200 nodes each, for a total of 186 random topologies. Fig. 10 shows the distribution of node degree of three generated topologies, one each from the three models. Table 2 lists the diameter and the maximum node degree of the three networks, and Figs. 11 and 12 show the distribution of hop counts and end-to-end distance between all node pairs in the three networks respectively. In Fig. 11 (12) the hop counts (e2e distances) are normalized by the diameter of the respective network.

4.2 Simulating IDMaps Infrastructure

Once a network is generated, we "build" an IDMaps infrastructure on it. In this section, we describe how the various Tracer placement and distance map computation algorithms and heuristics are implemented.

[8]Our Inet topology generator is available for download from http://topology.eecs.umich.edu/inet/.

Model	Hop Count	E2E Distance	Max Node Degree
Waxman	21	77,655	8
Tiers	35	14,635	20
Inet	18	78,379	24

Table 2: Network diameter and max node degree

Tracer Placement. In Section 3.2, we described two graph-theoretic approaches to Tracer placement and three heuristics. We implement the graph-theoretic approaches, we compute Tracer placement using the algorithms described. To implement Stub-AS Tracer placement, given B Tracers, we pick B nodes with the lowest degrees of connectivity to host Tracers. Conversely, for Transit-AS placement, we pick B nodes with the highest degrees of connectivity. We implement Mixed Tracer placement by giving equal probability to all nodes on the generated network to host a Tracer.

Distance Map Computation. A distance map consists of two parts: Tracer-Tracer VLs and Tracer-AP VLs. Each Tracer advertises the VLs it traces. We do not simulate VL tracing and advertisement or AP discovery in this study, and we only simulate a single IDMaps Client. Since IDMaps Clients operate independently, the use of a single IDMaps Client has no loss of generality. The simulated IDMaps client has a full list of Tracers and their locations. The Tracer-Tracer part of the distance map is computed either assuming a full-mesh among all Tracers, or by executing the original t-spanner algorithm shown in Fig. 7.

Each AP (node) can be traced by one or more Tracers. When each AP is traced by a single Tracer, the Tracer nearest to an AP is assigned to trace to the AP. If an AP is to be traced by more than one Tracer, Tracers are assigned to the AP in order of increasing distance. In our simulations, we assume all edges are bidirectional, and paths have symmetric and fixed costs. We study the effect of measurement error and stability on IDMaps' performance in a related work [54].

Once a distance map is built, the distance between two APs, A and B is estimated by summing up the distance from A to its nearest Tracer T_A, the distance from B to its nearest Tracer T_B, and the distance between T_A and T_B. When a full-mesh is computed between Tracers, the T_A to T_B distance is exactly the length of the shortest path between them on the underlying network. Otherwise, they are computed from the t-spanner. If A and B have multiple Tracers tracing to them, the distance between A and B is the shortest among all combinations of Tracer-AP VLs and Tracer-Tracer VLs for the Tracers and APs involved.

4.3 Performance Metric Computation

Ultimately, IDMaps will be evaluated by how useful its distance information is to applications. We evaluate the performance of IDMaps using nearest mirror selection as a prototypical application and adopt an application-level performance metric, P_{app}, which measures how often the determination of the nearest mirror to a client, using the information provided by IDMaps, results in a correct answer, i.e., the mirror the client would have been redirected based on a shortest path tree constructed from the underlying physical topology. Incidentally, the localized distance measurement service (see Section 2.2) such as provided by [9, 18] in effect constructs a shortest-path tree from each client (or stub network) to all mirrors. P_{app} thus can be considered as comparing the performance of IDMaps against the localized services in the best case scenario for the localized services, i.e., the distances from the clients to all mirrors are known a priori and are obtained at no cost. Performance comparison between localized services and IDMaps in the common case must take into account the shortcomings of localized services (see Section 2.2), chief among which is the time lag in obtaining distance to "uncharted" parts of the Internet due to the "on-demand" nature of the

Topology	Placement	B	T-T Map	T/AP
Waxman	Stub-AS	10	full-mesh	1
Tiers	Transit-AS	20	2-spanner	2
Inet	Mixed	40	10-spanner	3
	Min K-center	100		
	k-HST			

Table 3: Simulation Parameters

service, and the additional cost of collecting distance information due to the lack of information sharing between clients and the cost of maintaining each instance of the localized service.

Considering, however, that the goal of IDMaps is not to provide precise estimates of distances between hosts on the Internet, but rather estimates of relative distances between a source (e.g. a client) and a set of potential destinations (e.g. server mirrors), we adopt a lax version of this measure in this paper, as follows. In each simulation experiment, we first place n (from 3 to 24) server mirrors in our simulated network. We place the mirrors such that the distance between any two of them is at least $1/n$ the diameter of the network. We consider all the other nodes on the network as clients to the server and compute for each client the nearest mirror according to the distance map obtained from IDMaps and the nearest mirror according to the actual topology. For a given n-mirror placement, we compute P_{app} as the percentage of correct IDMaps' answers over total number of clients.

On the Internet, a client served by a server 15 ms away would probably not experience a perceptible difference from being served by a server 35 ms away, or that a server 200 ms away will not appear much closer than one 150 ms away, we consider IDMaps' server selection correct as long as the distance between a client and the nearest mirror determined by IDMaps is within a factor of λ times the distance between the client and the actual nearest mirror (in this paper, we use $\lambda = 2$).

We repeat this procedure for 1,000 different n-mirror placements, obtaining 1,000 P_{app} values in each experiment. In the next section, we present our simulation results by plotting the complementary distribution function[9] of these P_{app} values.

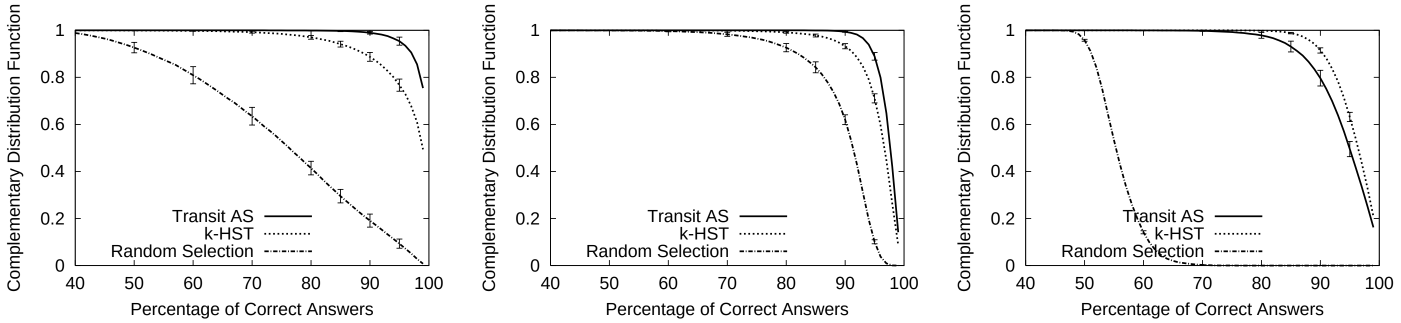
4.4 Simulation Results

Table 3 summarizes the parameters of our simulations. The heading of each column specifies the name of the parameter, and the various values tried are listed in the respective column. The column labeled "Topology" lists the three models we use to generate random topologies. The "Placement" column lists the Tracer placement algorithms and heuristics. The "B" columns lists the number of Tracers we use on 1000-node networks. The "T-T Map" column lists the methods used to compute the Tracer-Tracer part of the distance map. The "T/AP" column lists the number of Tracers tracing to an AP. We experimented with almost all of the 540 possible combinations of the parameters on 1,000 node networks and several of them on 4,200 node networks. In addition, we also examined the case of having more mirrors for a few representative simulation scenarios.

The major results of our study are:

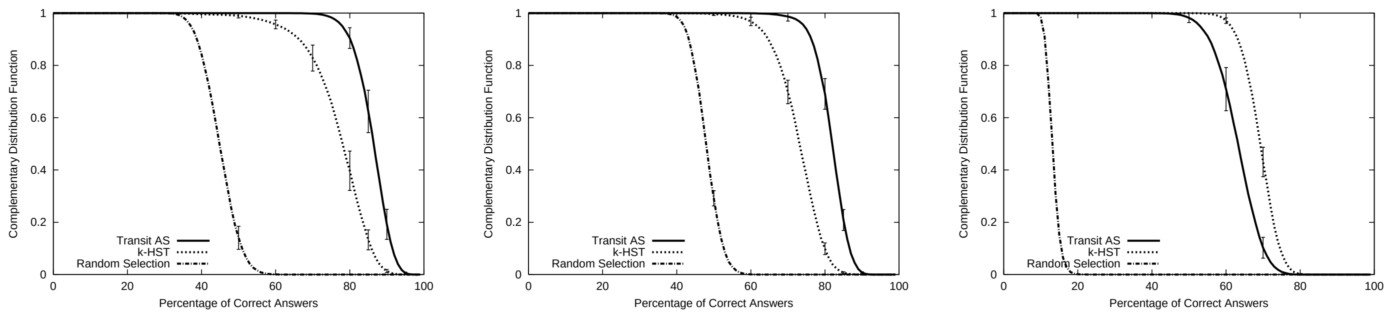
1. Mirror selection using IDMaps gives noticeable improvement over random selection.
2. Network topology can affect IDMaps' performance.
3. Tracer placement heuristics that do not rely on knowing the network topology can perform as well as or better than algorithms that require a priori knowledge of the topology.
4. Adding more Tracers (over a 2% threshold) gives diminishing return.

[9] The complementary distribution function, $\bar{F}(x) = 1 - F(x)$, where $F(x)$ is the cumulative distribution function of the random variable x.



a. Inet b. Waxman c. Tiers

Figure 13: 3-Mirror selection on 1,000-node network with 10 Tracers.



a. Inet b. Waxman c. Tiers

Figure 14: 24-Mirror selection on 1,000-node network with 10 Tracers.

5. Number of Tracer-Tracer VLs required for good performance can be on the order of B with a small constant.
6. Increasing the number of Tracers tracing to each AP improves IDMaps' performance with diminishing return.

These results apply to both the 1,000-node and 4,200-node networks. We present simulation data substantiating each of the above results in the following subsections. Due to space constraints, we are not able to include data confirming some of these results on the Internet [55].

4.4.1 Mirror Selection

Results presented in this subsection are obtained from simulations on 1,000-node randomly generated topologies. In all cases, 3 mirrors are manually placed on the network, the number of Tracers deployed is 10 (1% of nodes), the distance maps are built by computing full-meshes between the Tracers, with only a single Tracer tracing to each AP.

We compare the results of random selection against selection using the distance map generated by IDMaps. The metric of comparison is P_{app}. Each line in Fig. 13 shows the complementary distribution function of 1,000 P_{app} values as explained in the previous section. Each line is the average of 31 simulations using different random topologies, the error bars show the 95% confidence interval. For example, the line labeled "Transit-AS" in Fig. 13a shows that on an Inet-generated topology, when mirrors are selected based on the distance map computed from Tracers placed by the Transit-AS heuristic, the probability that at least 80% of all clients will be directed to the "correct" mirror is 100% (recall our definition of correctness from the previous section); however, the probability that up to 98% of all clients will be directed to the correct mirror is only 85%. We start the x-axis of the figure at 40% to increase legibility. The line labeled "k-HST" is the result when the k-HST algorithm is used to place Tracers. The k-HST algorithm requires knowledge of the topology (see Section 3.2.1). The line labeled "Random Selection" is the result when mirrors are randomly selected without using a distance map. As expected, it performs well for less than 40% correctness and the performance deteriorates beyond 60% correctness. Mirror selection using distance maps outperforms random selection regardless of the Tracer placement algorithm.

We include only the best and worst performing Tracer placement algorithms in Figs. 13 for legibility of the graphs. The relative performance of the various placement algorithms is presented in Section 4.4.3. Fig. 14 shows results from simulations with 24 mirrors. Qualitatively, these results agree with our conclusion that mirror selection using distance maps outperforms random selection.

4.4.2 Effect of Topology

Figs. 13b and 13c show the results of running the same set of simulations as in the previous section, but on topologies generated from the Waxman and Tiers models respectively. Again, the error bars on each figure shows the 95% confidence interval computed from 31 randomly seeded topologies. While mirror selection using a distance map provides better performance than random selection in all cases, performance on the Tiers generated topology exhibit a qualitatively different behavior than those in the other two topologies. For example, the Transit-AS heuristic gives better IDMaps performance than the k-HST algorithm on topologies generated from the Inet and Waxman models, but not so in the topology generated from Tiers.

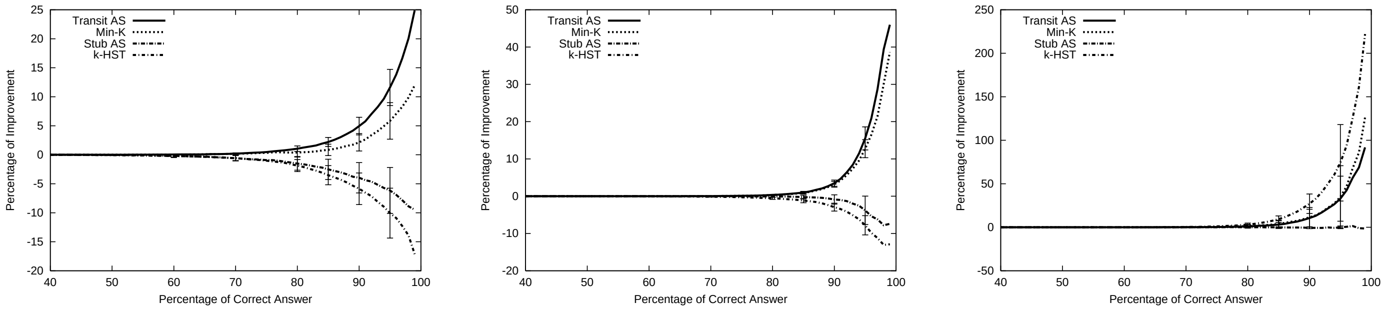
We offer a hypothesis for the relatively poor performance of random mirror selection on Tiers topology. Fig. 12 shows that almost all the end-to-end distances in Inet generated network fall between 20% and 60% of the network diameter. When we randomly pick two distances from this network, it is highly likely that they will fall within this range. Consequently, one distance will be no more than 3 times longer than the other. So given our definition of the performance metric, even the random selection can give acceptable performance. As can be seen by comparing Fig. 13b against Figs. 13a and 13c, this is more evident in the network generated from the Waxman model, where the distances fall between 30% and 70% of the network diameter. However, the distance distribution for the Tiers topology is much more dispersed, and the range is between 10% and 70% of the diameter. It is much harder for two randomly picked distances to be close within a factor 3. This is corroborated by the poor results "Random Selection" returns. We note again that despite the significant differences in the three models, IDMaps is able to provide noticeable improvements in mirror selection in all three cases.

4.4.3 Performance of Placement Algorithms

To compare the relative performance of the various Tracer placement algorithms and heuristics, we repeat the same simulations as in the previous two subsections, once for each placement algorithm. Then using the complementary distribution function of the P_{app} values obtained from running the Mixed placement algorithm as the baseline, we compute the improvement of each placement algorithm relative to Mixed placement. The results are presented in Figs. 15a, 15b, and 15c for networks generated using the Inet, Waxman, and Tiers models respectively. There is no clear winning placement algorithm across all topologies, but the minimum K-center algorithm and Transit placement consistently performed well in all three topologies. In general, the simple heuristics can often perform as well as the graph theoretic placement strategies. In [55] we also present results of applying the graph theoretic placement algorithms on distance maps computed from the Transit-AS heuristics.

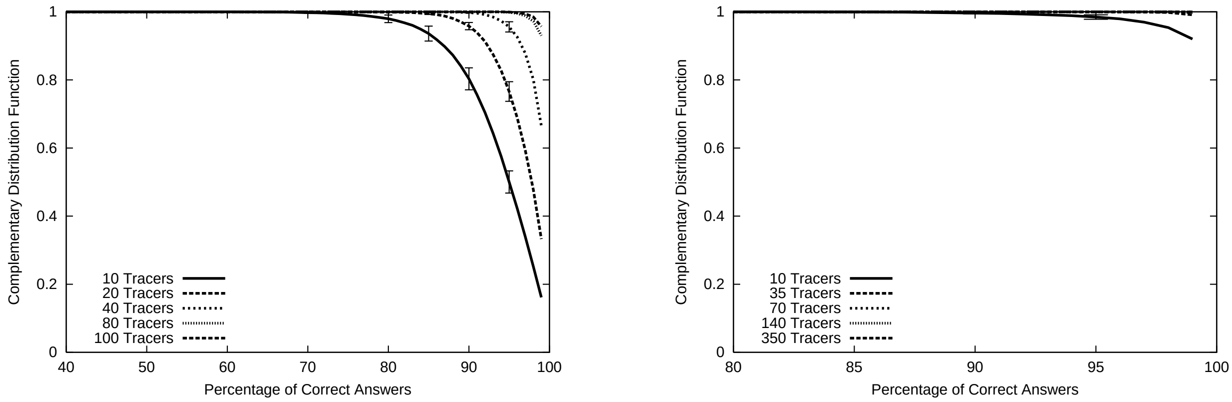
4.4.4 Having More Tracers

In this subsection, we study the effect of increasing the number of Tracers on IDMaps' performance. Fig. 16a shows the results of running the Transit-AS placement algorithm on a 1,000-node network gen-



a. Inet b. Waxman c. Tiers

Figure 15: Improvement of placement algorithms over the "Mixed" algorithm on 1,000-node network, 10 Tracers.



a. 1,000-node Tiers network b. 4,200-node Inet network

Figure 16: Mirror selection using IDMaps with varying number of Tracers.

erated using the Tiers model. Increasing the number of Tracers from 10 to 20 improves performance perceptibly, with diminishing improvements for further increases. Comparing Fig. 16a against Fig. 13c from Section 4.4.2 we see that increasing the number of Tracers from 10 to 20 makes the performance of IDMaps using the Transit-AS placement algorithm comparable to that using the k-HST algorithm with 10 Tracers.

Fig. 16b shows the results of running the Transit-AS placement algorithm on a 4,200-node network generated using the Inet model. Again, we see a perceptible improvement in IDMaps performance when the number of Tracers increases from 10 to 35, with diminishing improvements for further increases. Also of significance is that having only .2% of all nodes serving as Tracers already provides correct answer 90% of the time with very high probability. To the extent that larger networks means denser networks,[10] a Tracer can serve more nodes in a larger network than it does in smaller networks. Thus to achieve the same IDMaps performance, the number of Tracers needed to serve a larger network does not necessarily increase as fast as the increase in network size. Overall, we do not need a large scale IDMaps deployment to realize an improvement in the metric of interest, P_{app}.

[10]The number of hosts and ASs on the Internet has been growing very fast over the past decade, but the diameter of the Internet, i.e., the longest path between two points on the Internet, has stayed roughly the same.

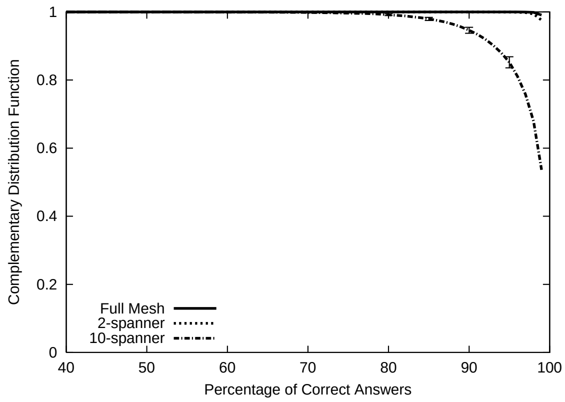


Figure 17: Effect of t-spanner on 1,000-node I net network with 100 Tracers.

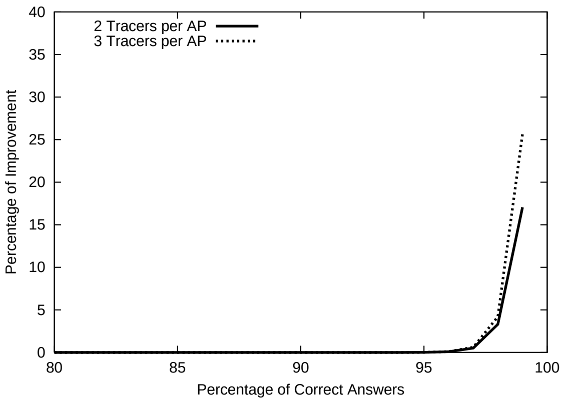


Figure 18: Mirror selection on 1,000-node Waxman network with 2 and 3 Tracers / AP.

Model	Inet		Waxman		Tiers	
Placement	2-spanner	10-spanner	2-spanner	10-spanner	2-spanner	10-spanner
Stub	628	198	654	198	268	202
Mixed	520	200	466	198	264	200
Transit	434	198	386	202	262	198
Min K-center	402	198	434	196	266	202

Table 4: Number of VLs in t-spanners of 100 Tracers .

4.4.5 Distance Map Reduction

In all the simulations reported so far, the distance maps are built by computing full-mesh Tracer-Tracer VLs. Figure 17 shows the results of running the Transit-AS algorithm to place 100 Tracers on a 1,000-node network generated using the Inet model, with Tracer-Tracer VLs computed as a full-mesh and as t-spanners. For $t = 2$, there is no perceptible difference in performance; for $t = 10$, the performance is worse. Qualitatively similar results are observed for topologies generated by the Waxman and Tiers models, with worse performance for $t = 10$ in the Tiers case.

Using a t-spanner in place of a full-mesh can significantly reduce the number of Tracer-Tracer VLs that must be traced, advertised, and stored. Table 4 shows that for all the topologies we experimented with, the number of VLs used by both 2- and 10-spanners are on the order of B with a small constant multiplier. In contrast, the number of VLs required to maintain a full-mesh for $B = 100$ is 4,950 edges. (The theoretical upper bound on the number of edges in a t-spanner is O($B^(1 + 1/t)$)).

4.4.6 Multiple Tracers per AP

In all our simulations so far we have assumed that only a single Tracer traces each AP. We showed in Section 3.3.2 (Fig. 8) that in some cases having more than one Tracer tracing an AP may result in better distance estimates. We now present some performance results from scenarios in which there are 2 or 3 Tracers tracing each AP. On 1000-node networks, we place 100 Tracers using the Transit-AS algorithm, and compute a full-mesh for Tracer-Tracer VLs. Using the performance of IDMaps where only one Tracer traces each AP as the baseline, we compute the percentage improvement of increasing the number of Tracers per AP.

Fig. 18 shows that on a 1000-node network generated using the Waxman model, compared to having only one Tracer per AP, the probability of having at least 98% correct answer is increased by 17% when each AP is traced by 2 Tracers, and is increased by 25% when each AP is traced by 3 Tracers. We only consider up to 3 Tracers per AP since currently 85% of ASs in the Internet have degree of connectivity of at most 3 [53].

5 Conclusion

It has become increasingly evident that some kind of distance map service is necessary for distributed applications in the Internet. However, the question of how to build such a distance map remains largely unexplored. In this paper, we propose a global distance measurement infrastructure called IDMaps and tackle the question of how it can be placed on the Internet to collect distance information.

In the context of nearest mirror selection for clients, we showed that, significant improvement over random selection can be achieved using placement heuristics that do not require the full topological knowledge. In addition, we showed that the IDMaps overhead can be minimized by grouping Internet addresses into APs to reduce the number of measurements, the number of Tracers required to provide useful distance estimations is rather small, and applying t-spanners to the Tracer-Tracer VLs can result in linear measurement overhead with respect to the number of Tracers. Overall, this study has provided positive results to demonstrate that a useful Internet distance map service can indeed be built scalably.

References

[1] P. Francis et al., “An Architecture for a Global Internet Host Distance Estimation Service,” Proc. of IEEE INFOCOM 1999, March 1999.

[2] S. Jamin et al., “On the Placement of Internet Instrumentation,” Proc. of IEEE INFOCOM 2000, March 2000.

[3] B. Kantor and P. Lapsley, “Network news transfer protocol: A proposed standard for the stream-based transmission of news,” RFC 977, Internet Engineering Task Force, Feb. 1986.

[4] S. Bhattacharjee et al., “Application-Layer Anycasting,” Proc. of IEEE INFOCOM ’97, Apr. 1997.

[5] W. Stevens, TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1994.

[6] S.E. Deering and D.R. Cheriton, “Multicast Routing in Internetworks and Extended LANs,” ACM Transactions on Computer Systems, vol. 8, no. 2, pp. 85–110, May 1990.

[7] K. Moore, J. Cox, and S. Green, “Sonar - a network proximity service,” Internet-Draft, url: http://www.netlib.org/utk/projects/sonar/, Feb. 1996.

[8] P. Francis, “Host proximity service (hops),” Preprint, Aug. 1998.

[9] M. Stemm, R. Katz, and S. Seshan, “A Network Measurement Architecture for Adaptive Applications,” Proc. of IEEE INFOCOM ’00, pp. 2C–3, Mar. 2000.

[10] N. Miller and P. Steenkiste, “Collecting Network Status Information for Network-Aware Applications,” Mar. 2000.

[11] C. Labovitz et al., “Internet performance measurement and analysis project,” url: http://www.merit.edu/ipma/, 1998.

[12] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, “An Architecture for Large-Scale Internet Measurement,” IEEE Communications Magazine, vol. 36, no. 8, pp. 48–54, August 1998.

[13] A. Adams et al., “The Use of End-to-End Multicast Measurements for Characterizing Internal Network Behavior,” IEEE Communications Magazine, pp. 152–158, May 2000.

[14] R. Govindan and H. Tangmunarunkit, “Heuristics for Internet Map Discovery,” Proc. of IEEE INFOCOM 2000, 2000.

[15] R. Siamwalla, R. Sharma, and S. Keshav, “Discovering Internet Topology,” Unpublished manuscript.

[16] K. C. Claffy and D. McRobb, “Measurement and Visualization of Internet Connectivity and Performance,” http://www.caida.org/Tools/Skitter/.

[17] Hal Burch and Bill Cheswick, “Mapping the Internet,” IEEE Computer, vol. 32, no. 4, pp. 97–98, April 1999.

[18] Robert L. Carter and Mark E. Crovella, “Server Selection using Dynamic Path Characterization in Wide-Area Networks,” Proc. of IEEE INFOCOM ’97, April 1997.

[19] Robert L. Carter and Mark E. Crovella, “Dynamic Server Selection using Bandwidth Probing in Wide-Area Networks,” Tech. Rep. BU-CS-96-007, Boston University, March 1996.

[20] Mark E. Crovella and Robert L. Carter, “Dynamic Server Selection in the Internet,” Proc. of IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems’95, June 1995.

[21] Sandra G. Dykes, Clinton L. Jeffery, and Kay A. Robbins, “An Empirical Evaluation of Client-Side Server Selection Algorithms,” Proc. of IEEE INFOCOM ’2000, 2000.

[22] James D. Guyton and Michael F. Schwartz, “Locating Nearby Copies of Replicated Internet Servers,” Proc. of ACM SIGCOMM ’95, pp. 288–298, 1995.

[23] A. Myers, P. Dinda, and H. Zhang, ““Performance Characteristics of Mirror Servers on the Internet”,” Proc. of IEEE INFOCOM ’99, Mar. 1999.

[24] Katia Obraczka and Fabio Silva, “Looking at Network Latency for Server Proximity,” Tech. Rep. USC-CS-99-714, University of Southern California, 1999.

[25] M. Sayal, Y. Breibart, P. Scheuermann, and R. Vingralek, “Selection Algorithms for Replicated Web Servers,” Performance Evaluation Review, vol. 26, no. 3, pp. 44–50, Dec 1998.

[26] Anees Shaikh, Renu Tewari, and Mukesh Agrawal, “On the Effectiveness of DNS-based Server Selection,” Proc. of IEEE INFOCOM ’2001, 2001.

[27] P.R. McManus, “A passive system for server selection within mirrored resource environments using as path length heuristics,” http://proximate.appliedtheory.com/, Jun. 1999.

[28] Zongming Fei, Samrat Bhattacharjee, Ellen W. Zegura, and Mostafa Ammar, “A novel server selection technique for improving the response time of a replicated service,” Proc. of IEEE INFOCOM, 1998.

[29] V. Cardellini, M. Colajanni, and P.S. Yu, “Dynamic Load Balancing on Web Server Systems,” IEEE Internet Computing, pp. 28–39, May-June 1999.

[30] T. Bates, “The CIDR Report,” url: http://www.employees.org/˜tbates/cidr-report.html, June 1998.

[31] Y. Rekhter and T. Li, “A border gateway protocol 4 (bgp-4),” RFC 1771, Internet Engineering Task Force, Mar. 1995.

[32] R. Govindan and H. Tangmunarunkit, ““Heuristics for Internet Map Discovery”,” Proc. of IEEE INFOCOM 2000, pp. 3C–2, Mar. 2000.

[33] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, Cambridge, MA: MIT Press, 1990.

[34] S. Hotz, “Routing Information Organization to Support Scalable Interdomain Routing with Heterogenous Path Requirements,” Tech. Rep. Ph.D. Thesis, Univ. of Southern California, CS Dept., 1994.

[35] J.D. Guyton and M.F. Schwartz, “Locating Nearby Copies of Replicated Internet Servers,” Proceedings of ACM SIGCOMM, August 1995.

[36] S. Savage et al., “The End-to-End Effects of Internet Path Selection,” Proc. of ACM SIGCOMM ’99, Sep. 1999.

[37] V. Paxson, “End-to-End Routing Behavior in the Internet,” Proc. of ACM SIGCOMM ’96, pp. 25–38, Aug. 1996.

[38] Merit Networks, “Npd (network probe daemon) project,” URL: http://www.merit.edu/ipma/npd/, 1996.

[39] MirrorNet, “Multiple traceroute gateway,” URL: http://www.tracert.com, 1999.

[40] C. Toregas, R. Swain, C. Revelle, and L. Bergma, “The location of emergency service facilities,” Operstiond Research, vol. 19, pp. 1363–1373, 1971.

[41] Y. Bartal, “Probabilistic approximation of metric space and its algorithmic applications,” in 37th Annual IEEE Symposium on Foundations of Computer Science, October 1996.

[42] V. Vazirani, Approximation Methods, Springer-Verlag, 1999.

[43] Baruch Awerbuch and Yuval Shavitt, “Topology aggregation for directed graphs,” IEEE/ACM Transactions on Networking, vol. 9, no. 1, pp. 82–90, February 2001.

[44] M.R. Garey and D.S. Johnson, Computers and Intractability, NY, NY: W.H. Freeman and Co., 1979.

[45] I. Althöfer, G. Das, D. Dopkin, D. Joseph, and J. Soares, “On sparse spanners of weighted graphs,” Discrete and Computational Geometry, vol. 9, pp. 81 – 100, 1993.

[46] D. Peleg and E. Upfal, “A tradeoff between space and efficiency for routing tables,” in 20th ACM Symposium on the Theory of Computing, May 1988, pp. 43 – 52.

[47] D. Peleg and A.A. Schäffer, “Graph spanners,” Journal of Graph Theory, vol. 13, no. 1, pp. 99 – 116, 1989.

[48] L. Cai, “NP-completeness of minimum spanner problems,” Discrete Applied Mathematics, vol. 48, pp. 187 – 194, 1994.

[49] P. Francis, “Yallcast: extending the internet multicast architecture,” Tech. Rep., ACIRI/ICSI, September 1999, http://www.aciri.org/yoid/.

[50] Y-H. Chu, S. Rao, and H. Zhang, “A case for endsystem-only multicast,” Preprint. http://www.cs.cmu.edu/~kunwadee/research/other/en dsystem-index.html, December 1999.

[51] B.M. Waxman, “Routing of Multipoint Connections,” IEEE Journal of Selected Areas in Communication, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[52] M. Doar, “A Better Model for Generating Test Networks,” Proc. of IEEE GLOBECOM, Nov. 1996.

[53] M. Faloutsos, P. Faloutsos, and C. Faloutsos, “On Power-Law Relationships of the Internet Topology,” Proc. of ACM SIGCOMM ’99, pp. 251–262, Aug. 1999.

[54] IDMaps Project, “Work in progress,” URL: http://idmaps.eecs.umich.edu/, 1999.

[55] S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt, “Constrained Mirror Placement on the Internet,” Proc. of IEEE INFOCOM ’2001, April 2001.