Network Topology Inference Based on End-to-End Measurements

Xing Jin, *Student Member, IEEE*, W.-P. Ken Yiu, *Student Member, IEEE*, S.-H. Gary Chan, *Senior Member, IEEE*, and Yajun Wang

***Abstract*—We consider using traceroute-like end-to-end measurement to infer the underlay topology for a group of hosts. One major issue is the measurement cost. Given N hosts in an asymmetric network without anonymous routers, traditionally full $N(N-1)$ traceroutes are needed to determine the underlay topology. We investigate how to efficiently infer an underlay topology with low measurement cost, and propose a heuristic called Max-Delta. In the heuristic, a server selects appropriate host-pairs to measure in each iteration so as to reveal the most undiscovered information on the underlay.**

We further observe that the presence of anonymous routers significantly distorts and inflates the inferred topology. Previous research has shown that obtaining both exact and approximate topology in the presence of anonymous routers under certain consistency constraints is intractable. We hence propose fast algorithms on how to practically construct an approximate topology by relaxing some constraints. We investigate and compare two algorithms to merge anonymous routers. The first one uses Isomap to map routers into a multidimensional space and merges anonymous routers according to their interdistances. The second algorithm is based on neighbor router information, which trades off some accuracy with speed.

We evaluate our inference algorithms on Internet-like and real Internet topologies. Our results show that almost full measurement is needed to fully discover the underlay topology. However, substantial reduction in measurements can be achieved if a little accuracy, say 5%, can be compromised. Moreover, our merging algorithms in the presence of anonymous routers can efficiently infer an underlay topology with good accuracy.

***Index Terms*—Anonymous router, end-to-end measurement, measurement cost, topology inference.**

I. Introduction

WITH THE RAPID growth of the Internet, overlay networks have been increasingly used to deploy network services. Examples include application-layer multicast (ALM), peer-to-peer file sharing, and overlay path routing [1]–[3]. In order to build an efficient overlay network, the knowledge of underlay is essentially important. In fact, it has been shown that topology-aware ALM can achieve substantially low end-to-end delay, low physical link stress, and high tree bandwidth [4]–[6].

Manuscript received October 1, 2005; revised June 29, 2006. This work was supported in part by the Competitive Earmarked Research Grant HKUST6156/03E of the Research Grant Council in Hong Kong, and in part by the Area of Excellence in Information Technology of the Hong Kong University Grant Council under Grant AoE/E-01/99.

The authors are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: csvenus@cse.ust.hk; kenyiu@cse.ust.hk; gchan@cse.ust.hk; yalding@cse.ust.hk).

Digital Object Identifier 10.1109/JSAC.2006.884016

We consider inferring the underlay network topology among a group of hosts by means of end-to-end measurements. Such techniques would be very important and useful to, for examples: 1) the construction of an efficient overlay network [2], [3], [6]–[8]; 2) the design of application-layer multicast (ALM) protocols [4], [5]; and 3) tomography-based schemes which often rely on the knowledge of underlay topologies to efficiently infer the network properties [9].

To infer the underlay topology among a group of hosts through end-to-end measurements, traceroute-like tools extracting the router-level path between a pair of hosts are often used [10], [11]. However, such tools may take as long as minutes to identify a router-level path and generate many network packets. Given a group of N hosts, conducting full $O(N^2)$ measurements is hence costly and not scalable.

Furthermore, traceroute is implemented with Internet control message protocol (ICMP). In traceroute, the source sends out a series of IP datagrams with increasing time-to-live (TTL) to the destination. From the returned ICMP error messages, it obtains intermediate router information such as router name, address and round-trip time (RTT). However, some routers process ICMP messages differently.

1) It may not return ICMP error messages. Consequently, the router appears as unknown as indicated by "$*$" in traceroute results.
2) It returns ICMP error messages only when its load is light. As a result, the router appears as "$*$" in some cases, while as a normal router in others.
3) It simply discards ICMP messages. In this case, all the subsequent routers in the path appear as "$*$" in the traceroute result.

Following the notations in [12], we call such unknown routers *anonymous routers*, and others *known routers*. Measurement results containing anonymous routers substantially increase the difficulty in topology inference.

In this paper, we address the following two important issues in end-to-end topology inference.

- *Efficient inference of underlay information in the absence of anonymous routers*: We first consider the case with no anonymous routers. We study the following problem: given N hosts in an overlay network, how to infer the router-level topology with low number of path measurements? We consider incrementally measuring the topology. A basic principle is to discover as many new links and routers as possible in each traceroute. However, the difficulty is that we do not know the links and routers on a path, and hence the *new* ones until a traceroute has been conducted.
  We propose an inference heuristic called *Max-Delta*, where hosts first utilize lightweight tools such as GNP or Vivaldi

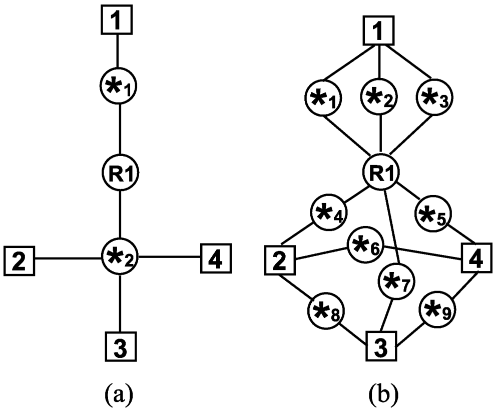


Fig. 1. An example of anonymous router inflation in the straightforwardly inferred topology. a) Actual topology. b) Inferred topology from traceroute results.

to estimate their coordinates [13], [14]. A server then collects host coordinates and chooses the best set of host-pairs for measurements in each iteration. Our simulation results show that although almost full $O(N^2)$ traceroutes are needed to discover the complete underlay topology, Max-Delta can construct a highly accurate topology with substantially fewer measurements.

- *Efficient topology inference in the presence of anonymous routers*: For each known router in a traceroute result, its RTT from the traceroute source as well as its name/IP address is available. However, for an anonymous router, only its presence in the router-level path is known. To infer the underlay topology in the presence of anonymous routers, we may consider that each occurrence of an anonymous router is a unique one. However, this leads to high inflation of anonymous routers in the resultant topology. We illustrate this in Fig. 1, where the actual underlay topology is shown in Fig. 1(a) with hosts labeled as 1, 2, 3, and 4. R_1 is a known router, while $*_1$, $*_2$ are two type-1 anonymous routers. With pairwise full traceroutes among the four hosts, the inferred topology is shown in Fig. 1(b) (assuming paths are symmetric). Clearly, even a few anonymous routers lead to a highly inflated inferred topology.

  Previous research has studied how to infer a topology by minimizing the number of anonymous routers through merging, while meeting the following consistency requirements: (a) trace preservation: the inferred topology should agree with all the traceroute paths and (b) distance preservation: the length of the shortest path between two nodes (i.e., hosts or routers) in the inferred topology should not be shorter than the traceroute result [12]. It has been shown that this problem belongs to the hardest class of NP-complete problems. Formally speaking, it is NP-hard to construct a topology which is consistent with traceroute results and the ratio of its total number of routers to that in the optimal solution is within a factor n^δ, where n is the number of routers in the traceroute results and δ is a constant. They have proposed an approximate heuristic to merge anonymous routers, while keeping the consistency constraints. However, the approach is still computationally expensive (we will show the complexity of consistency check in Section IV-A).

  In this paper, we greatly reduce the problem complexity by adopting a different approach to allow a small portion of inconsistent merging. We propose two merging algorithms. The first one uses Isomap [15] to map routers into a multidimensional space according to the inter-router distances. With the estimated router coordinates, we can compute the distance between any pair of anonymous routers and merge those close pairs. To further reduce the complexity, we propose a neighbor matching algorithm, where we merge the pair of anonymous routers which share the same neighbor(s) of known routers or hosts. This algorithm is of much lower complexity at the cost of some accuracy. Our simulation results shows that the algorithms incur low inconsistency in the inferred topology while achieving a substantial speedup. Most overlay applications are expected to be insensitive to such inference error.

Note that the Internet is not a symmetric network. The traceroute path from host A to host B may not be the reverse of the path from B to A. This does not affect our Max-Delta or router merging algorithms. For ease of exposition and illustrative purpose, however, we will in the following assume that the traceroute path from A to B is the reverse of the path from B to A.

The rest of this paper is organized as follows. In Section II, we review the related work. In Section III, we discuss how to efficiently infer an underlay topology in the absence of anonymous routers. In Section IV, we describe the anonymous router problem and propose our merging algorithms. In Section V, we present simulation results based on Internet-like and real Internet topologies. We finally conclude in Section VI.

II. Related Work

There are many ways to infer a network topology. Network tomography techniques periodically send probing traffic and exploit the performance in correlation to infer network topologies [16], [17]. However, because the network properties measured (e.g., loss rate or delay) are often unstable and inaccurate, it is difficult to infer an accurate topology. Border gateway protocol (BGP) routing tables can provide AS-level information, but they usually are not available to normal hosts in the Internet [18], [19]. We hence adopt traceroute, which can obtain explicit router-level information by end hosts. Traceroute-like tools have been widely used in Internet measurements such as Skitter *et al.* [20]–[22]. Skitter sends traceroute packets from different locations worldwide to actively measure the Internet topology. Mercator utilizes a modified version of traceroute to reduce probing time. Rocketfuel combines information from BGP tables, traceroutes, and DNS to infer Internet service provider (ISP) topologies. All these works focus on Internet- or ISP-level topology inference and the major concern is how to discover a *complete* network topology including all the routers and links. However, in our study, we are only interested in the topology among a certain group of hosts that are arbitrarily distributed in the Internet. Furthermore, we only need a highly accurate topology, because most overlay applications are tolerant to small distortion of the underlay topology. The key problems are hence how to reduce the measurement cost and eliminate the measurement noises.

Donnet *et al.* note that large-scale traceroute measurement such as Skitter has high redundancy [23]. That is, the traceroutes from the same monitor (or traceroute source) towards multiple destinations often overlap, and the traceroutes from multiple sources to the same destinations also have overlap. To reduce such redundancies, they propose a Doubletree algorithm. Given a monitor and a destination, the traceroute starts at some intermediate point between them. The probing then proceeds towards the destination and backwards towards the source with different stopping rules. In either case, the probing stops whenever an already discovered router is met. Our work reduces the measurement redundancy in another way. Given a group of hosts, each host is a monitor and all the others are its destinations. We note that a host cannot or does not need to traceroute all its destinations (because of the requirement on measurement cost or accuracy). We then design an algorithm for destination selection to allow hosts to efficiently discover the routers and links with a small number of traceroutes. On the other hand, it is possible to integrate the Doubletree algorithm into our work. Namely, after destination selection, a traceroute can start and stop under Doubletree's supervision. The measurement redundancy and cost can hence be further reduced. However, a potential problem in the integration is the sharing of the global stopping set. The global stopping set contains the discovered routers near the destinations, which is used as one of the stopping rules in Doubletree and should be shared among all the monitors. In the Doubletree study, the number of monitors is small (e.g., 24 monitors in their experiments) and the communication overhead for sharing is negligible. But in our study, the global stopping set should be shared among all the hosts (e.g., hundreds of hosts), which may lead to huge communication overhead.

Barford *et al.* study the relationship between the measurement accuracy and the number of traceroute sources in topology inference [24]. Given a list of destinations, they find that the marginal utility of adding additional traceroute sources declines rapidly after the second or third one. In other words, the first two or three sources can discover the majority of the complete topology by tracerouting the destinations. Here, the complete topology is obtained by combining the traceroute results from all the sources to the given destinations. In their experiments, the number of sources is much less than the number of destinations (e.g., 8 sources and 1277 destinations, or 12 sources and 313,709 destinations). But in our study, each host is a source as well as others' destination. The number of sources is equal to the number of destinations. Therefore, their conclusion may not hold in our case. Furthermore, in order to reduce the measurement time, we require hosts to conduct traceroutes in parallel. We do not use two or three hosts as sources to traceroute all the others, even though this can discover the majority of the underlay. Because this takes a long measurement time and puts uneven measurement loads on hosts.

In traceroute measurement, the presence of anonymous routers is inevitable. Broido *et al.* report that nearly 1/3 of probed paths contain anonymous, private or invalid routers [25]. They propose two methods to address it. One is to add *arcs* which bypass anonymous routers and connect the known routers. This method, despite of its simplicity, hides much topology information. Another method introduces *placeholders*, where each anonymous router is represented by a unique name. To reduce router inflation, a chain of anonymous routers between two known routers is discarded from the inferred topology if there exists another shorter path between the two known routers. However, the resultant topology still suffers huge router inflation. In our study, we consider how to reduce router inflation and quickly construct an approximate topology. Our simulation results show that our merging algorithms can significantly reduce router inflation, and the inferred topology is highly accurate.

Anonymous router problem is formally studied in [12]. Yao *et al.* study how to infer a topology given some traceroute results, where the inferred topology should be consistent with the traceroute results, while containing the minimum number of anonymous routers. They show that producing either an exact or an approximate solution is NP-hard. They then propose a heuristic which in each step chooses an anonymous router u and merges all the anonymous routers that are *mergeable* to u (two anonymous routers are mergeable if the distance and trace preservations are kept after merging). However, as we will show later, it is of high computational complexity to check whether two anonymous routers are mergeable. This heuristic is, hence, time-consuming. In fact, their simulations were only based on small-size topologies (the largest one contains 50 routers and 125 links). In this paper, we study faster approaches with much lower computational complexity. A major difference between our formulations and [12] is that there are two types of nodes in our topologies (i.e., routers and hosts) and we are only interested in end-to-end path consistency, while [12] is for more general topologies with only routers.

III. Topology Inference in the Absence of Anonymous Routers

In this section, we present heuristics on how to efficiently infer a network topology in the absence of anonymous routers. We first study in Section III-A how to infer the topology among a group of given hosts, and then address the issue of group dynamics in Section III-B.

A. Inference Based on N Hosts

Traceroute-like tools often consume much time and network resources. Hosts should then conduct traceroutes efficiently so that each traceroute reveals the most undiscovered information on the underlay. We assume that the path between a pair of hosts is unique and stable, namely, a path does not fluctuate in the measurement session. This is reasonable because previous study has shown that end-to-end Internet paths tend to be stable for significant lengths of time, such as a day [26]. As mentioned, we consider that paths are symmetric in our discussion.

The problem is how to infer the router-level topology among a given group of N hosts with low number of traceroutes. As the Internet is a relatively sparse graph (i.e., the number of links is on the same order of the number of routers), we expect that full $N(N-1)/2$ traceroutes are not necessary [27]. Fig. 2 shows an example, where 1, 2, 3, and 4 are hosts and A, B, C, and D are routers. The dashed lines indicate the overlay paths among hosts. As shown, paths 1-2, 2-3, and 4-1 reveal all the underlay

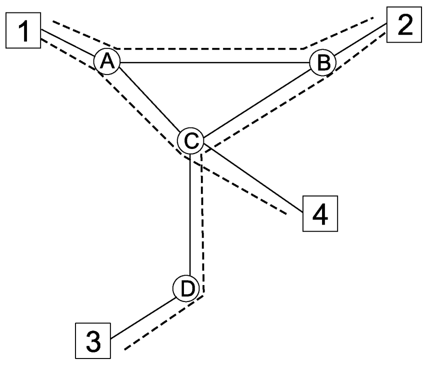


Fig. 2. An example of an underlay network. The dashed lines indicate the overlay paths among hosts.

links and routers, and therefore we only need to measure three, instead of all the possible six paths.

If we do not need to know the underlay topology with full accuracy (discovering the links with, for example, 95% accuracy), the number of measurements per host can be further reduced. The issue is then how to choose the most representative paths to traceroute so as to reveal as much undiscovered information on the underlay as possible. However, the difficulty is that we are not aware of the links or routers on a path until we have conducted traceroute on it. We hence turn to a simpler metric characterizing a path, namely, the path distance. The distance between two hosts on the discovered topology can be computed by the shortest-path algorithm, while the distance between them in the real network can be approximately estimated by a coordinate system. If the gap between the two values is large, it is with high probability that some links between the hosts (leading to a shorter path) have not been discovered.

We consider that each host reports to a central server its network coordinates, which can be estimated by low-cost tools such as GNP or Vivaldi [13], [14]. The server uses host coordinates to select a target to traceroute for each host in each round/iteration. In order to maximize the parallelism of traceroutes among the hosts in each iteration (and hence reduce the time to infer the topology), the server assigns one traceroute target to each host if possible. The hosts then traceroute their own targets and report the results to the server. The server combines all these results and based on that, starts the next iteration on target assignment. Such process is repeated until the server achieves a certain inference accuracy. Note that in an iteration, it is possible that a host is not assigned any target to traceroute. This is the case when the paths from the host to all the other hosts have been measured.

We study the following target selection mechanisms at the server.

- *Random Probing (Random)*: Given a host, the server assigns to it a target which is randomly chosen from all its unmeasured hosts. This is the simplest approach.
- *Longest Path Probing (Longest)*: Usually a longer path in the network contains more hops (or routers); therefore, among all the unmeasured paths, a longer one may contain more undiscovered links/routers. In Longest Path Probing, the server in each iteration chooses the farthest (in the coordinate space) unmeasured host from a given host as the target.
  A concern of this mechanism is that if a host is far from many others in the group (i.e., an outlier), this host would be susceptible to many traceroutes from the others at the beginning. These traceroutes likely share much common underlay information towards the outlier, thereby defeating the purpose of revealing more undiscovered information. Furthermore, due to the exhaustive traceroutes from the other hosts, there may not be any target left for the outlier to measure in the later iterations. This leads to a reduction in parallelism, and hence inefficiency in measurements.
- *Max-Delta Probing (Max-Delta)*: Denote the distance between two hosts a and b in the coordinate space as $Euclidean(a, b)$, and the length of the shortest path between them on the discovered topology as $D_p(a, b)$. Let

$$\Delta(a, b) = D_p(a, b) - Euclidean(a, b).$$

  For each host, the server computes Δ for all its unmeasured pairs and chooses the one with the maximal Δ as its target. In the first iteration, each host is assigned a target so that all the traceroute paths form a connected graph (one way to achieve it is to form a ring where host 1 traceroutes host 2, host 2 traceroutes host 3, and so on).

B. Group Dynamics

So far, our discussion is in the context of N given hosts. In fact, our algorithms can be straightforwardly extended to handle group dynamics where hosts may join or leave at any time. Clearly, the issue of host leaving is trivial and we only need to consider host joining.

For Random Probing and Longest Path Probing, when a new host joins the group, it contacts the server, which, based on the current group members, chooses targets for the host to traceroute. For Max-Delta Probing, the new host contacts the server which first randomly selects a host as its target. In the subsequent iterations, the server computes Δ for all the unmeasured pairs of the new host and selects the one with the maximal Δ as the target.

IV. Inference With Anonymous Routers

In this section, we study topology inference in the presence of anonymous routers. In Section IV-A, we show that traditional approaches that keep the distance and trace consistencies have high computational complexity and are not practical for even a medium-sized network with hundreds of routers. To reduce the complexity, we relax these constraints and propose two fast algorithms in Sections IV-B and IV-C, respectively.

A. The Complexity With Anonymous Routers

As anonymous routers greatly inflate the topology, we consider the following problem: given a group of N hosts and a set of traceroute results among them, how to construct an inferred topology by reducing the number of anonymous routers?

As we have described in Section I, there exist three types of anonymous routers. We deal with them separately. First of all, we use the *arc* method in [25] to deal with routers which discard ICMP messages (i.e., type-3). Suppose a traceroute path from host A to host B contains a type-3 router $*_x$. All the routers following $*_x$ in the traceroute path must be "$*$." Denote the router directly before $*_x$ as X. We first check whether traceroute from B to A has been conducted. If not, we add an arc to directly connect X and B. Otherwise, the traceroute path from B to A must also contain a type-3 router $*_y$. Similarly, denote the router directly before $*_y$ as Y. We add an arc to connect X and Y.

After introducing the arcs, the anonymous routers in the resultant topology are of either type-1 or type-2. We will work on this topology in the following. If we assume that each of the remaining anonymous routers is a unique one, we will get a topology which is consistent with traceroutes but suffers high inflation of routers and links. We hence need to merge the anonymous routers.

To keep the distance and trace consistencies as in [12], we need to check whether two anonymous routers are mergeable. To do that, we compute all the interhost shortest paths in the topology after merging and compare them with traceroute results one by one. Suppose N is the number of hosts, n_k is the number of known routers, and n_i is the number of anonymous routers in the initially inferred topology. Computing single-source shortest paths in a graph with V vertices by the Dijkstra algorithm takes $O(V^2)$ time [28], i.e., $O((N + n_k + n_i)^2)$ in our topology. To compute all the interhost shortest paths, the complexity is $O(N(N + n_k + n_i)^2)$. There are a total of $O(N^2)$ paths to be compared, thus the total complexity of checking one pair of anonymous routers is $O(N(N + n_k + n_i)^2 + N^2) = O(N(N + n_k + n_i)^2)$. Furthermore, it has been shown that the mergeable relationship is not transitive. That is, if $*_1$ is mergeable with $*_2$, and $*_2$ is mergeable with $*_3$, it does not mean that $*_1$ is mergeable with $*_3$. An additional check between $*_1$ and $*_3$ is necessary [12]. In summary, given n_i anonymous routers in the topology, at least $O(n_i)$ pairs of anonymous routers need to be compared (in the worst case $O(n_i^2)$ pairs), leading to a total of at least $O(N(N + n_k + n_i)^2 n_i)$ complexity.

Simulations and Internet measurements indicate that n_k and n_i are usually much larger than N, leading to a high check complexity. We hence relax the consistency constraints by allowing some inconsistent merging. We propose two algorithms to merge anonymous routers in the following sections.

B. Isomap Merging Algorithm

Isomap estimates point coordinates in a multidimensional space given the distances between them [15]. We can use Isomap to estimate router coordinates based on traceroute results. In this way, multiple occurrences of the same anonymous router may result in similar coordinates and can then be merged.

In the following, we first review Isomap and then present router merging algorithms based on the delays or hops in traceroutes. At last, we analyze the algorithm complexity.

Multidimensional scaling (MDS) and principal component analysis (PCA) have been widely applied to capture the intercorrelation of high-dimensional data in low-dimensional space. PCA finds a low-dimensional embedding of data points that best preserves their variance as measured in the high-dimensional input space. Classical MDS finds an embedding that preserves the interpoint distances, which is equivalent to PCA when the distances are Euclidean. However, MDS requires the distances between all pairs of points as input. If the missing distances are simply replaced by infinity values, the accuracy of results would be seriously affected. Note that it is impossible to obtain pairwise router distances from traceroutes, therefore MDS is not so useful here.

In this paper, we turn to Isomap, which allows an incomplete distance matrix as input to estimate point coordinates in a multidimensional space. Isomap is, in fact, a generalized MDS method. It views the problem of high dimensionality to low dimensionality transformation as a graph problem. The Isomap algorithm consists of three steps.

1) Given a distance matrix, Isomap first constructs a neighborhood graph on top of the points. Namely, each point needs to select some points as its neighbors and adds edges to them. The neighbors can be the points within a certain distance range or a certain number of closest points. All the points, hence, form a connected graph.
2) Isomap then computes pairwise shortest-path distances in the neighborhood graph by the Floyd–Warshall algorithm or Dijkstra algorithm. The distance between any two points (in the neighborhood graph) is then known and a complete distance matrix is available.
3) In the final step, Isomap applies MDS to the complete distance matrix to estimate point coordinates.

In a traceroute result, the network distance between the source and an intermediate known router is available. It can be in terms of delays (RTT) or hops. Delay-based embedding [13], [14], [29]–[31] is often more accurate than hop-based embedding [30], leading to more accurate merging. This is because the RTT between two hosts often correlates with their geographic distance, which is approximately in a two-dimensional Euclidean space. However, delay-based embedding has the following drawbacks. (a) The link delay may not be accurate and stable, especially in heavy-loaded networks. (b) The delays associated with anonymous routers are not available from traceroutes. Therefore, their estimated coordinates are inaccurate even if the embedding of known routers and hosts is fully accurate.

In the following, we call the delay-based Isomap merging algorithm the *Isomap-delay* algorithm, and the hop-based Isomap merging algorithm the *Isomap-hop* algorithm. In real applications, users may choose either one of them. We describe the merging algorithms as follows.

1) *Initial pruning*: Check the neighbors of anonymous routers. If two anonymous routers or one anonymous router and one known router share the same neighbors (known routers or hosts), merge them directly. (To check whether an anonymous router is mergeable to some known router, we only need to compare the anonymous router with its neighbors' neighbors.) For example, in Fig. 1(b), $*_1$ and $*_2$ lie between host 1 and router R_1. We can merge them into one router. The reason of such pruning is that these merging preserves both the distance and the trace

TABLE I
DELAY INFORMATION FROM TRACEROUTE RESULTS (UNIT: ms)

Src-Dest	Router-level Path	Delay Measured in Traceroute	Delay Computed from Traceroute	Delay Associated with Anonymous Routers
$1-2$	$1-R_1-*_1-R_2-2$	$d(1,R_1)=3$, $d(1,R_2)=14$, $d(1,2)=21$	$d(R_1,R_2)=11$, $d(R_1,2)=18$, $d(R_2,2)=7$	$d(R_1,*_1)=d(*_1,R_2)=5.5$, $d(1,*_1)=8.5$, $d(*_1,2)=12.5$
$1-3$	$1-R_1-*_2-R_3-3$	$d(1,R_1)=3$, $d(1,R_3)=12$, $d(1,3)=18$	$d(R_1,R_3)=9$, $d(R_1,3)=15$, $d(R_3,3)=6$	$d(R_1,*_2)=d(*_2,R_3)=4.5$, $d(1,*_2)=7.5$, $d(*_2,3)=10.5$
$2-3$	$2-R_2-*_3-R_3-3$	$d(2,R_2)=7$, $d(2,R_3)=19$, $d(2,3)=25$	$d(R_2,R_3)=12$, $d(R_2,3)=18$, $d(R_3,3)=6$	$d(R_2,*_3)=d(*_3,R_3)=6$, $d(2,*_3)=13$, $d(*_3,3)=12$

consistencies. Furthermore, in the Internet, the path segment between a pair of routers two hops away is usually stable. Therefore, this pruning works in most cases.

2) *Construction of distance matrix*: We need to distinguish here the operations in Isomap-delay and Isomap-hop algorithms.

a) Isomap-delay algorithm: Collect and analyze round-trip delays from traceroute results. In a traceroute path, the delay between any two known nodes (known routers or hosts) either is directly available or can be computed. However, the delays associated with anonymous routers are not available. Suppose A and B are two valid IP addresses in a traceroute, sandwiched by a list of anonymous routers $b_1, \ldots, b_n$, in that order. We assume that these anonymous routers are evenly distributed between A and B, and accordingly compute $delay(b_i, b_j)$ as $(j-i)/(n+1) \times delay(A, B)$, where $delay(X_1, X_2)$ is the delay between X_1 and X_2.
Suppose the total number of nodes in the inferred topology (including known and anonymous routers, and hosts) is n_t. We build a $n_t \times n_t$ distance matrix G as

$$G(i,j) = \begin{cases} 0, & \text{if } i=j \\ \min\left(d(i,j), d(j,i)\right), & \text{if both } d(i,j) \text{ and} \\ & \quad d(j,i) \text{ exist} \\ d(i,j), & \text{if only } d(i,j) \text{ exists} \\ d(j,i), & \text{if only } d(j,i) \text{ exists} \\ \infty, & \text{otherwise} \end{cases}$$

where $d(i,j)$ is the minimum delay from i to j in traceroute results.

b) Isomap-hop algorithm: Collect network connectivity information from traceroute results, and build a symmetric $n_t \times n_t$ distance matrix G' as

$$G'(i,j) = \begin{cases} 0, & \text{if } i=j \\ 1, & \text{if } i \text{ and } j \text{ are directly connected} \\ & \quad \text{in at least one path} \\ \infty, & \text{otherwise.} \end{cases}$$

3) *Coordinate estimation*: Apply Isomap to G or G' to compute the coordinates of routers and hosts. It has been shown that Internet coordinates can be approximately modeled by multidimensional Euclidean space [13], [14], [29], [30]. We hence use five-dimensional (5-D) Euclidean space in our study.

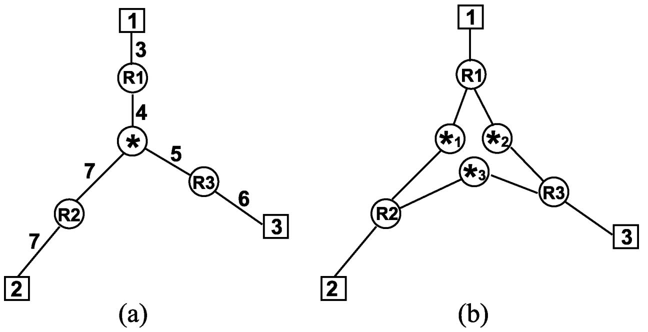


Fig. 3. An example of topology inference. (a) Actual topology. (b) Initially inferred topology.

4) *Router merging*: Compute the distance between any pair of anonymous routers according to their coordinates. Merge anonymous routers as follows.

a) Merge two anonymous routers within distance Δ_1.

b) Merge two anonymous routers which share one same neighbor (known routers or hosts) and are within distance Δ_2.

c) Do not merge two anonymous routers which appear in the same path.
Δ_1 and Δ_2 are two predefined thresholds. Clearly, a large threshold increases incorrect merging, while a small one decreases correct merging.

Let us illustrate a merging example in Fig. 3. Fig. 3(a) shows the actual underlay topology, which contains three hosts labeled as 1, 2, and 3, three known routers labeled as R_1, R_2 and R_3, and one type-1 anonymous router. The labels along lines indicate the delays of links in the unit of milliseconds (ms). With pairwise traceroutes (i.e., path $1 \rightarrow 2$, $1 \rightarrow 3$, and $2 \rightarrow 3$), we obtain an inferred topology, as shown in Fig. 3(b).

In the Isomap-delay algorithm, we get the delay information as Table I shows. The third column "delay measured in traceroute" shows the delays directly returned by traceroutes. The fourth column shows the delays among known routers and hosts which are computed according to router sequences in paths and the directly measured delays. The fifth column shows the delays associated with anonymous routers by assuming these anonymous routers are evenly distributed between their known neighbors.

We then construct the distance matrix G as Table II shows. Isomap takes this distance matrix as input and estimates the coordinates of $*_1$, $*_2$ and $*_3$ in 5-D space as (2.36, 2.02, 0, 0, 0), (−3.58, 1.20, 0, 0, 0), and (0.75, −2.19, 0, 0, 0), respectively. As

TABLE II
DISTANCE MATRIX G

	1	2	3	R_1	R_2	R_3	$*_1$	$*_2$	$*_3$
1	0	21	18	3	14	12	8.5	7.5	∞
2	21	0	25	18	7	19	12.5	∞	13
3	18	25	0	15	18	6	∞	10.5	12
R_1	3	18	15	0	11	9	5.5	4.5	∞
R_2	14	7	18	11	0	12	5.5	∞	6
R_3	12	19	6	9	12	0	∞	4.5	6
$*_1$	8.5	12.5	∞	5.5	5.5	∞	0	∞	∞
$*_2$	7.5	∞	10.5	4.5	∞	4.5	∞	0	∞
$*_3$	∞	13	12	∞	6	6	∞	∞	0

TABLE III
DISTANCE MATRIX G'

	1	2	3	R_1	R_2	R_3	$*_1$	$*_2$	$*_3$
1	0	∞	∞	1	∞	∞	∞	∞	∞
2	∞	0	∞	∞	1	∞	∞	∞	∞
3	∞	∞	0	∞	∞	1	∞	∞	∞
R_1	1	∞	∞	0	∞	∞	1	1	∞
R_2	∞	1	∞	∞	0	∞	1	∞	1
R_3	∞	∞	1	∞	∞	0	∞	1	1
$*_1$	∞	∞	∞	1	1	∞	0	∞	∞
$*_2$	∞	∞	∞	1	∞	1	∞	0	∞
$*_3$	∞	∞	∞	∞	1	1	∞	∞	0

a result, the distances between $*_1$ and $*_2$, $*_1$ and $*_3$, and $*_2$ and $*_3$ are computed as 6.00, 4.51 and 5.50 ms, respectively. If Δ_1 is set to 10 ms, we can merge all the three anonymous routers.

In the Isomap-hop algorithm, we construct a distance matrix G' as in Table III. Applying Isomap to G', we obtain the coordinates of $*_1$, $*_2$, and $*_3$ as (0.65, −1.53, −0.75, 0.05, 0.10), (0.90, 1.02, 0.75, 0.05, −0.06), and (−1.55, 0.51, 0.75, −0.10, −0.04), respectively. The distances between $*_1$ and $*_2$, $*_1$ and $*_3$, and $*_2$ and $*_3$ are 2.74, 2.64, and 2.74, respectively. With proper Δ_1 and Δ_2, we may merge two or three of them.

We finally discuss the complexity of the algorithms, given that the time and space complexities of Isomap are $O(M^3)$ and $O(M^2)$, respectively, where M is the number of input points. We first analyze the time complexity. In the pruning procedure, we compare all $O(n_i^2)$ pairs of anonymous routers. Each anonymous router has only two neighbors since each anonymous router is assumed to be a unique one. Therefore, the comparison of one pair takes $O(1)$ time. To handle type-2 routers, we compare each anonymous router with its neighbors' neighbors. In the worst case, we need to compare $O(n_i n_k)$ pairs of routers. Each comparison takes $O(1)$ time since each anonymous router has two neighbors (if a known router has multiple neighbors, a hashing function can be used to organize its neighbors). As a result, the whole pruning procedure takes $O(n_i^2 + n_i n_k)$ time. The construction of the distance matrix needs to process a total of $O(N^2)$ paths. We assume that the number of routers in a path does not exceed a certain constant, therefore the complexity of constructing the distance matrix is $O(N^2)$. The Isomap step takes $O((N + n_k + n_i)^3)$ time. Afterwards, it takes $O(n_i^2)$ time to compute the distances between anonymous routers and merge them. In total, the overall complexity is $O(n_i^2+n_i n_k+N^2+(N+n_k+n_i)^3+n_i^2) = O((N+n_k+n_i)^3)$.

The space complexity is analyzed as follows. The initially inferred topology contains $(N + n_k + n_i)$ nodes. The links among known routers and hosts take up at most $O((N + n_k)^2)$ storage space. The links associated with anonymous routers can be stored in $O(n_i)$ space, because each anonymous router has two neighbors and two adjacent links. So the initially inferred topology can be stored in $O((N + n_k)^2 + n_i)$ space. The distance matrix, Isomap, and the coordinates need at most $O((N + n_k + n_i)^2)$, $O((N + n_k + n_i)^2)$ and $O(N + n_k + n_i)$ spaces, respectively. Therefore, the total space complexity is $O((N + n_k + n_i)^2)$.

C. Neighbor Matching Algorithm

We now present a simpler algorithm, the neighbor matching algorithm, which trades off some accuracy for lower complexity. We merge the pairs of anonymous routers which share at least one neighbor (known router or host) and do not appear in the same traceroute path. We keep comparing all the anonymous router pairs and repeat this procedure until no more pairs can be merged. For example, in Fig. 3(b), we merge $*_1$ and $*_2$ because they have the same neighbor R_1. Denote this new router as $*_{12}$, which keeps all the links previously adjacent to $*_1$ or $*_2$. We proceed to merge $*_{12}$ and $*_3$ since they share the same neighbors: R_2 and R_3. In this way, we finally merge all the anonymous routers together. Clearly, this approach may over-merge anonymous routers.

The time complexity of the neighbor matching algorithm is roughly analyzed in terms of the total number of router pairs compared. In the first iteration, we compare all $O(n_i^2)$ anonymous router pairs and possibly merge some of them. Suppose we merge k_1 pairs of routers in this iteration. In the second iteration, we only need to compare these k_1 newly generated routers with each other and with other routers, i.e., $O(k_1 \times k_1 + k_1 \times (n_i - k_1 - 1)) = O(k_1 \times n_i)$ pairs. Suppose there are a total of t iterations before the algorithm stops, and in each iteration, $k_1, k_2, \ldots, k_t$ pairs are merged, in that sequence. The total number of pairs that need to compare is then

$$\begin{aligned} O\left(n_i^2 + \sum_{j=1}^{t}(k_j \times n_i)\right) &= O\left(n_i^2 + n_i \times \sum_{j=1}^{t} k_j\right) \\ &\le O\left(n_i^2 + n_i \times (n_i - 1)\right) \\ &= O\left(n_i^2\right). \end{aligned}$$

Regarding the space complexity, observe that each merging decreases the number of routers in the topology by one and also decreases the number of links. The maximum storage space is then required for the initially inferred topology, which is $O((N + n_k)^2 + n_i)$.

V. ILLUSTRATIVE SIMULATION RESULTS

In this section, we present the simulation results on Internet-like and real Internet topologies. The results and conclusions on different topologies are qualitatively the same. We hence focus on one topology (i.e., the PlanetLab topology) and only selectively present some results on other topologies.

We conduct traceroute measurement on PlanetLab [32]. We randomly select 79 nodes from PlanetLab and conduct pairwise traceroutes to obtain the underlay topology among them. Due to network and node dynamics (some nodes unexpectedly failed during our measurements), a small portion of the traceroutes cannot be completed. The resultant topology contains 5589 overlay paths (out of total $79 \times 78 = 6162$ ones), around 1950 links and 946 known routers. In addition, the number of occurrences of the three types of anonymous routers are 82, 184, and 208, respectively. To evaluate our algorithms, we select around 60 nodes from the topology and neglect traceroutes among them that contain anonymous routers. We assume that paths are symmetric by only using one path between a pair of nodes. By careful selection of the nodes, the remaining topology consists of around 96% of the pairwise paths. We use this topology as the complete underlay topology among the nodes.

We also evaluate our algorithms on a real router-level Internet topology. This topology is obtained by the Mercator project and Lucent Bell Laboratories in November 1999 [21]. It contains 284,805 routers and 860,683 links. A weakness of this topology is that it only keeps connectivity information but not router-level delay, and therefore, it is not so useful to the study without anonymous routers.

Finally, we generate five Transit-Stub topologies with Georgia Tech's network topology generator [33]. The topologies are a two-layer hierarchy of transit networks (with 8 transit domains, each with 16 randomly-distributed routers) and stub networks (with 256 domains, each with 12 randomly distributed routers). Each topology contains 3200 routers and about 20,000 links. A host is connected to a stub router with 1 ms delay, while the delays of core links are given by the topology generator. On the Mercator and Transit-Stub topologies, we use shortest-path routing to identify a path between a pair of hosts and assume that paths are symmetric.

A. *Results in the Absence of Anonymous Routers*

We first evaluate the topology inference algorithms in the absence of anonymous routers. We use GNP to estimates host coordinates. We select 20 landmarks based on N-cluster-median criterion, as in [13]. We define the following evaluation metrics in our study.

- *Link ratio* (β), defined as the ratio of the number of links in the inferred topology to the total number of links in the actual underlay topology. Note that in the absence of anonymous routers, every link appearing in the inferred topology is an actual link on the underlay. Therefore, the inferred topology is exact if and only if $\beta = 1$.
- *Router ratio* (γ), defined as the ratio of the number of routers in the inferred topology to the total number of routers in the actual underlay topology. In the absence of anonymous routers, every occurrence of a router in the inferred topology corresponds to an actual router on the underlay, and exact match in all the routers happens if and only if $\gamma = 1$.
- *Measurement load* (R), defined as the number of traceroutes performed to achieve a certain level of link or router ratio. In the absence of anonymous routers, $R = (N-1)/2$ is enough to infer a topology with full accuracy. This is the upper bound of the number of traceroutes required.

We also evaluate the theoretically minimum number of traceroutes to reconstruct an underlay topology. This minimum can be obtained by solving a *set-covering* problem. An instance

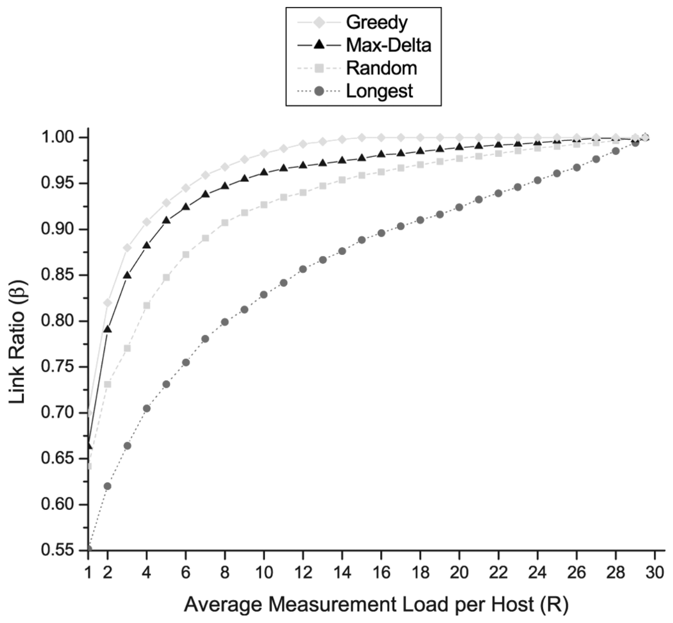


Fig. 4. Link ratio versus average measurement load per host ($N = 60$).

(X, F) of the set-covering problem consists of a finite set X and a family F of subsets of X. Every element of X belongs to at least one subset in F. The target is to find a minimum-size subset $C \subseteq F$ whose members cover all the elements of X [28]. In our inference problem, X is the set of all routers or links among the hosts. A subset in F represents routers or links in a traceroute path. We would like to select the minimum number of traceroutes to reconstruct the underlay topology, in terms of either routers or links. Note that we are only interested in the total number of traceroutes while not considering measurement parallelism among hosts. As well known, the set-covering problem is NP-hard. We use the traditional greedy approximation to address it, which has been proven to be polynomial-time $(\ln |X|+1)$-approximable [28]. In the following figures, this result is denoted as "Greedy." Note that this result is based on the assumption that we have complete knowledge of the underlay before traceroutes, and is hence not achievable in practice.

We do not present the results on the Mercator topology since it does not contain delays and we cannot test Longest Path Probing and Max-Delta Probing on it.

PlanetLab Topology: Fig. 4 plots the link ratio versus R, where the group size N is 60. Given a certain measurement load, the greedy algorithm can reveal the most links. Max-Delta's performance is close to that of Greedy. To discover a completely accurate topology with $\beta = 1$, Greedy, Max-Delta, Random, and Longest require the measurement loads of 15, 29.5, 29.5, and 29.5 at each host, respectively. Except Greedy, all the other three algorithms require full measurements to discover all the underlay links. However, if some accuracy, say 5% links, can be compromised, substantial reduction in measurement load can be achieved, which is 53% in Greedy, 70% in Max-Delta, 53% in Random, and 19% in Longest. As shown, although most links can be discovered in the first several iterations, there exist a few ones that are difficult to be discovered. If we neglect these links, the measurement load can be substantially reduced.

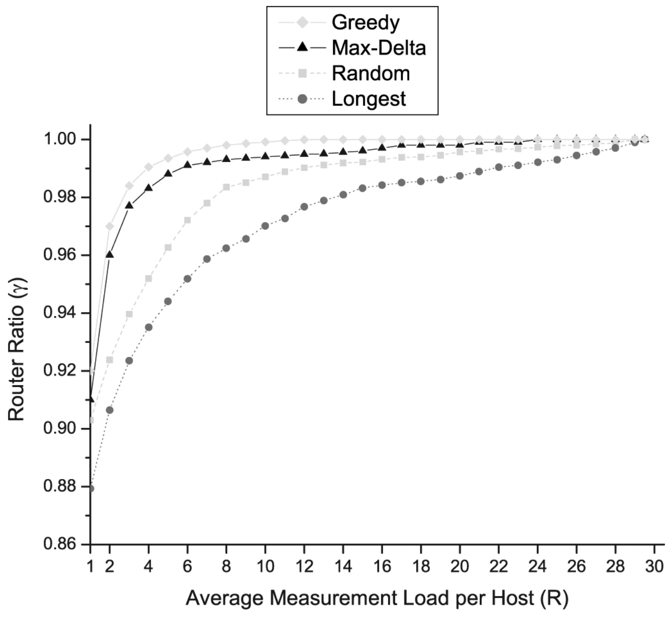


Fig. 5. Router ratio versus average measurement load per host ($N = 60$).

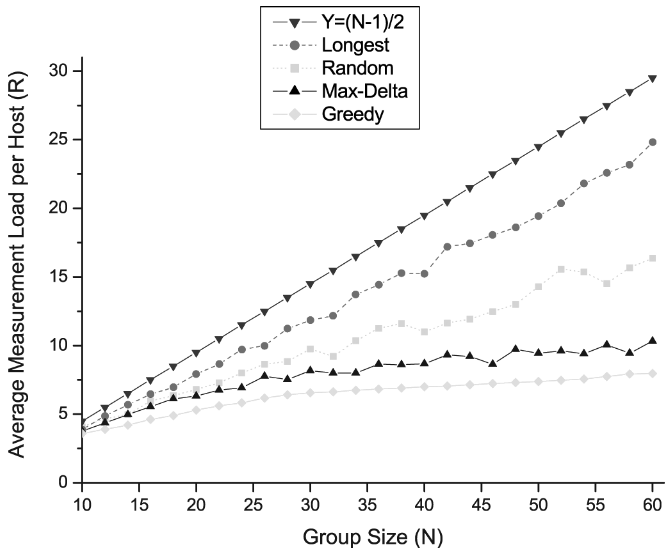


Fig. 6. Average measurement load per host to achieve $\beta = 0.95$.

Fig. 5 shows the router ratio versus R. Similar to Fig. 4, Greedy performs the best, Max-Delta the next, and then Random and Longest. By comparing Figs. 4 and 5, we see that routers are much easier to be discovered than links. A low number of traceroutes can lead to a high router ratio but a relatively lower link ratio. This is because a router is often adjacent to multiple links and can be discovered once one of these links is discovered.

Fig. 6 shows the average measurement load per host R versus N to achieve $\beta = 0.95$. As mentioned, the line $Y = (N-1)/2$ indicates the number of traceroutes in the worst case to discover the network with full accuracy. Max-Delta outperforms Random, Longest, and the full measurement method by achieving averagely 20%, 36%, and 47% reductions in the measurement load, respectively. Its result is around 22% higher than Greedy. Longest does not perform well. We find that in

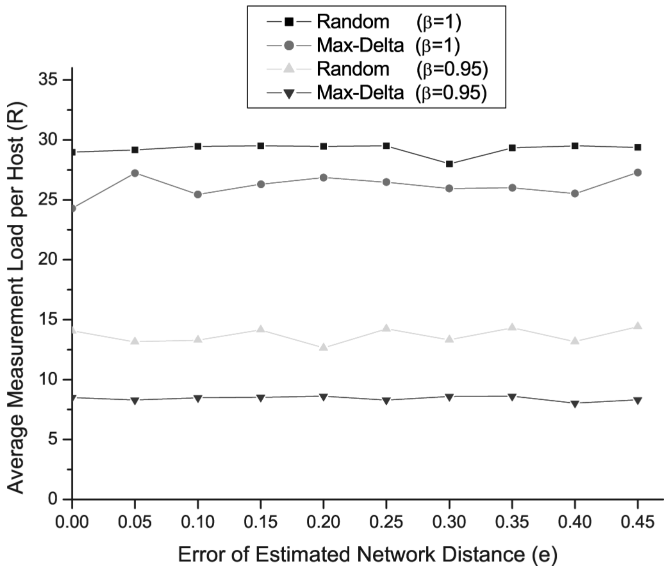


Fig. 7. Performance of Max-Delta and Random in the presence of distance estimation error ($N = 60$).

Longest there are always some outliers being tracerouted by many other hosts. This leads to inefficiency in discovering new links. As the group size increases, the measurement loads of all the algorithms increase. Max-Delta increases much slower than Random, Longest, and the full measurement method, which means that it is more scalable to large groups.

Fig. 7 shows the performance of Max-Delta and Random with different estimation errors in network distance. We assume that the relative error in RTT estimation is uniformly distributed within $(-e, e)$, $0 \leq e < 0.5$. We see that their performance does not sensitively depend on the error, even with e as high as 45%. Clearly, Random does not use RTT information in selecting traceroute targets and its result does not depend on the estimation error. Max-Delta shows its high robustness to the distance estimation error.

Transit-Stub Topologies: We further evaluate the inference algorithms on the Transit-Stub topologies. Fig. 8(a) shows the link ratio versus R, where the group size N is 256. The trends of the results are similar to those in Fig. 4. Greedy and Max-Delta achieve the best and second best performance, while Longest and Random achieve the worst and second worst performance. Max-Delta has significantly outperformed Random and Longest in the first several iterations. When R is 9, Max-Delta can discover 95.4% links in the topology, while Random and Longest can only discover 86.5% and 72.2% links, respectively. Therefore, given a certain requirement on the link ratio, say, $\beta = 0.95$, Max-Delta can achieve it with much less traceroutes than Random and Longest.

Fig. 8(b) shows the average measurement load per host R versus N to achieve $\beta = 0.95$. Longest has the worst performance. It requires almost the full measurement to achieve $\beta = 0.95$. The reason has been explained above. On average, Max-Delta achieves 62% and 77% reduction in the measurement load than Random and Longest, respectively. Greedy significantly outperforms Max-Delta only when the group size is large ($N > 350$ in the figure). In general, Max-Delta achieves slightly worse performance than Greedy.

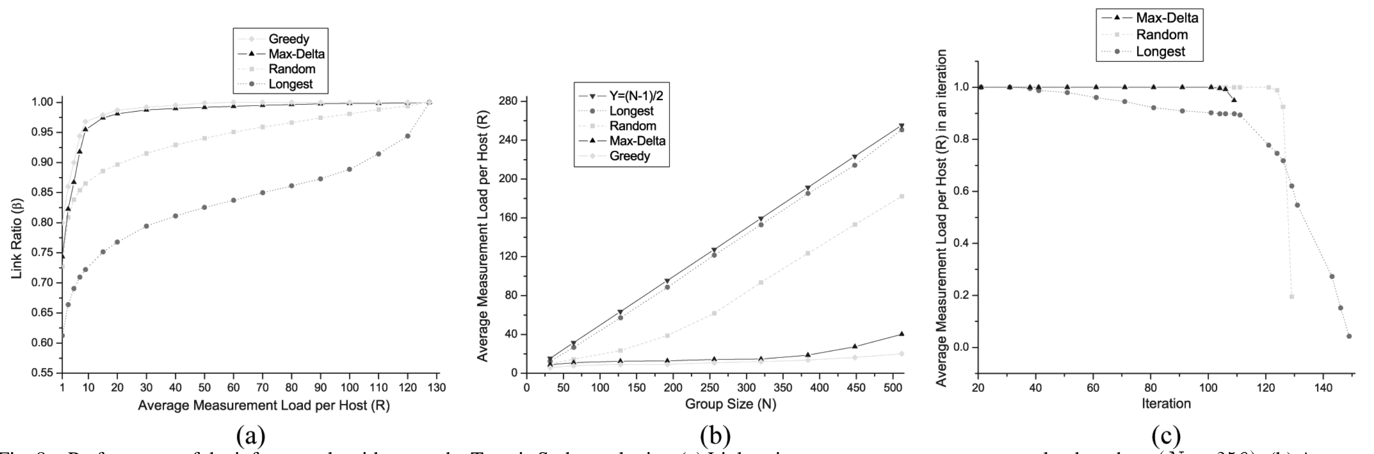


Fig. 8. Performance of the inference algorithms on the Transit-Stub topologies. (a) Link ratio versus average measurement load per host $(N = 256)$. (b) Average measurement load per host to achieve $\beta = 0.95$. (c) Average measurement load per host in different iterations of a typical simulation $(N = 256)$.

Fig. 8(c) shows the average measurement load per host in different iterations of a typical simulation. An average measurement load of 1.0 means perfect parallelism in measurement. That is, each host is able to find a traceroute target and conducts traceroute once in the iteration. The end of a curve in the figure indicates the stop of the inference algorithm, namely, discovering the fully underlay topology. Max-Delta, Random, and Longest arrive at the end at the 109th, 129th, and 149th iterations, respectively. Clearly, Max-Delta is the quickest to discover a fully complete underlay topology. In fact, Max-Delta achieves perfect parallelism in its first 101 iterations. But Longest can only achieve perfect parallelism in its first 31 iterations. In the following iterations, some hosts will not be able to find any traceroute targets. Clearly, these hosts tamper the parallelism in inference and increase the inference time.

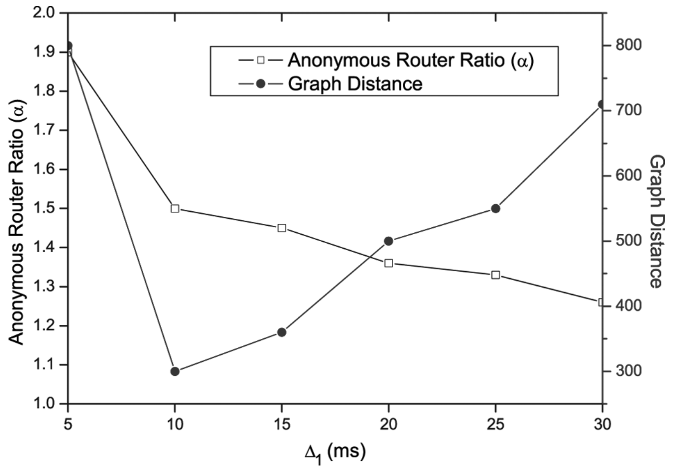


Fig. 9. Parameter tuning for Δ_1 in the Isomap-delay algorithm.

B. Results With Anonymous Routers

In the presence of anonymous routers, link and router ratios are no longer sufficient to evaluate an inferred topology. This is because $\beta = \gamma = 1$ does not mean a match in topologies (we may mistakenly merge or introduce routers/links). We therefore introduce the following additional metrics.

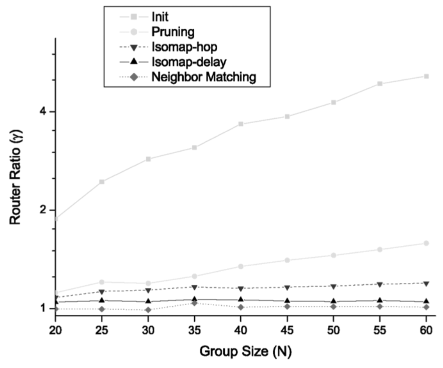
- *Anonymous router ratio* (α), defined as the ratio of the number of anonymous routers in the inferred topology to the number of anonymous routers in the actual topology.
- *Graph distance*, defined as the minimum number of primitive operations (i.e., vertex insertion, vertex deletion and vertex update) applied to the inferred topology to make it isomorphic with the actual topology. This metric is formally defined in [34]. The smaller the graph distance is, the more similar two graphs are.
- *Hop gap*, the hop gap between a pair of hosts A and B is defined as $1 - (Hop(A, B) \text{ in the inferred topology}/ Hop(A, B) \text{ in the actual topology})$. We are interested in the average hop gap among all pairs of hosts.

In our simulations, we assume that full traceroutes are used (i.e., $R = (N - 1)/2$). Unless otherwise stated, a random 5% of the routers are anonymous (type-1).

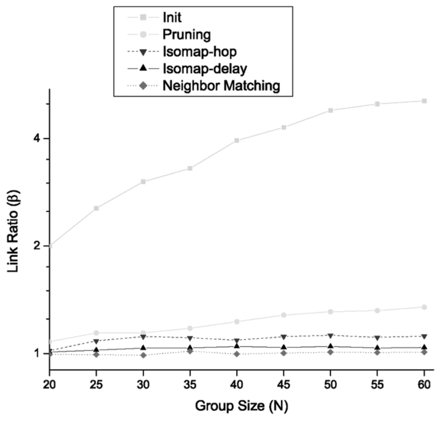
PlanetLab Topology: Fig. 9 shows the performance of the Isomap-delay algorithm with different Δ_1 values. Clearly, as Δ_1 increases, more anonymous routers are merged and α in the resultant topology decreases. However, when Δ_1 increases, the possibility of incorrect merging also increases. This is shown from the curve of the graph distance, which decreases first and then increases. We hence set Δ_1 to 10 ms to achieve good tradeoff between α and the graph distance. Similarly, we set Δ_2 to 30 ms. In the Isomap-hop algorithm, it is good to set them to 0.05 and 0.2, respectively.

Fig. 10 shows the performance of the merging algorithms on the PlanetLab topology. The lines labeled “Init” and “Pruning” indicate the results on the initially inferred topology and the topology after simple pruning, respectively.

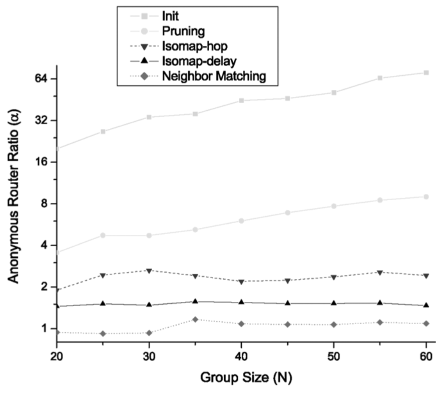
In Fig. 10(a), we clearly see that there is a high router inflation in the initially inferred topology, with an average of 3.6 and a maximum of 5.1. The router ratio increases with the group size, therefore in a larger group the inflation will be more serious. Simple pruning can averagely reduce the inflation by 60%, but γ is still high in the resultant topology. The three merging algorithms further reduce γ to close to 1, with an average of 1.15, 1.05, and 1.009, respectively. In all the three algorithms, γ increases slowly with the group size. It shows that these algorithms are efficient even in a large-scale network. Among the three merging algorithms, neighbor matching merges the



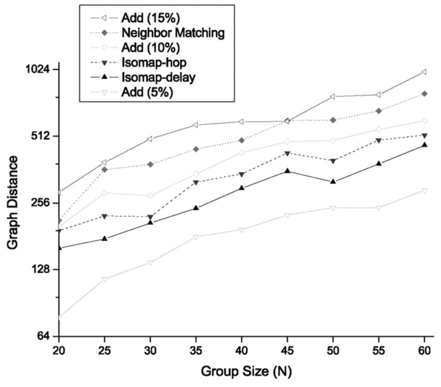
(a)



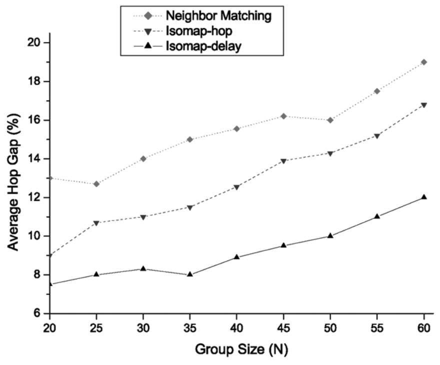
(b)



(c)



(d)



(e)

Fig. 10. Performance of the different merging algorithms on the PlanetLab topology (with 5% routers as anonymous). (a) Router ratio. (b) Link ratio. (c) Anonymous router ratio. (d) Graph distance. (e) Average hog gap.

most anonymous routers while Isomap-hop merges the least. In fact, some γ values in neighbor matching are less than 1, which shows that it is too aggressive in merging and tends to over-merge routers.

Fig. 10(b) and (c) show the link ratio β and the anonymous router ratio α, respectively. We again see that there is high inflation in the inferred topology without merging, especially for anonymous router ratio. In Fig. 10(c), with the merging algorithms, anonymous router ratios are reduced to a low value around or less than 2, with reductions of around 95% and 65% as compared with the initially inferred topology and the topology with only pruning, respectively.

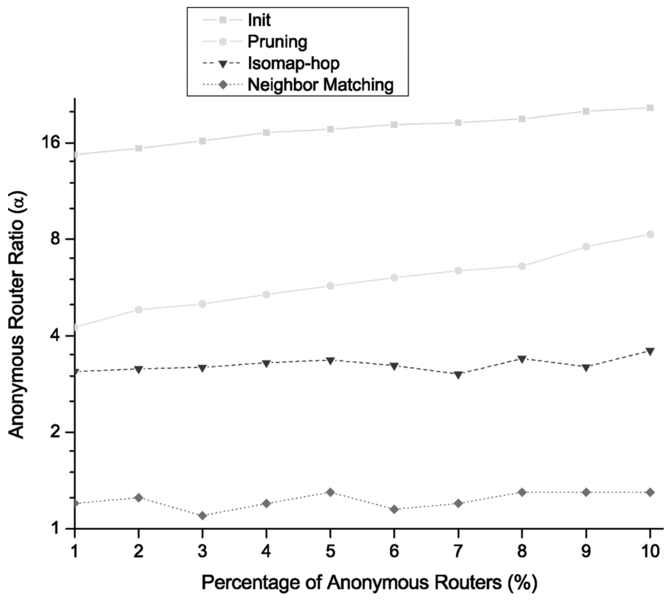
Fig. 10(d) shows the graph distance between the inferred topology and the actual topology. In the figure, we also show the graph distance between the actual topology and a topology generated by randomly adding a certain percentage of links to the actual one. As the group size increases, the graph distances of all these topologies increase, mainly due to higher inflation. Among the three inferred topologies, the Isomap-delay topology is the most similar to the actual topology, followed by Isomap-hop, and then neighbor matching. Isomap-delay infers a topology with similar graph distance as the topology with about 5% additional links. The Isomap-hop topology performs similarly to the one with 10% additional links.

Fig. 10(e) shows the average hop gap of the three topologies. Isomap-delay performs the best, while neighbor matching performs the worst. All of them achieve relatively low average hop gap (less than 20%). We expect that most overlay applications are not sensitive to such discrepancy.

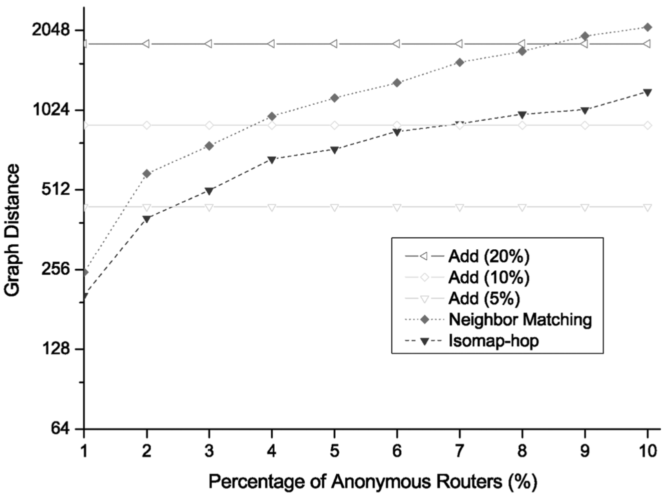
In summary, the simplest neighbor matching algorithm tends to over-merge routers and introduces the highest error. Isomap-delay achieves better performance by its higher complexity. It also performs better than Isomap-hop for all the metrics considered. This is because Isomap works the best on Euclidean distances among points, but the Isomap-hop algorithm only uses 0/1 hop values, which introduces error in the fitting of routers to a multidimensional space. However, in the networks where delay is not stable and accurate, Isomap-hop is more useful and applicable.

Mercator Topology and Transit-Stub Topologies: We further conduct simulations on the Mercator and Transit-Stub topologies. Due to the lack of RTT among routers in the Mercator topology, we do not evaluate the Isomap-delay algorithm on it. Instead, we evaluate the other two algorithms with different percentages of anonymous routers on this topology. We note that the conclusions on these topologies are qualitatively the same as that on the PlanetLab topology. We hence only show some representative results here.

Fig. 11 shows the performance of the merging algorithms on the Mercator topology. The anonymous routers significantly inflate the network. Simple pruning can reduce the inflation by 66%. Based on it, Isomap-hop and neighbor matching make further reduction by 44% and 79%, respectively. Generally, neighbor matching merges more anonymous routers than

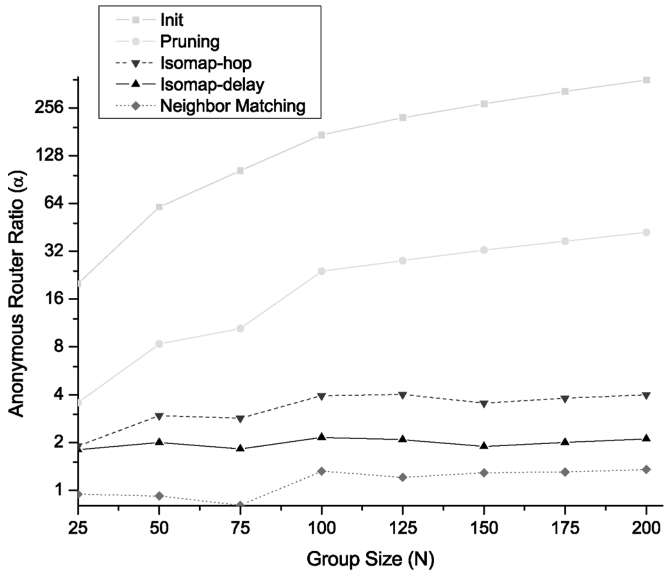


(a)

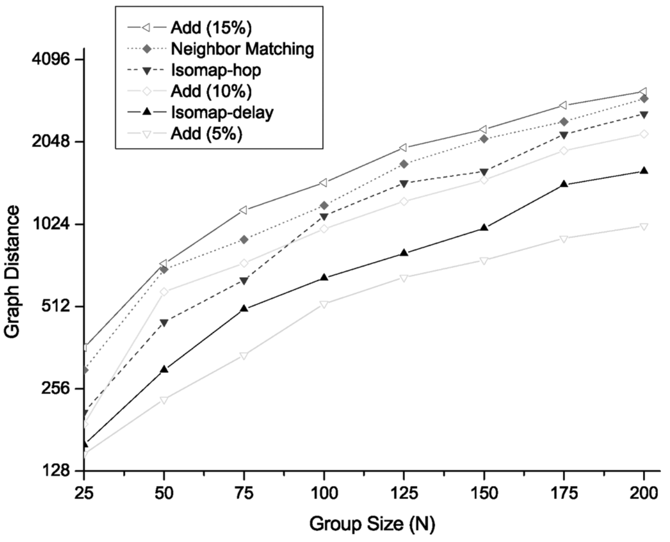


(b)

Fig. 11. Performance of the different merging algorithms on the Mercator topology $(N = 100)$. (a) Anonymous router ratio. (b) Graph distance.



(a)



(b)

Fig. 12. Performance of the different merging algorithms on the Transit-Stub topologies (with 5% routers as anonymous). (a) Anonymous router ratio. (b) Graph distance.

Isomap-hop, but it also makes more mistakes by showing larger graph distance. In Fig. 11(b), the graph distances of the inferred topologies quickly increase as the percentage of anonymous routers increase. When the percentage of anonymous routers is large (say, larger than 7%), the merging error is also large. In that case, application-layer measurement may not be sufficient to infer a highly accurate topology.

Fig. 12 shows the performance of the merging algorithms on the Transit-Stub topologies. We note that α of the initially inferred topology in Fig. 12(a) is much larger than that in Fig. 11(a). For example, α of the initially inferred topology with 5% anonymous routers among a group of 100 hosts are 17 and 172 based on the Mercator topology and the Transit-Stub topologies, respectively. One reason is that the total number of routers in the Mercator topology is much larger than that in the Transit-Stub topologies. We have randomly selected routers to attach hosts and used shortest-path routing to identify interhost paths. With a huge amount of routers in the Mercator topology, the shortest paths have few overlaps. This is different from the case in the Transit-Stub topologies, where routers in the core are more frequently visited than others and paths adjacent to one host often share the same edge networks.

VI. Conclusion

In this paper, we study how to infer the underlay topology among a group of hosts through end-to-end measurement tools such as traceroute. Such techniques are useful in many applications, for example, constructing an efficient overlay or building a high-bandwidth overlay tree. We study two important issues in topology inference.

The first one is how to *efficiently* infer a network topology in the absence of anonymous routers. We propose Max-Delta, where a server assigns the traceroute targets for hosts so as to reveal the most undiscovered underlay information. Our simulation results show that to obtain a topology with 100% accuracy, each host needs to traceroute almost all the others, regardless of the inference algorithm. However, to obtain a topology with 95% links of the actual one, Max-Delta can reduce the measurement load at each host by 20% as compared with a random measurement method, and by 47% as compared with the full measurement method (on the PlanetLab topology). Max-Delta can, hence, infer a highly accurate topology with low measurement load.

We further study the anonymous router problem in topology inference. Anonymous routers seriously distort the inferred topology and increase the complexity of inferring a topology consistent with traceroutes. Our study shows that preserving the distance and trace consistencies in topology inference is of unpractically high complexity. We hence relax the constraints and present two fast algorithms to merge anonymous routers. Our simulation results show that the Isomap-delay algorithm performs the best out of all the algorithms studied. On the PlanetLab topology with 5% anonymous routers, it can infer a topology with anonymous router ratio around 1.5 and average hop gap around 9.2%. Most overlay applications are expected to be able to tolerate such discrepancy.

References

[1] Y. H. Chu, S. Rao, S. Seshan, and H. Zhang, “A case for end system multicast,” *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, “A scalable content-addressable network,” in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 161–172.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, “Resilient overlay networks,” in *Proc. ACM SOSP*, Oct. 2001, pp. 131–145.

[4] M. Kwon and S. Fahmy, “Topology-aware overlay networks for group communication,” in *Proc. ACM NOSSDAV*, May 2002, pp. 127–136.

[5] X. Jin, Y. Wang, and S.-H. G. Chan, “Fast overlay tree based on efficient end-to-end measurements,” in *Proc. IEEE ICC*, May 2005, pp. 1319–1323.

[6] J. Han, D. Watson, and F. Jahanian, “Topology aware overlay networks,” in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 2554–2565.

[7] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, “OpenDHT: A public DHT service and its uses,” in *Proc. ACM SIGCOMM’05*, Aug. 2005, pp. 73–84.

[8] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, J. Hellerstein, and S. Shenker, “A case study in building layered DHT applications,” in *Proc. ACM SIGCOMM*, Aug. 2005, pp. 97–108.

[9] Y. Chen, D. Bindel, H. Song, and R. Katz, “An algebraic approach to practical and scalable overlay network monitoring,” in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 55–66.

[10] Traceroute. [Online]. Available: http://www.traceroute.org/

[11] V. Jacobson, “Pathchar,” 1997. [Online]. Available: http://www.caida.org/tools/utilities/others/pathchar/

[12] B. Yao, R. Viswanathan, F. Chang, and D. G. Waddington, “Topology inference in the presence of anonymous routers,” in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 353–363.

[13] T. S. E. Ng and H. Zhang, “Predicting Internet network distance with coordinates-based approaches,” in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 170–179.

[14] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, “Vivaldi: a decentralized network coordinate system,” in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 15–26.

[15] J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, pp. 2319–2323, Dec. 2000.

[16] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang, “Maximum likelihood network topology identification from edge-based unicast measurements,” in *Proc. ACM SIGMETRICS*, 2002, pp. 11–20.

[17] M. Coates, A. Hero, R. Nowak, and B. Yu, “Internet tomography,” *IEEE Signal Process. Mag.*, vol. 19, pp. 47–65, May 2002.

[18] D. G. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan, “Topology inference from BGP routing dynamics,” in *Proc. ACM SIGCOMM IMW*, Nov. 2002, pp. 243–248.

[19] F. Wang and L. Gao, “On inferring and characterizing Internet routing policies,” in *Proc. ACM SIGCOMM IMC*, Oct. 2003, pp. 15–26.

[20] “Skitter.” 2002. [Online]. Available: http://www.caida.org/tools/measurement/skitter/index.xml

[21] R. Govindan and H. Tangmunarunkit, “Heuristics for Internet map discovery,” in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 1371–1380.

[22] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP topologies with Rocketfuel,” in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 133–145.

[23] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, “Efficient algorithms for large-scale topology discovery,” in *Proc. ACM SIGMETRICS*, Jun. 2005, pp. 327–338.

[24] P. Barford, A. Bestavros, J. Byers, and M. Crovella, “On the marginal utility of network topology measurements,” in *Proc. ACM SIGCOMM IMW*, Nov. 2001, pp. 5–17.

[25] A. Broido and K. C. Claffy, “Internet topology: Connectivity of IP graphs,” in *Proc. SPIE ITCom*, Aug. 2001.

[26] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, “On the constancy of Internet path properties,” in *Proc. ACM SIGCOMM IMW*, Nov. 2001, pp. 197–211.

[27] M. Faloutsos, P. Faloutsos, and C. Faloutsos, “On power-law relationships of the Internet topology,” in *Proc. ACM SIGCOMM*, Sep. 1999, pp. 251–262.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[29] H. Lim, J. C. Hou, and C.-H. Choi, “Constructing Internet coordinate system based on delay measurement,” in *Proc. ACM SIGCOMM IWC*, Oct. 2003, pp. 129–142.

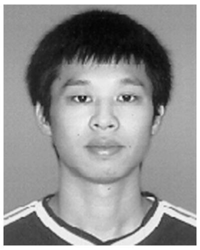
[30] M. Costa, M. Castro, A. Rowstron, and P. Key, “PIC: Practical Internet coordinates for distance estimation,” in *Proc. ICDCS*, Mar. 2004, pp. 178–187.

[31] B. Wong, A. Slivkins, and E. G. Sirer, “Meridian: a lightweight network location service without virtual coordinates,” in *Proc. ACM SIGCOMM*, Aug. 2005, pp. 85–96.

[32] Planetlab. [Online]. Available: http://www.planet-lab.org

[33] E. Zegura, K. Calvert, and S. Bhattacharjee, “How to model an internetwork,” in *Proc. IEEE INFOCOM*, Mar. 1996, vol. 2, pp. 594–602.

[34] A. N. Papadopoulos and Y. Manolopoulos, “Structure-based similarity search with graph histograms,” in *Proc. DEXA*, Sep. 1999, pp. 174–178.



Xing Jin (S’04) received the B.Eng. degree in computer science and technology from Tsinghua University, Beijing, China, in 2002. He is currently working towards the Ph.D. degree at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon.

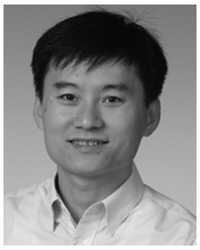
His research interests include overlay multicast with applications and QoS issues, Internet topology inference, end-to-end measurements, and peer-to-peer streaming.

Mr. Jin was awarded the Microsoft Research Fellowship in 2005. He is a Junior Editor of the *Journal of Multimedia* since 2006.

W.-P. Ken Yiu (S'03) received the B.Eng. and M.Phil. degrees in computer science from the Hong Kong University of Science and Technology (HKUST), Kowloon, in 2002 and 2004, respectively. He is currently working towards the Ph.D. degree at the Department of Computer Science and Engineering, HKUST.

His research interests include computer networks, peer-to-peer systems, multimedia networking, and network security.

Mr. Yiu was awarded the Academic Achievement Medal from HKUST in 2002, and the Sir Edward Youde Memorial Fellowship from Sir Edward Youde Memorial Fund in 2005 and 2006. He is a student member of the IEEE Computer Society.



S.-H. Gary Chan (S'89–M'98–SM'03) received the B.S.E. degree (Highest Honor) in electrical engineering from Princeton University, Princeton, NJ, in 1993, with certificates in applied and computational mathematics, engineering physics, and engineering and management systems, and the M.S.E. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1994 and 1999, respectively, with a minor in business administration.

He is currently an Associate Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon, and an Adjunct Researcher with Microsoft Research Asia, Beijing. He was a Visiting Assistant Professor in Networking with the Department of Computer Science, University of California, Davis, CA, from 1998 to 1999. During 1992–93, he was a Research Intern at the NEC Research Institute, Princeton, NJ. His research interest includes multimedia networking, peer-to-peer technologies and streaming, and wireless communication networks.

Dr. Chan is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa. He was a William and Leila Fellow at Stanford University during 1993–1994. At Princeton University, he was the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence in 1993. He served as a Vice-Chair of IEEE COMSOC Multimedia Communications Technical Committee (MMTC) from 2003 to 2006. He is a Guest Editor for the *IEEE Communication Magazine* (Special Issue on Peer-to-Peer Multimedia Streaming), 2007, and Springer *Multimedia Tools and Applications* (Special Issue on Advances in Consumer Communications and Networking), 2007. He is Co-Chair of the Multimedia Symposia for IEEE GLOBECOM (2006) and IEEE ICC (2007). He was the Co-Chair for the Workshop on Advances in Peer-to-Peer Multimedia Streaming for the ACM Multimedia Conference (2005) and IEEE ICC (2005).

Yajun Wang received the B.Eng. degree in computer science from the University of Science and Technology of China, Hefei, in 2002. He is currently working towards the Ph.D. degree at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon.

His research interests include computational geometry, combinatorics, algorithms, and data structures.