An Architecture for a Global Internet Host Distance Estimation Service

Paul Francis
NTT Software Laboratories
3-9-11 Midori-cho
Musashino-shi, Tokyo, 180
francis@slab.ntt.co.jp

Sugih Jamin*
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
jamin@eecs.umich.edu

Vern Paxson
Network Research Group
LBNL
Berkeley, CA 94720
vern@ee.lbl.gov

Lixia Zhang
CS Department
UCLA,
Los Angeles, CA 90095
lixia@cs.ucla.edu

Daniel F. Gryniewicz
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
dang@eecs.umich.edu

Yixin Jin
CS Department
UCLA,
Los Angeles, CA 90095
yjin@cs.ucla.edu

Abstract

There is an increasing need for Internet hosts to be able to quickly and efficiently learn the distance, in terms of metrics such as latency or bandwidth, between Internet hosts. For example, to select the nearest of multiple equal-content web servers. This paper explores technical issues related to the creation of a public infrastructure service to provide such information. In so doing, we suggests an architecture, called IDMaps, whereby Internet distance information is distributed over the Internet, using IP multicast groups, in the form of a virtual distance map. Systems listening to the groups can estimate the distance between any pair of IP addresses by running a spanning tree algorithm over the received distance map. We also present the results of experiments that give preliminary evidence supporting the architecture. This work thus lays the initial foundation for future work in this new area.

*At UM, this research is supported in part by equipment grants from Sun Microsystems Inc., Digital Equipment Corp., and Intel Corp.

1 Introduction

It is increasingly the case that a given Internet interaction could be satisfied by one of a number of Internet hosts. Examples range from short-lived interactions such as a single web page access to any one of multiple equal-content web servers to a long-term peering relationship between two news (NNTP) servers [1].

In any such interaction, all other things being equal, it is advantageous to access the "nearest" choice. By near we mean in terms of Internet performance metrics, such as low latency or high bandwidth. Even when all other things are not equal, such as the case where different web servers have different response times, it is still useful to include distance to each candidate host as one of several criteria for making a selection [2].

One approach to obtaining this distance information is for the initiating host to measure it itself, using either unicast (`ping`, `traceroute` [3]) or multicast (expanding ring search [4]) tools. While these tools have a wide range of uses, their utility is generally limited by their overhead. For instance, the cost of running a single `traceroute` can exceed the cost of the web page access itself. More important still, a large number of hosts making independent and frequent measurements could have a severe impact on performance overall. Ideally, measurements made by one system (host or router) should be made available, at low cost, to other hosts.

A useful general service for the Internet would be one whereby a host could 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 those of the gains achieved by using the service. A simple protocol for such a service (SONAR) was discussed in the IETF as early as February 1996 [5], and in April 1997 as a more general service called HOPS (Host Proximity Service) [6]. Both of these efforts proposed lightweight client/server query/reply protocols along the lines of a DNS query/reply. Both also required that the server be able to produce an answer in a very short time—preferably, though not necessarily, by using information already stored locally.

This paper is concerned with the problem of how servers in such a SONAR/HOPS service can obtain the distance information needed to answer queries. Specifically, we explore the following questions:

1. Which systems originally produce the distance information, and how is it produced?
2. How does the distance information get from these producing systems to the servers?
3. What form does the distance information take, and how is it used to produce answers for specific pairs of Internet hosts?

Several efforts to obtain various sorts of distance information and other Internet characteristics are currently being undertaken, e.g. the SPAND, FELIX, Octopus, IPMA, NIMI, and MINC projects [7, 8, 9, 10, 11, 12]. These projects serve various purposes, such as general Internet characterization, maintenance, and end-system distance estimation. Much of this work applies to some aspects of the first question above. None of this work, however, is being done in the context of a general SONAR/HOPS service. As such, it does not really get at the latter questions or the aspects of the first question that pertain to a general SONAR/HOPS service.

This paper, then, takes a first stab at exploring a global, as opposed to end-host based, architecture for Internet host distance estimation and distribution. We discuss basic aspects of the questions outlined above, and based on this propose a general architecture for an underlying service that provides the basic information used by a SONAR/HOPS service. This underlying service is called IDMaps, for Internet Distance Map Service.

Guided in part by what applications need, but more by basic limitations in technology, section 2 outlines the goals of this work. Section 3 starts with a broad discussion of the fundamental characteristics of the problem and the nature of the solution. It presents and develops two variants of the solution, "Hop-by-Hop" and "End-to-End"—the latter of which is more elegant but less well understood—and ends with a discussion of their pros and cons. Section 4 describes the results of a preliminary analysis that tests some of the basic assumptions of the End-to-End solution. These results suggest that it has promise, and encourage us to move forward with a more detailed design and implementation. Finally, Section 5 discusses some of the many open issues remaining.

2 IDMaps Goals

It is difficult to state precise goals for even a SONAR/HOPS service, much less the IDMaps service. This is because SONAR/HOPS could be called upon to support a wide range of applications, from web clients that want a web server from which to retrieve a small amount of data once, to Network Time Protocol (NTP) servers that want to establish a long term association with a peer system.

We could go through the exercise of making a list of potential applications and extracting out the most demanding requirements (and to some extent, at least informally, we have done this), but in fact this exercise quickly reveals that we can't even come close to satisfying all conceivable requirements. For instance, 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 a host wishing to select a server supplying a live video feed, for example.

Rather, we've taken somewhat 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. Following is a discussion of the resulting goals.

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 that it is easy to provide. First, it is easy to measure. A small number of packets can produce a good rough estimate. Second, it is relatively independent of the exact path between the two hosts, though by no means fully independent.

This relative path independence is important because the only way to know the exact path between two Internet hosts is to measure it from those two hosts, for instance using `traceroute`. Even routers don't know the exact path between any two Internet hosts, much less other hosts. By definition, a general-purpose IDMaps service does not have access to such exact path information.

Bandwidth is clearly important for many applications, but compared to latency it is hard to provide. This is primarily because it is more sensitive to the exact path—a single low-bandwidth link (or for that matter, heavily loaded link) dictates the bandwidth for the whole path. In addition, it can be somewhat more difficult to measure. Nevertheless, there is some hope of being able to provide reasonable bandwidth estimates. For instance, we can expect there to be some uniformity of bandwidths across the Internet backbone networks—one doesn't expect to find a very thin pipe in the middle of an otherwise thick-pipe topology.

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 do not expect the information provided by a SONAR/HOPS service to be highly accurate. For reasons discussed throughout this paper, we believe highly accurate distance estimates (say, within 5% or 10% of the average distance that could be measured by the end-host itself) are impossible to achieve scalably.

Our goal, rather, is to obtain accuracy to within a factor of 2 with very high probability, and often do better than that. While this may seem bad at first glance, this level of accuracy would, for instance, allow an application to usefully select between web servers that were reported to be 20ms, 100ms, and 500ms away. The (reported) 20ms server, which might actually be as far as 40ms, would still be better than the (reported) 100ms server, which might be as close as 50ms. Even being able to discriminate like this among systems that are very close, very far, or somewhere in between (something we refer to as distance triage) is useful for a wide range of applications.

Applications that require more accurate knowledge may at least use this rough information as a guide to making further measurements of their own. (Indeed, for most applications, Internet distance will be only one of several factors that may effect the choice of destination host. For instance, the 100ms host in the above example could still turn out to be the best choice, for instance because the 20ms host was heavily loaded.)

Timeliness of the Distance Information Here we must consider two kinds of distance information—load sensitive and "raw" (the distance seen assuming no load on the network, which generally can be measured by saving the best of a number of measurements).

The raw distance information will be on the order of hours if not days old. In other words, the distance information will not reflect network failures, and will only slowly reflect "permanent" topology changes. The former would require either 1) very frequent monitoring of the Internet, which is out of the question, or 2) participation from the routers themselves, which we don't assume. Even if network failure information were available, given the heavy load that BGP updates exert on routers, even after significant damping of network change information [13], it is clear that the IDMaps infrastructure could not carry such information.

As for load sensitive distance information, it is questionable as to whether it should be provided at all. Certainly instantaneous or near-instantaneous (within 15 or 20 seconds) load information is impossible, given the highly variable nature of Internet performance.

There is a concern that any kind of load sensitive distance information will generate a feedback loop oscillation, where applications choose servers based on IDMaps distance information, which in turn effects the distance information. Given the wide range of applications that might make use of IDMaps load sensitive information, and the difficulty of understanding their behavior, it may be arbitrarily difficult to dampen or eliminate such oscillations.

It may be useful to provide raw distance information on a time-of-day (and time-of-week etc.) basis, to more accurately reflect predictable time-of-day variations, but doing so is not incompatible with a timeliness goal of hours or days.

Total Magnitude of Distance Information As stated in the introduction section, the proposed SONAR/HOPS services assume response times on the order of DNS response time. This in turn suggests that the SONAR/HOPS server must be holding all of the information required to calculate and generate a reply to a SONAR/HOPS client.

A SONAR/HOPS service should be easily deployable, and should thus not require overly expensive computer or communications equipment. This leads to the goal that a typical server system (NT server, Sun workstation) should be able to obtain and store in memory all the information provided by IDMaps, without undue strain on its resources.

Total Cost of Distance Calculation Continuing the thread, the cost of calculating the distance for any pair of Internet hosts from the IDMaps information should also not place undue strain on a typical server system.

Scope of Distance Information We assume that the distance information applies only to the "public" portion of the Internet—that is, the backbone networks, information derived from BGP, and possibly the public side of firewalls or border routers of private networks. This restriction derives from the fact that private network distance information is not obtainable. Even if the information were obtainable, however, it may still be desirable not to include it for scalability reasons.

This is not to suggest that distance information inside private networks is not important. Here we assume that the architecture presented in this paper can be replicated within a private internet, but otherwise do not address the issue.

Distance Information End-points A goal of IDMaps is that the distance information it provides allows an estimate of the distance between any two valid IP addresses in the Internet. It is important to discuss this particular goal because it significantly increases the complexity of the IDMaps service.

The alternative to this goal would be a service whereby distance information provided by a given server can only provide distance information about pairs of hosts where one of them is in the proximity of that server. Such a service is much simpler because each system only has to contend with O(N) information, where N is the number of possible destinations, rather than IDMap's O(N^2) information—all possible pairs of Internet destinations.

The O(N) service is useful to those systems near the server, which is often the case. Indeed, this is the service proposed by SONAR, based on its simplicity and perceived usefulness.

In defense of the more ambitious IDMaps goal, we point out its advantages in the common application of selecting one of multiple equal-content web servers. Typical web clients today do not have the ability to make such a selection. It must be done by a proxy on behalf of, and transparently to, the client. This proxy either captures the DNS request of the client and returns the IP address of the selected server, or it captures the HTTP GET of the client and forwards the client to the identical page on the selected server. The commercial Distributed Director product of Cisco operates this way.

Either way, the proxy is generally at a different location from both the client and the servers, and so could not use the O(N) service.

In addition, good web server selection requires knowledge of server load in addition to Internet distance. This information can more efficiently be maintained in the proxy system, for instance by its periodically querying the servers.

While we assume that other applications exist or will exist that may take advantage of the ability to learn the distance between a pair of remote hosts, we believe that the one mentioned here taken alone is sufficient justification for this goal.

Measurement of Distance Information The distance information in IDMaps is produced by systems we call Tracers. By necessity, Tracers must be deployed at many different Internet locations. This is because distance information relating to a particular location in the Internet can only be measured from that point.

In order to encourage the largest possible installation of Tracers, it is necessary that the operation of a Tracer in the context of IDMaps be not substantially more costly, in terms of both human and equipment resources, than the operation of a standalone Internet measurement tool. The idea here is that sites that might normally establish an Internet measurement tool for their own use would be willing to allow that tool to also participate in the overall IDMaps infrastructure.

To achieve this basic goal, Tracers must be fully self-configuring, and must not require large amounts of information to operate. In particular, they themselves must not require the distance information generated by other Tracers for their correct operation.

3 Basic Architecture

This section outlines the architecture for the IDMaps service. We address the basic questions set forth in the Introduction section. Ultimately, we produced two distinct variations of the architecture: Hop-by-hop (HbH) and End-to-End (E2E). While these two approaches differ in the type of distance information used to make estimates, they both share a significant number of basic concepts.

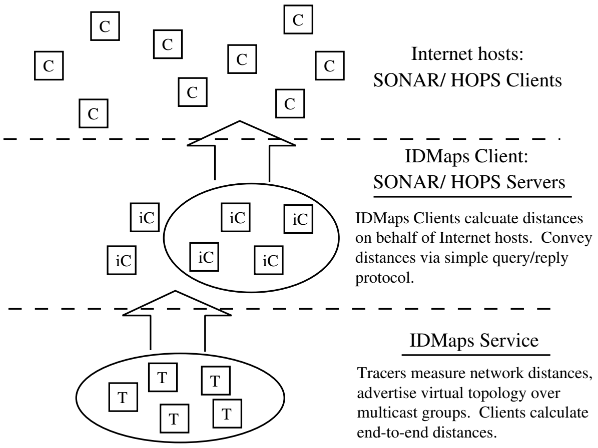


Figure 1: Basic Model: Two Tiers of Functionality

We start by describing these shared concepts. We then describe the two different approaches, and then discuss additional aspects of the architecture, highlighting the differences between the two approaches. The purpose here is to lay down basic concepts and issues rather than provide a detailed design, which remains for future work.

3.1 Separation of Tracers and Clients

Figure 1 illustrates a basic model whereby Tracers collect or measure and advertise Internet distance information. Collectively, this distance information is called the distance map, and consists of the collected or measured raw distances (often simply called distances). Systems called Clients obtain the distance map and use it to estimate distances between Internet hosts. SONAR and HOPS servers are examples of potential IDMaps Clients.

The full separation of the Tracer and Client functions is a useful modularity, as it allows for different applications to obtain and use the distance maps. More to the point, however, the full separation is necessary because the differences in functionality places incompatible constraints on the two types of systems. Client systems should be deployed in such a way that they are easily accessible by their clients, and that they are easily managed by their operators. These systems generally require large memories, and depending on how many clients they serve may require high-bandwidth connectivity.

Tracers, on the other hand, must be placed where they are able to obtain raw distance information. As described later on, this placement is largely dictated by the Internet topology and by the desired accuracy of the IDMaps service. Furthermore, Tracers may be relatively small machines, with many of them contributing only a small number of raw distances to the distance map. Therefore, both the types of machines used as Tracers and Clients, and their locations in the Internet, are different.

3.2 Simple Forms of Distance Information

The simplest (and most accurate) form the distance information provided by IDMaps could take is a list of the distances from every globally reachable IP address[1] to every other. The distance from one IP address to another is then determined by simply indexing the list to the appropriate entry (say, using a hashing algorithm), and reading the number. The sheer scale of this information (H^2, where H is millions of hosts) makes this simple approach infeasible, as does the intractable problem of even just finding all such hosts in an ever-changing Internet in the first place.

The next simplest would be a list of the distances from every globally reachable Address Prefix (AP) in the Internet to every other (Figure 2). Here an AP is defined as a block of IP addresses that is aggregatable at a single ISP (Internet Service Provider) backbone router. Note that such a block may be a sub-block of a CIDR (Classless Inter Domain Routing) block assigned to a given ISP. Determining the distance from one IP address to another is only slightly more complicated—each IP address is first mapped into its AP, and the AP is then indexed in the list.

This approach is less accurate than the first approach, because it does not take into consideration the location of the specific IP address within the AP. This can be a problem for APs that are geographically disperse (say, an AP belonging to a global corporation) and have multiple Internet access points. Nevertheless, we expect the approximation to provide adequate accuracy for most addresses and most applications.

Unlike determining the global set of IP addresses, determining the set of APs, while non-trivial, seems feasible (see Section 3.8). The scale of the information, however, is still prohibitive. There are some 50,000 and growing assigned CIDR blocks [14], and probably several times that many distinct APs. Probing, disseminating, and storing the full list of P^2 pairs of AP-AP distances (easily a terabyte, given 200,000 APs and 25 bytes per list entry) is equally out of the question.

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 [15]. 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. The resulting accuracy of the estimated distances, however, is highly suspect. Many ASs are global in scope, and multiple ASs 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) but belong to different ASs, while other IP hosts are very far apart but belong to the same AS.

Another approach is to still use some clustering of APs, but to use some unit of clustering other than the AS. A natural design choice is to select certain systems, let's call them boxes, distributed around the Internet, so that every AP is relatively close to one or more boxes. The distances between these boxes are listed. Also listed are the distances between the APs and their nearest box(es). The distance between any two APs can then be calculated as the sum of the distances from the APs to their nearest boxes, and the distance between the boxes. The resulting accuracy is dependent on how close each AP is to a box. Assuming that we can manipulate the number and location of boxes, we have a tuning knob for increasing accuracy at the expense of more listed raw distances.

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

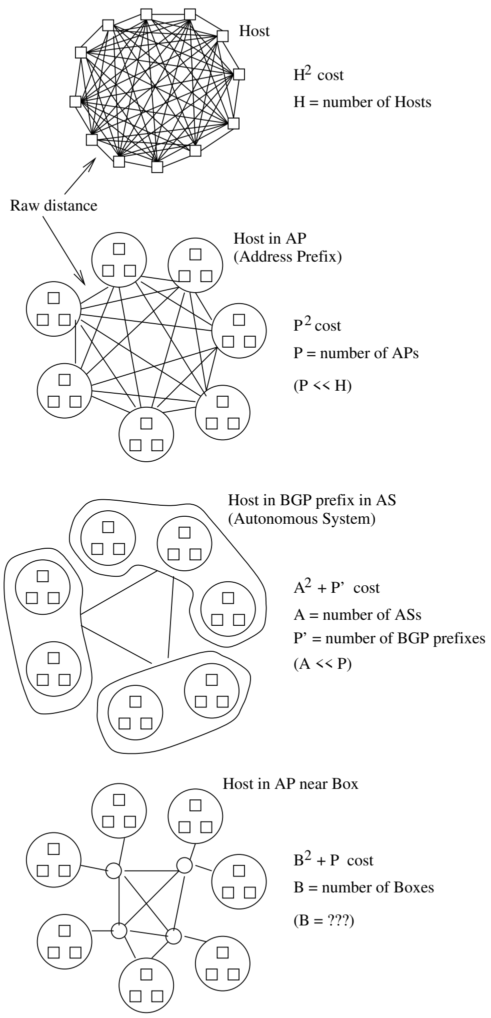


Figure 2: Various Forms of Distance Information

This approach scales as B^2+P, where B is the number of boxes. Assuming that P, the number of APs, is a manageable number (no more than several hundred thousand), the question then becomes, how big is B? If B is on the order of 10,000, then the size of the list is again too large. If on the other hand B is on the order of 500, then the N^2 component is roughly the same as the P component and, at least in terms of simple storage and lookup, manageable.

It is impossible to predict with precision the number of boxes needed, but we can make a rough guess as follows. The 1992 Times Atlas of the World sees fit to list the populations of roughly 300 metropolitan areas (most with population over 1,000,000). It is easy to imagine that each of these areas could benefit from having one or more boxes,

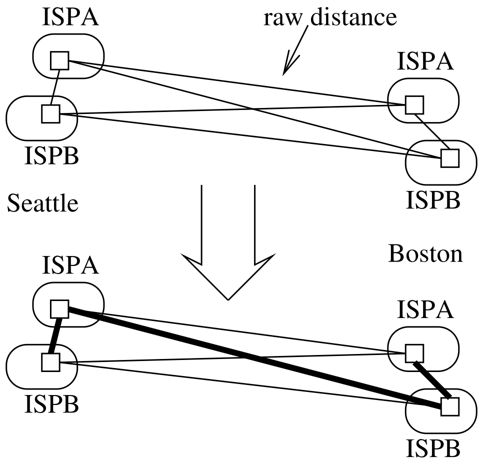


Figure 3: Redundant Distances: "Long-leg/Short-leg" Effect

and further that there could be one or more boxes per major ISP in the area. This would allow, for instance, distributed game applications to find game partners in the same metropolitan area and ISP, keeping latency at a minimum. Assuming 5 ISPs per area, 500 areas globally, and 2 boxes per ISP, this translates into 5,000 boxes globally, or over 25 million box-box distances.

While it may very well be that fewer boxes than this is required, for a given level of accuracy, it may equally well be that more are required. It therefore seems somewhat risky to base a design on B^2 box-box distances.

3.3 The Distance Map

However, it is not necessary to list all B^2 box-box distances to achieve good accuracy. For instance, assuming multiple boxes in both Seattle and Washington DC (for instance, one per ISP), it would almost certainly not be very useful to know all of the the distances between them.[2] Knowing only one of them would allow a sufficient distance approximation between hosts in Seattle and hosts in Washington DC (Figure 3).

Likewise, it is also unnecessary to know both the distance from a box in Seattle to a box in Wash-

[2]We understand here that geographical distance does not necessarily directly relate to network distance (though often it does), for instance because of odd routings. We use geographical locations here to simplify the discussion.

ington DC and the distance from the box in Seattle to a box in nearby Baltimore. If only the distance from, say, Seattle to Washington DC is known, the Seattle to Baltimore distance can be estimated as roughly the same.

Both of these examples say in essence that a given unknown distance can be estimated by concatenating a set of known distances together, provided that all but one of the known distances are short (relative to the single long distance). We call this the "long-leg/short-leg" effect. Let (x, y) denote the distance from x to y. 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, then the bound is tight and the estimate accurate. Deriving a distance estimate from this bound has been referred to as "triangulation" [16, 17].

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, then we are in fact making an assumption of 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 significant complex topology completely hidden from network-layer-only viewpoints such as available to IDMaps. If violated, it can render the triangle inequality incorrect: (a, c) might be much higher than $(a, b) + (b, c)$ or much lower than $|(a, b) - (b, c)|$.

The Internet certainly contains exceptions to the assumption of efficient routing. Just how prevalent and how serious the resulting IDMaps inaccuracies remain as key questions, but difficult to assess without first building a widely deployed measurement infrastructure such as IDMaps itself.

Given efficient routing, there is an additional way that raw distances between boxes can be reduced. It may be unnecessary to know the distances from Seattle to Chicago, Chicago to Washington DC, and Seattle to Washington DC. If Chicago is pretty much on the path from Seattle to Washington DC, then it is not necessary to know the Seattle to Washington DC distance. It can be estimated as the concatenation of the Seattle to Chicago and Chicago to Washington DC distances. We call this the "on the path" effect (Figure 4).

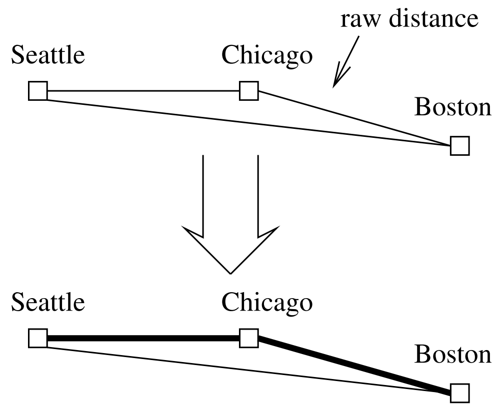


Figure 4: Redundant Distance: "On the Path" Effect

The question then becomes, how can we know if a given intermediate box b is really "on the path" between a and c? Or, how do we know that b is Chicago and not Tokyo? As it turns out, we don't really have to know that. As long as some intermediate box is on the path, then by simply running a shortest-path spanning tree over the known topology of distances, the right path is teased out (again, assuming efficient routing). In terms of our example, say we know all the distances from Seattle to Chicago, Denver, Dallas, Mexico City, Panama City, and Buenos Aires, and the distances from those cities to Washington DC. If we run a spanning tree algorithm over that topology of distances, the path through Chicago will be the shortest, and will serve as our estimate. This is illustrated in Figure 5.

The key question then is, how do we know whether any of the intermediate locations are "on the path". If out of the previous list of cities the distances to Chicago and Denver are not known, then the shortest path is through Dallas, and the resulting estimate is not very good.

The problem can be stated as follows. Assuming a number of boxes placed at various locations around the Internet, how can we determine which box-box distances are useful and which are redundant? In other words, how do we decide what raw distances to include in the distance map? In Sec-

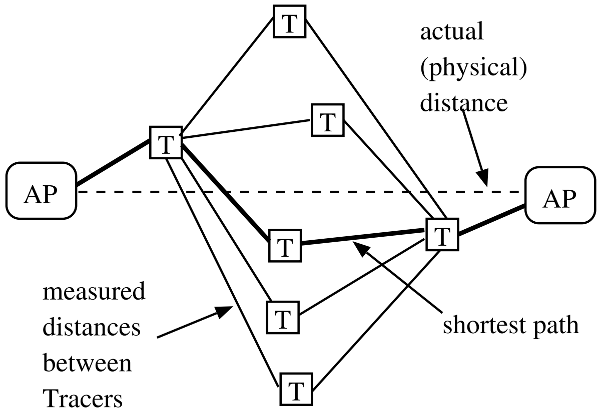


Figure 5: Shortest Path Triangle.

tions 3.5 and 3.6, we describe two approaches to this specific problem, the Hop-by-Hop (HbH) and End-to-End (E2E) approaches.

3.4 Distance Computation

From the point-of-view of the Client, the distance maps for either the HbH or E2E schemes have certain common properties. Both contain APs, boxes, AP-box distances, and box-box distances. Each AP has only one or a few AP-box distances associated with it. In particular, the distances to its closest AP(s). Boxes may have a good deal more distances associated with them, but intuitively we envision a maximum of 30 or 40 distances for a better-than-average connected box. This is nowhere near the hundreds or thousands of potential distances that would exist with B^2 distance connectivity.

To estimate the distance from one IP address to another, the Client takes the following steps:

1. Determine which AP(s) each address is in. Since each AP is expressed as an address prefix, this is a straight-forward lookup process.

2. Determine which box the APs are connected to. This is also a simple table lookup.

3. Run a spanning-tree algorithm over the topology of boxes to find the shortest distance from the boxes of the source APs to the boxes of the destination APs. This shortest distance is the estimated distance.

The exact operation of the spanning-tree calculation in step 3 varies between the HbH and E2E models (and between possible variations of the E2E model). These differences are described in the following sections.

An important point regarding this calculation is that all or nearly all of the box-box topology (that is, the "pruned" box-box topology, not the N^2 topology) must be stored to generate an accurate distance estimate. The same is not true for the AP-box connectivity. Strictly speaking, only the connectivity of the source and destination APs are needed—in order to determine the source and destination boxes. This has important ramifications on the methods for disseminating distance map information (Section 3.7).

3.5 The HbH Model

In the HbH model, the Tracers probe all transit backbone routers, and model every transit backbone router as a box, and every physical link between them as a box-box distance. In other words, the box-box topology is nothing more than the actual Internet transit backbone router topology. Likewise, actual Internet access links from subscriber networks to their ISP transit backbone routers (perhaps through one or more non-transit subscriber access routers) are modeled as AP-box distances.

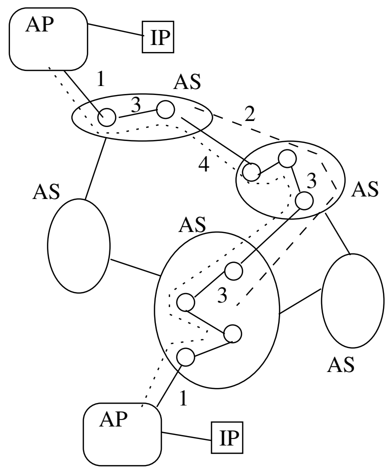
This model has a certain intuitive appeal in that the spanning tree algorithm run over the box-box topology is an approximation of actual Internet routing algorithms, which themselves are spanning-tree algorithms. Assuming efficient routing, the real paths used in the Internet will be close to those calculated by the spanning-tree operation.

Because the actual Internet routing algorithms have two levels, inter-AS routing (BGP) and intra-AS routing, the box-box topology, as well as its spanning-tree algorithm, reflects this. In other words, the boxes in the distance map have to be labeled as belonging to a given AS.

Figure 6 simplistically illustrates how the spanning-tree algorithm (step 3 above) is executed:

1. Determine which Autonomous System (AS) each AP connects to.

2. Running a spanning tree algorithm over the



1. Autonomous Systems (ASs) of APs
2. AS Path
3. Router path in each AS
4. Concatenate router paths

Figure 6: Spanning Tree Algorithm for HbH Model

inter-AS topology, calculate the AS path from one AS to the other.

3. For each AS in the AS path, starting with the source AS, calculate the shortest router path from the entry router (box) to the next AS (intra-AS topology).

4. Concatenate the calculated router paths to obtain the complete path.

The cost of the distance map in this model is roughly $P + R + L$, where P is the number of APs, R is the number of transit backbone routers, and L the links between them. Storing the entire backbone router topology may at first glance seem like an excessive amount of information. The size of the router topology information, however, is on the order of, if not well less than, that of the AP connectivity information ($R + L < P$). This is because there is substantial fan-out from a transit backbone router to multiple non-transit subscriber access routers to still more distinct APs. Since information about the connectivity of each distinct AP (P) must be stored in any event (for either model), keeping the router topology does not make matters significantly worse as far as memory requirements go.

Since the immediate up-down status of routers and links does not have to be monitored (as is required in routing algorithms to prevent loops and black holes), the bandwidth required to transmit router topology information is proportional to the size of the router topology (and, again, therefore less than that required to transmit the AP connectivity information).

Finally, by taking advantage of the hierarchical structure of the Internet (Autonomous System-level/router-level), the spanning tree calculation is efficient because it is broken up into small pieces rather than run over the whole Internet backbone topology.

3.6 The E2E Model

In the E2E model, the Tracers themselves are the boxes, the box-box distances are actual distance measurements made between Tracers,[3] and the AP-box distances are actual distance measurements made from Tracers to APs.

The spanning tree algorithm is executed just as described in Section 3.4 above. No additional mechanisms, such as with the HbH model, are required.

Where the HbH model is appealing for its tight reflection of the actual Internet topology, the E2E model is appealing for its abstract simplicity. It does not directly solve, however, the problem of knowing which box-box distances are useful and which are redundant.

While we have not worked out specifics, we believe a good basic approach to this problem is to simply try out new box-box distances, determine how useful they are, and keep only those that prove themselves to be useful. Tracers would do this independently of the activity of other Tracers, therefore keeping Tracer operation as simple as possible.

For example, each Tracer $T1$ could periodically select another Tracer $T2$ and measure the distance to $T2$. Since Tracers already know distances to nearby APs, Tracer $T1$ could query Tracer $T2$

[3] The actual distance 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 Tracer's router to Tracer's router distance simply as the distance between two Tracers.

for a list of $T2$'s nearby APs and the corresponding distances. Given only these raw distances, Tracer $T1$ can calculate the distance from one of its own nearby APs to one of $T2$'s nearby APs.

Tracer $T1$ can then formulate a query to a Client for the estimated distance between an IP address in its AP and one in $T2$'s AP. If the total distance reported by the Client is close to that known by $T1$, then clearly the distance $(T1, T2)$ is not a useful one. The box-box topology already stored by the Client contains a path of distances approximating distance $(T1, T2)$. If on the other hand the reported total distance is substantially greater than $(T1, T2)$, then distance $(T1, T2)$ is clearly a useful addition to the box-box topology. In this latter case, Tracer $T1$ could start to advertise the new distance. In this way, new useful raw distances would be discovered.

At the same time, Clients would monitor their topologies to discover redundant distances. A simple and efficient way they could do this is to keep tabs of the number of times each raw distance is used as part of the shortest path used for a given distance estimate. Little-used distances could then be tested by running a spanning-tree between their end-point boxes, but without including the distance being tested. When a given raw distance proved to be of not much use, the Tracer advertising it could be directly informed by the Client.

Clearly this "best-effort" approach would not achieve the optimal topology for a given number of box-box distances. In particular, while this approach can tell the usefulness of a new distance as an addition to a given topology, it cannot tell when a new distance is beneficial as a replacement to one or more already-established distances. Determining the latter, however, appears to be computationally expensive. Our hope is that determining the latter, or otherwise finding complex ways to generate near-optimal topologies, is unnecessary—rather that the simple approach outlined above produces "good enough" topologies.

3.7 Disseminating the Distance Map

The Tracers, whether or not they themselves are the boxes in the distance map, are the ultimate source of the distance map, and must convey the distance map to the Clients. The characteristics of this distribution can be summarized as follows:

1. There are a large number of both producers (Tracers) and consumers (Clients) of the information.
2. Each Tracer produces a small portion of the distance map (several to several hundreds of raw distances).
3. Each raw distance changes infrequently (from several hours to a day).
4. Generally speaking, a typical Client will store the complete box-box part of the distance map. For the E2E model, the size of this part of the distance map is unknown, but can be roughly estimated to be anywhere from 100 Kbytes to upwards of 5 Mbytes (5000 boxes with 100,000 distances and 50 bytes per distance).
5. Some Clients must store the full AP-box part of the distance map (other Clients may simple query these Clients for the specific AP connectivity of interest, as discussed later in this section). This part of the distance map may be estimated at upwards of 10 Mbytes (for 200,000 APs).
6. Since each Client independently estimates distances, the distance map does not need to be identical or perfectly up to date at all Clients.

The ideal mechanism for the dissemination of the distance map is IP multicast. It allows each Tracer to transmit its small portion of the distance map without coordination with other Tracers (though in the HbH model coordination is needed for other reasons). The Tracers never need to know about 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). In this sense, IP multicast serves as a valuable discovery mechanism as well as a distribution mechanism.

Because the distance map does not have to be error-free or synchronized among Clients, the non-guaranteed delivery characteristic of IP multicast is not a problem. IP multicast also allows the Tracers to transmit their raw distances without having to listen to those of other Tracers, which is important because a Tracer may not be a powerful machine.

The basic idea is that each Tracer periodically (with random skew in the period, to prevent any kind of inadvertent synchronization 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. Each transmitted raw distance has a time-to-live associated with it. The time-to-live is set to be several times longer than the transmit period. Clients simply delete any raw distances whose age exceeds the assigned time-to-live.

It is useful to have different multicast groups for different information. At a minimum, there are separate multicast groups for the box-box distances and the AP-box distances. This allows for a model whereby a given Client that does not need to store the full AP-box connectivity information can listen to only the box-box group(s).[4]

Because of the large volume of AP-box information, it should be distributed over multiple multicast groups according to some locality-of-reference criteria, for instance, proximity to a locus or cluster of boxes.

3.8 Discovering APs

Both models have in common the problem of discovering APs. The difficulty here is that the address blocks advertised by ISPs in BGP do not necessarily represent a single group of addresses in one 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.

[4]If a given Client is not storing all AP-box distances, then it requires some means of learning the ones it needs for a given calculation. This must be solved in the context of a given system of Clients, and is outside the scope of the IDMaps service proper. However, a typical mechanism would be for some Clients to obtain all AP-box distances, and for Clients that do not have them to query the Clients that do. The querying Clients could discover the queryable Clients through a separate multicast group over which the queryable Clients advertise themselves.

Ideally, a large number of Tracers will be installed for the express purpose of providing accurate distance information for a given site. These "dedicated" Tracers can easily be configured with site AP information, and this information can be advertised over the AP-box distance multicast group.

For APs not covered by either of the above, "general purpose" Tracers will have to discover the address boundaries of APs, as well as which APs are nearby. (These general purpose Tracers are not necessarily the light-weight systems described in the Goals section.) The basic approach will be for a few of these Tracers to obtain address blocks from BGP and advertise them over the AP-box distance multicast group. 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 the addresses are in the same location, then the block can be considered to be a single AP. Here location is determined through the IP address of the last system responding to the `traceroute` that does not have an address from the block being probed. If probes to two IP addresses produce the same such penultimate address, 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.

3.9 Discovering AP-to-Box Distances

Once (the address boundaries of) an AP is discovered, its AP-box distance must be determined. For the HbH model, the box is the AP's access backbone router. Almost any Tracer can discover this backbone router, because the routes of probes from different locations will converge on the same backbone router. Once discovered, the AP-box distance is advertised by the discovering Tracer.

The same is not true for the E2E model. In the E2E model, unless an AP is pre-configured into a dedicated Tracer, only the Tracers nearest to the AP itself can discover and subsequently advertise the AP-box distance. As a result, when a Tracer first discovers an AP, it assumes itself to be the closest Tracer and advertises its distance to the AP as the AP-box 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.

3.10 Discovering Distances in General

While at first glance the discovery of AP-box distances for the HbH model described above appears simpler than for the E2E model, in fact distance discovery in the H2H model is overall more complex.

In the E2E model, each distance can only be probed by the Tracer at the end of the distance. While, over time, this can result in a large number of probes for a Tracer, the activity is simple because it does not need to be coordinated with other Tracers. Each Tracer can independently decide what other Tracers or APs to probe, and when to probe them. Note that the E2E model requires a separate multicast group over which Tracers advertise themselves. This group is monitored by all Tracers (dedicated and general purpose alike) so that they may learn of each other. Self advertisements can be relatively infrequent (once every several hours), so this multicast group will be lightly loaded.

Probing in the HbH model is more complex, for several reasons. To see why, we must first consider the probing regime of Tracers in the HbH model. Unlike the E2E model, where Tracers know in advance every box, Tracers in the HbH model do not know in advance what the Internet topology looks like—that is, what routers and links exist. They must actively find them. This is done by executing `traceroutes` to randomly selected APs, which returns a list of routers in a path. This `traceroute` activity is a continuous background activity of Tracers, and must be done whether or not a given AP has been discovered.

Once a link has been discovered, it must individually be probed periodically. Unlike the E2E model, where each distance can only be probed by Tracers on the end of the distance, a link in the HbH model can potentially be probed by a large number of Tracers. In order not to swamp any given link with probes, the Tracers must coordinate their activity so that a given link is only being probed by one or a few Tracers at a time. This in turn requires that Tracers are aware of the activity of other Tracers, which means that Tracers must listen to the box-box distance multicast group.

This group, however, carries a substantial volume of information (the backbone topology of the Internet), so requiring that Tracers listen to it goes counter to the goal of Tracers being light-weight. To cope with this, we must partition the box-box multicast group into multiple multicast groups, for instance one per AS. After discovering a link through a `traceroute`, a Tracer must first determine the AS of the corresponding routers, and from that the multicast group the link is being advertised over.[5] The Tracer then listens to that group, and determines how many other Tracers are probing and advertising the link and how far from the link they are. If there are several Tracers advertising the link, and they are closer to the link, then the Tracer may choose not to continue to probe and advertise the link. Otherwise, the Tracer may choose to probe and advertise the link, and must coordinate this probing with that of the other active Tracers.

Finally, a complication arises from the fact that a given router may respond to TTL Expired with different addresses for different interfaces. This results in different Tracers, probing from different locations, seeing different addresses for the same router. DNS reverse lookups cannot always be used to resolve these to the same router, resulting in a single router appearing as multiple boxes in the topology.

3.11 Summary

The E2E model, at this early stage of design, appears simpler and more elegant than the HbH model. Its main drawback at this time is that we do not know if the E2E model can scalably produce sufficiently accurate estimates. In other words, we do not know if the approach outlined in Section 3.6

[5] Without getting into details, this is done through the use of yet another multicast group, which advertises AS-group mappings.

really will produce a virtual topology that adequately reflects the physical topology without requiring an excessive number of distances. (Strictly speaking we do not know this for certain with the HbH model. The fact that the HbH model reflects the actual Internet topology, however, gives us a certain confidence that it will produce good estimates.)

Ultimately, with either model, we believe the only way to test their scaling and accuracy is to build prototypes and try them in the Internet. Because of its simplicity, we would prefer to try the E2E model first. In order to get a better feel for how it might perform, we have done some preliminary simulations of a small E2E topology using previously gathered Internet measurement data. The remainder of this paper describes that simulation.

We note that if the E2E model described above proves not to scale adequately, we may be able to engineer improvements while staying within the E2E framework. For instance, we could increase the total number of Tracers by having two tiers of E2E topologies—one global and known by all Clients, and another local and known only to Clients that care about finer granularity in the locale. Such "improvements", however, add substantial complexity to the original model.

Finally we note that the two models are not necessarily mutually exclusive. There is no reason, for instance, that a box-box distance in the E2E model cannot in fact represent the distance between two directly connected routers and vice versa. Whether doing so is useful, however, can only be determined through experimentation with a working system.

4 Experimental Evaluation

In this section we present some preliminary experimental results on the feasibility of computing Internet distances based on E2E model. The basic question we want to investigate is: given a number of Tracers distributed around the Internet, how close would the distance between two points estimated by IDMaps be to the actual distance?

To answer this question, we use `traceroute` data collected with the network probe daemon (NPD) tool described in [18]. A number of sites on the Internet were recruited to run NPDs. At random intervals, these NPDs were asked to measure the route to another NPD site using `traceroute`. A full 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 [18]. 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 $D2$.

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

Data Set	Total	Shortest
$D1.1$	8382	458
$D1.2$	11563	621
$D1.3$	13729	701
$D1.4$	13887	761
$D1.5$	20004	815
$D1.6$	26334	962
$D2.1$	85011	1978
$D2.2$	98642	2213
$D2.3$	57798	1559

Each source s that participated in an NPD experiment did a number of `traceroutes` to each destination d participating in the experiment. For each data set, we estimate the latency between s and 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 9 data sets. The first 6 data sets are from experiment $D1$ the remaining 3 are from experiment $D2$.

These triangles simulate IDMaps' estimation of a path between two Internet locations, i.e. if ac is the actual path between a and c, IDMaps, having

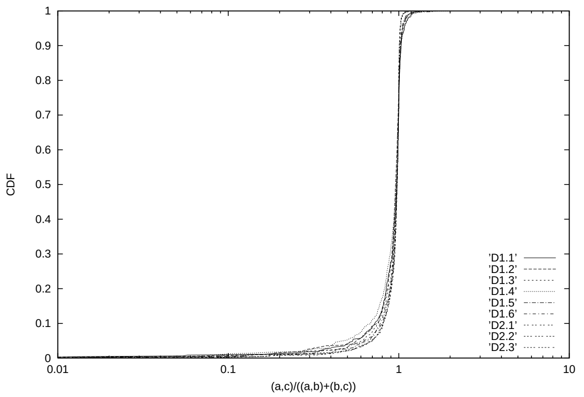


Figure 7: Cumulative Distribution Function (CDF) of the ratio of $(a, c)/((a, b) + (b, c))$ for shortest-path triangles.

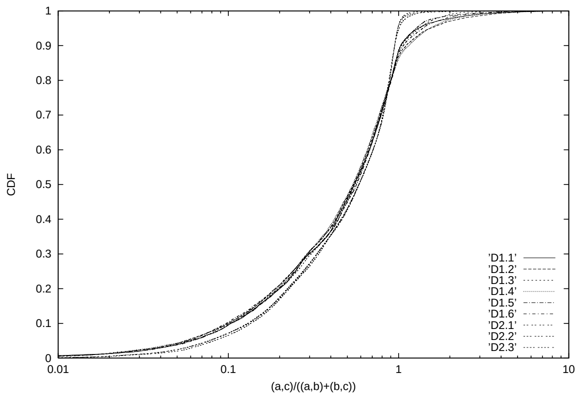


Figure 8: Cumulative Distribution Function (CDF) of the ratio of $(a, c)/((a, b) + (b, c))$ for all triangles.

only the ab and bc distances, estimates ac's distance as $(a, b) + (b, c)$. Hosts a and c represent the two Internet addresses (or their APs) whose distance we want to know. All other hosts b represent Tracers in the distance map through which the calculated path may go. Our question is, even with the relatively small number of Tracers represented, can we achieve a reasonably good estimate from the calculated shortest path?

An answer of "yes" tells us that a relatively small number of Tracers may potentially serve effectively as intermediate nodes in a shortest path distance estimate. This in turns means that the large majority of Tracer-to-Tracer distances do not need to be included in a distance map. We hasten to emphasize, however, that the experiment here is preliminary, and that experimentation over a working IDMaps system is required.

4.1 Triangulation Error

We now look at the triangulation error for all of the triangles formed from our NPD experiments. The metric we are studying is the ratio of the length of the ac leg of each triangle over the sum of the length of its two other sides $(a, b) + (b, c)$. The closer this ratio is to 1, the smaller the triangulation error, and the more accurate the distance estimate obtainable from IDMaps.

We have speculated in Section 3.3 that when the number of Tracers involved in IDMaps is large, the shortest path computed between two APs would better approximate the actual distance. By shortest-path here we mean that given all the bs that can potentially be used to estimate the distance ac as $(a, b) + (b, c)$, the shortest-path is the path that involves the b that provides the smallest $(a, b) + (b, c)$ (see Figure 5). Column 3 of Table 1 lists the number of shortest-path triangles computed from each data set.

Figure 7 shows that the ratio of the actual (a, c) distance over the sum of length of the other two legs are quite close to 1. The lines in the figure are practically on top of each other; we do not see the need to differentiate them. The figure shows that less than 10% of all shortest-path triangles formed have their ac leg shorter than half of the sum of the other two legs.

For comparative purposes, we show in Figure 8 the cumulative distribution function of the triangulation error from triangles involving all potential bs, not just those providing the shortest distance estimate. The figure shows, for example, that the actual distance between a and c in about 40% of the triangles formed is shorter than half that of the sum of the length of the ab and bc legs, which provides initial confirmation to our speculation that with more Tracers distributed around the Internet, the shortest-

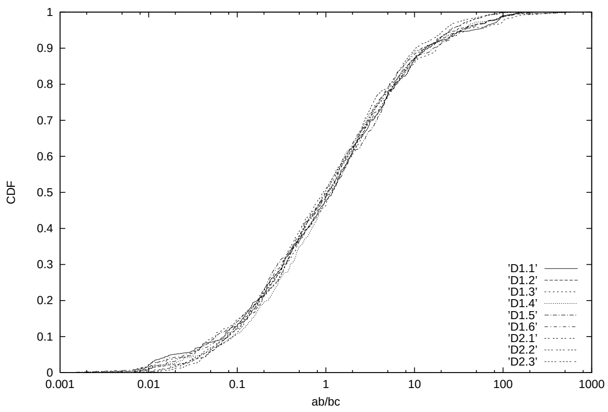


Figure 9: Ratio of the length of the ab leg over that of the bc leg.

path estimates will more closely approximate the actual distances.

Figure 9 shows the ratio of the length of the ab leg of all shortest-path triangles over the length of their corresponding bc leg. The wide range of ratios present in the graph means that both the long-leg/short-leg effect (Seattle–Washington DC–Baltimore) and the "on the path" effect (Seattle–Chicago–Washington DC) are coming into play.

In summary, while we do not make any claim as to the potential accuracy of IDMaps' distance estimates, results presented in this section convince us that we should continue to explore the use of the E2E model in distance estimation.

5 Issues

This section briefly discusses a few of the most important of the many issues facing this work. The primary issue, already mentioned, is simply whether or not reasonable distance estimates can be made using the E2E model presented. Even if so, any number of performance issues must still be resolved: how many Tracers are needed, where must they be placed, how often must they trace distances, how much traffic is generated, what are the storage/latency trade-offs of Clients, and so on. While careful simulations may shed some light on these questions, we believe ultimately that the only real way to answer them is to build the system and try it on a large scale.

Acknowledgments

We have benefited from interesting discussions with Sally Floyd. We thank Susan Hares, Craig Labovitz, and Dun Liu for making the $D2$ data set available to us.

References

[1] 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.

[2] S. Bhattacharjee et al., ""Application-Layer Anycasting"," Proc. of IEEE INFOCOM '97, Apr. 1997.

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

[4] 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.

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

[6] P. Francis, "Host proximity service (hops)," URL: http://www.ingrid.org/hops, Aug. 1998.

[7] M. Stemm, R. Katz, and S. Seshan, "Spand: Shared passive network performance discovery," url: http://spand.cs.berkeley.edu/.

[8] C. Huitema et al., "Project felix: Independent monitoring for network survivability," url: ftp://ftp.bellcore.com/pub/mwg/felix/, Sep. 1997.

[9] S. Keshav, R. Sharma, and R. Siamwalla, "Project octopus: Network topology discovery," url: http://www.cs.cornell.edu/cnrg/topology/Default.html, May 1998.

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

[11] V. Paxson et al., ""An Architecture for Large-Scale Internet Measurement"," IEEE Communications Magazine, To appear 1998.

[12] R. Caceres et al., "Minc multicast-based inference of network-internal characteristics," url: http://www.research.att.com/˜duffield/minc/, 1998.

[13] C. Labovitz, G.R. Malan, and F. Jahanian, ""Internet Routing Instability"," Proc. of ACM SIGCOMM '97, 1997.

[14] T. Bates, "The cidr report," url: http://www.employees.org/˜tbates/cidr-report.html, June 1998.

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

[16] 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.

[17] James D. Guyton and Michael F. Schwartz, "Locating nearby copies of replicated internet servers," in Proceedings of ACM SIGCOMM, August 1995.

[18] V. Paxson, "End-to-End Routing Behavior in the Internet," Proc. of ACM SIGCOMM '96, pp. 25–38, Aug. 1996.