A Distributed Approach to End-to-End Network Topology Inference

Xing Jin Qiuyan Xia S.-H. Gary Chan
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
Email: {csvenus, xiaqy, gchan}@cse.ust.hk

***Abstract*— To construct an efficient overlay network, the information of underlay is important. However, the inference of an underlay topology is not easy. We consider using end-to-end measurement tools such as traceroute to infer the underlay topology among a group of hosts. Since pair-wise traceroutes among hosts take a long time and generate much network traffic, Max-Delta has been proposed to infer a highly accurate topology with a low number of traceroutes. However, Max-Delta relies on a central server to collect traceroute results and to select paths for hosts to measure. It is hence not scalable to large groups. In this paper, we investigate a distributed version of Max-Delta scheme in order to support scalable inference. In our scheme, each host joins an overlay tree before conducting traceroutes. A host then independently selects paths to traceroute and exchanges traceroute results with others through the overlay tree. As a result, each host can maintain a partially discovered topology. We have studied two key issues in the scheme, i.e., how to construct a low-diameter overlay tree and how to reduce bandwidth consumption in measurements.**

As compared to Max-Delta, our scheme is fully distributed and scalable. In the scheme, each host computes its own traceroute targets, and the computational loads are distributed to all the hosts instead of a single server. Furthermore, each host only exchanges data with a few other hosts and does not need to set up connections with all the other hosts. Simulation results show that the constructed tree has a low diameter and can support quick data exchange among hosts, and that the use of a lookup table for routers can significantly reduce bandwidth consumption in data exchange.

I. Introduction

In the recent years, overlay networks have been increasingly used to deploy network services. Examples include overlay path routing, application-layer multicast (ALM), peer-to-peer file sharing, and so on [1]–[3]. In order to build an efficient overlay network, the knowledge of underlay is 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].

We consider inferring the underlay network topology among a group of hosts by means of end-to-end measurements, where traceroute-like tools extracting the router-level path between a pair of hosts are often used [7]. Given a group of N hosts, conducting full $O(N^2)$ traceroutes among them can certainly construct an accurate topology (we do not consider measurement noise such as anonymous routers or router alias here). However, since traceroute may take as long as minutes to identify a router-level path and generate many network packets, such pair-wise measurements are costly and not scalable. We hence consider inferring an approximate topology with much fewer traceroutes.

This work was supported, in part, by the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (GHP/045/05).

Max-Delta has been proposed to efficiently infer the underlay topology among a group of hosts [6], [8]. It divides the inference process into multiple iterations. In each iteration, a central server collects the traceroute results from hosts. In order to reveal as much undiscovered information on the underlay as possible, the server will select some representative paths for hosts to traceroute in the next iteration. This process is repeated until a certain measurement accuracy or measurement cost is achieved. The simulation results on Transit-Stub topologies have shown that in a group of 256 hosts, each host only needs to traceroute around $10 - 14$ other hosts to discover 95% underlay links together [8]. This reduces the number of traceroutes at each host by 62% as compared to a random measurement method, and by 89% as compared to the full measurement method.

However, a limitation of Max-Delta is that the server may not be able to support a large group. Firstly, in each iteration, the server takes $O(V_p \log V_p + E_p + N)$ time to select a traceroute target for a host, where N is the number of hosts in the group, V_p and E_p are the numbers of nodes (including hosts and routers) and links in the partially discovered underlay topology, respectively. In total, the server takes $O(N(V_p \log V_p + E_p + N))$ time to select traceroute targets for all the N hosts in one iteration. This complexity is considerably high when the group size is large. Secondly, the server needs to periodically accept traceroute results from all the hosts and send traceroute targets to them. When the group size is large, the server may not be able to simultaneously set up so many connections.

In this paper, we propose a distributed inference scheme to support large groups. In our scheme, each host joins an overlay tree before topology inference. A host then conducts traceroutes and distributes the results to all the other hosts through the overlay tree. As a result, each host can receive traceroute results from other hosts and maintain a partially discovered topology. Based on that, a host can use Max-Delta to select traceroute targets by itself and continue conducting traceroutes.

We have studied two important issues in the scheme. The first one is how to construct the overlay tree so that hosts can quickly exchange traceroute results. Since each host sends traceroute results to others, we need to build a source-unspecific low-diameter overlay tree. Previous research has shown that this problem is NP-hard and there are no efficient distributed algorithms to address it [9], [10]. We propose a distributed tree construction algorithm based on the out-degree bounds of hosts. We first identify a host as the tree root and then insert new incoming hosts around the root. A host with a larger out-degree bound is put closer to the root in the tree. A low-diameter tree is hence formed. The second issue is how to compress traceroute results in order to reduce bandwidth consumption. We note that routers are often repeatedly visited in different traceroutes. We then set up a lookup table to map each router (i.e., the router IP and the router name) to an integer. Later on, we represent routers by integers in traceroute results.

As compared to Max-Delta, this scheme eliminates the central server from the system. It is fully distributed and scalable. In the scheme, each host computes its own traceroute targets, and the computational loads are distributed to all the hosts instead of a single server. Furthermore, each host only exchanges data with a few other hosts and does not need to set up connections with all the other hosts. The consumption of edge bandwidth at a host is hence reduced. We have done simulations on Transit-Stub topologies to evaluate the proposed scheme. The results show that the constructed tree has a low diameter and that the lookup table for routers can significantly reduce bandwidth consumption.

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. However, for ease of exposition and illustrative purpose, 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 the paper is organized as follows. In Section II we briefly review the related work. In Section III we discuss the design of the distributed inference scheme. In Section IV we present illustrative simulation results on Transit-Stub topologies. We finally conclude in Section V.

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 [11], [12]. 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 [13], [14]. 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, Mercator and Rocketfuel [15]–[17]. 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 Domain Name System (DNS) to infer ISP topologies. All these works focus on Internet-level 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 problem is hence how to reduce measurement cost.

Donnet et al. note that in large-scale traceroute measurements, a router is often repeatedly visited in different traceroutes [18], [19]. They hence propose a Doubletree algorithm to reduce the redundancy. 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. In either case, the probing stops whenever an already discovered router is met. Max-Delta and our work reduce 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 need not traceroute all its destinations because of the requirement on measurement cost or accuracy. We then design an algorithm for destination selection and select representative paths for hosts to traceroute. 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.

III. System Design

In this section, we present the distributed inference scheme. We first briefly review Max-Delta and then give an overview of the distributed inference scheme. We finally discuss in detail the key issues in the scheme.

A. *Review on Max-Delta*

Max-Delta has been proposed to efficiently infer the underlay topology among a group of hosts [6], [8]. In the scheme, hosts utilize light-weight tools such as GNP [20] or Vivaldi [21] to estimate their network coordinates and report them to a central server. The server then divides the inference process into multiple iterations. In each iteration, the server selects a target for each host to traceroute. The target is selected as follows. For a certain host A, suppose that the path between A and another host B has not been measured. The server computes the distance between A and B in the currently discovered topology $D_p(A, B)$ (using shortest path routing) and that in the real network $Euclidean(A, B)$ (using the network coordinates). If the gap between the two values

$\Delta(A,B) = D_p(A,B) - Euclidean(A,B)$ is large, it is with high probability that some links between A and B (leading to a shorter path in the discovered topology) are not discovered. For all unmeasured paths between A and other hosts, the server selects the path with the maximum Δ value as A's traceroute target. A then traceroutes the target path and reports the result to the server. The server then combines all the results obtained in the iteration and based on that, starts the next iteration on target assignment. Such process is repeated until a certain measurement accuracy or measurement cost is achieved. As discussed above, the scalability of Max-Delta is limited by the edge bandwidth and computational power of the central server. Therefore, we need to design a distributed inference scheme to support large groups.

B. Design of A Distributed Inference Scheme

We now describe a distributed scheme for topology inference. In the scheme, hosts exchange traceroute results through an overlay tree. Each host then maintains a partially discovered topology and uses the Max-Delta heuristic to select traceroute targets by itself. Suppose that A is a new incoming host. We describe the actions of A in a sequential order as follows.

1) *Estimate the network coordinates.*
   A first uses some tool (e.g., GNP or Vivaldi) to estimate its network coordinates. For example, if GNP is used, A should ping a few public landmarks and use the network distances to landmarks as well as the landmark coordinates to compute its own coordinates.
2) *Join the overlay tree.*
   A then identifies a host in the system as its parent to join the overlay tree. The detailed tree joining and maintenance mechanisms will be explained later.
3) *Conduct first-round traceroute.*
   The first-round traceroutes from all the hosts must form a connected graph among them. Otherwise, the distance between two hosts in the discovered topology D_p may be infinite. Therefore, we require A to traceroute the path to its parent in the first round. Clearly, if each host traceroutes the path to its parent in the tree, all the measured paths form a tree spanning the hosts on the overlay.
4) *Distribute the coordinates and first-round traceroute along the tree.*
   A sends its coordinates and first-round traceroute to its neighbors.
5) A then performs the following steps in parallel:
   5.1) *Select paths to traceroute.*
   A maintains a discovered topology based on its own traceroutes and traceroute results from other hosts. Based on that, A can select its traceroute targets as in Max-Delta. Clearly, it takes $O(V_p \log V_p + E_p + N)$ time to select one traceroute target.
   5.2) *Accept data, if any, from its neighbors.*
   5.3) *Periodically send data (including its own traceroute results and other hosts' traceroute results) to its neighbors.*
   A aggregates its own traceroute results and data received from its neighbors. It then periodically floods the data along the tree. Clearly, data received from a neighbor are forwarded to all its other neighbors in the tree, and its own traceroute results are sent to all the neighbors.

C. Scheme Details

In this section, we discuss two key issues in the distributed inference scheme, i.e., how to construct a low-diameter overlay tree and how to reduce bandwidth consumption in data exchange.

1) Tree Construction: In the inference process, each host needs to send traceroute results to others. We hence consider building a source-unspecific tree among hosts. To achieve rapid data exchange among hosts, we minimize the diameter of the tree, which is the longest simple path (in terms of the number of overlay hops) in the tree. In fact, previous research has shown that building a minimum-diameter degree-bounded spanning tree is NP-hard and there are no efficient distributed algorithms to address this problem [9], [10]. In this paper, we propose a distributed tree construction algorithm based on the out-degree bounds of hosts. In our algorithm, we first identify a host as the tree root and then insert new hosts around the root. Before introducing the algorithm, we first describe some settings for tree construction. In the tree construction process, we assume that the root is the only source in the system and is about to distribute data to all the other hosts. Each new host needs to identify a host as its *parent* in order to join the tree. Except for the parent, all the other neighbors of a host in the tree are called its *children*. Clearly, during real measurement, there is no such parent-child relationship between hosts since every host is a source.

Each host has an out-degree bound according to its edge bandwidth, which indicates how many children a host can have in the tree. The position of a host in the tree depends on its out-degree bound. The larger out-degree bound a host has, the closer to the root it is put. In other words, a host has a larger out-degree bound than all the hosts in its subtree. Denote B_i as the out-degree bound of host i, and denote D_i as the real out-degree of i in the tree. Furthermore, denote $Dist_i$ as the minimum number of overlay hops from i to the root in the tree. Clearly, $Dist_i = Dist_{i's\ parent} + 1$. Finally, we denote New_i as the the minimum number of overlay hops from a new host to the root in the tree if the new host becomes i's child or descendant. Given a host i, if $B_i > D_i$, $New_i = Dist_i + 1$. Otherwise, $New_i = \min\{New_y \mid \forall y \in i\text{'s children set}\}$. We require hosts in the tree to periodically exchange their $Dist$ and New values with the neighbors. All the hosts in the tree will finally know their $Dist$ and New values.

When a new incoming host wants to join the tree, it first contacts a public rendezvous point (RP) to obtain the IP address of the root. It then joins the tree as Algorithm 1 shows. We briefly explain the joining process of host i as follows. If i is the first joining host, i becomes the root and claims itself to the RP. If this is not the case, but i has a larger out-degree bound than the current root, i becomes the new root:

It accepts the current root and the children of the current root as its children, and then claims itself to the RP. If none of the above cases occur, i starts a recursive joining procedure from the root. That is, i identifies a host x to start JOIN(i, x), which is the root at the beginning. Note that by following our algorithm, we have $B_i \leq B_x$ in the JOIN(i, x) procedure (this is a pre-condition when invoking JOIN(i, x)). Therefore, i can be x's child or descendant, but cannot become x's ancestor. If x has residual out-degree (i.e., $B_x > D_x$), x accepts i as its child. Otherwise, i checks x's children. From all x's children whose out-degree bounds are smaller than that of i, the child with the maximum out-degree bound is selected, say, m. If there exists such a host m, i takes up m's position in the tree: i selects m's parent as its parent, accepts m's children as its children, and accepts m as its child. If there no such a host m (i.e., the out-degree bounds of x's children are all larger than or equal to that of i), i has to move one level down. i selects from x's children the host with the smallest New value, and repeats the joining process from this host. See Algorithm 1 for more details.

Note that the size of data exchanged between two hosts is relatively small. For example, a traceroute result requires only several kilo-bytes. Considering the capabilities of today's computers and networks, a powerful PC can have hundreds of neighbors and support simultaneous data exchange with them. The out-degree bounds of hosts in our scheme are hence much larger than those in bandwidth-demanding applications such as streaming or file sharing.

2) Reducing Bandwidth Consumption: In a traceroute result, a router is represented by its name and IP address, which often consist of $20-50$ digits or letters. On the other hand, we note that a router is often visited multiple times in different traceroutes. We can hence use a compact form to represent routers so as to reduce the size of traceroute results.

We map each router to an integer. Note that such mapping should be one-to-one so that traceroute results can be freely transformed between the two forms without any ambiguity. Furthermore, the mapping should be universal to all the hosts. Therefore, we identify one host in the tree to conduct the mapping. A possible choice is the tree root. The root maintains a lookup table for routers. When the root finds a new router in traceroute results, it inserts the router into the lookup table. It then replaces routers in traceroute results by their corresponding locations in the lookup table. In this way, each router can be represented by an integer, whose length depends on the total number of routers in the topology. The root then delivers the compact traceroute results as well as the lookup table to other hosts. Figure 1 shows an example of the raw traceroute result and compact traceroute result. Since a router may appear many times in the whole inference process, such a lookup table can significantly reduce the size of traceroute results.

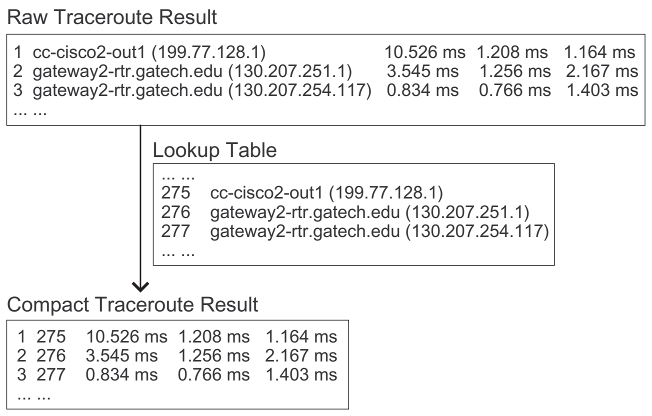
IV. ILLUSTRATIVE NUMERICAL RESULTS

In this section we evaluate our inference scheme through simulations on Internet-like topologies and measurements on

Algorithm 1 : JOINING PROCEDURE OF HOST i

```
1:  procedure TREEJOIN ($i$)
2:      if $i$ is the first joining host then
3:          $i$ becomes the root and notifies the RP.
4:          return
5:      end if
6:      if $B_i > B_{root}$ then
7:          $i$ becomes the new root: it accepts $root$ and $root$'s
            children as its children, and notifies the RP.
8:          return
9:      else
10:         JOIN ($i$, $root$)
11:         return
12:     end if
13: end procedure

14: procedure JOIN ($i$, $x$)
15:     if $B_x > D_x$ then
16:         $x$ accepts $i$ as its child.
17:         return
18:     else
19:         if $\exists t \in x$'s children set s.t. $B_t < B_i$ then
20:             select $x$'s child $m$ s.t. $B_m = \max\{B_y | \forall y \in$
                $x$'s children set AND $B_y < B_i\}$.
21:             $i$ replaces $m$ (i.e., $i$ selects $m$'s parent as its parent
                and accepts $m$'s children as its children) and
                accepts $m$ as its child.
22:             return
23:         else
24:             select $x$'s child $n$ s.t. $New_n = \min\{New_y | \forall y \in$
                $x$'s children set$\}$.
25:             JOIN ($i$, $n$)
26:             return
27:         end if
28:     end if
29: end procedure
```



```
1  cc-cisco2-out1 (199.77.128.1)                   10.526 ms  1.208 ms   1.164 ms
2  gateway2-rtr.gatech.edu (130.207.251.1)          3.545 ms   1.256 ms   2.167 ms
3  gateway2-rtr.gatech.edu (130.207.254.117)        0.834 ms   0.766 ms   1.403 ms
... ...
```

```
... ...
275   cc-cisco2-out1 (199.77.128.1)
276   gateway2-rtr.gatech.edu (130.207.251.1)
277   gateway2-rtr.gatech.edu (130.207.254.117)
... ...
```

```
1  275    10.526 ms  1.208 ms   1.164 ms
2  276    3.545 ms   1.256 ms   2.167 ms
3  277    0.834 ms   0.766 ms   1.403 ms
... ...
```

Fig. 1. Setting up a lookup table for routers.

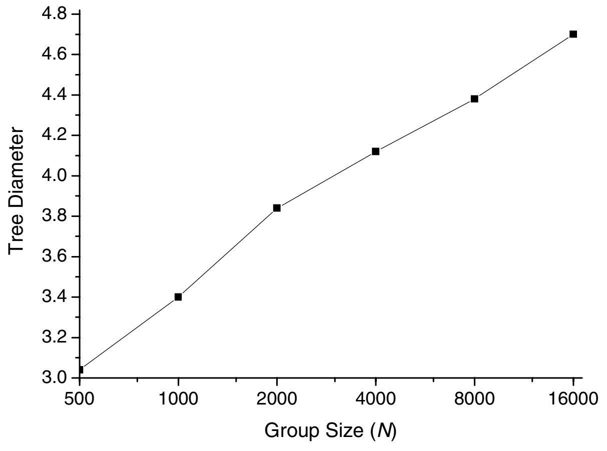


Fig. 2. Tree diameter versus group size.

PlanetLab [22].

A. Simulation Setup

We generate 5 *Transit-Stub* topologies with GT-ITM [23]. Each topology is a two-layer hierarchy of transit networks and stub networks, which contains 3200 routers and around $20,000$ links. We then randomly put N hosts into the network ($N = 500$ unless otherwise indicated). Each host is connected to a unique stub router with 1ms delay, while the delays of core links are given by the topology generator.

Furthermore, we randomly select 79 hosts from PlanetLab and conduct pair-wise traceroutes among them. Due to network and host dynamics (some hosts unexpectedly failed during our measurements), a small portion of the traceroutes cannot be completed. The resultant topology contains 5589 overlay paths (out of total $78 \times 79 = 6162$ ones), 1950 links, 946 known routers and some anonymous routers.

For a host in the system, the receiving of traceroute results from other hosts and the conducting of traceroutes are in parallel. In our simulations, for simplicity, we assume that data exchange between hosts is very quick and the conducting of one traceroute is considerably slow. In other words, we use similar settings as in Max-Delta, i.e., in each iteration, a host conducts one traceroute and receives one traceroute result from each of the other hosts. Given such settings, our scheme has the same measurement efficiency as Max-Delta. Since Max-Delta has been thoroughly evaluated in [6], [8], we do not repeatedly present similar results here and only present the new results. Note that these settings do not qualitatively affect the results below.

B. Results

Figure 2 shows the tree diameter versus the group size. The out-degree bounds of hosts are uniformly distributed within $[30, 100]$. As discussed above, data exchanged are of small size and hosts can have relatively large out-degree bounds. This is different from bandwidth-demanding applications such as streaming or file sharing, where a host can only have several neighbors. From the figure, we can see that the tree diameter is kept low. When $N = 16000$, the tree diameter is only 4.7. Therefore, data exchange among hosts can be quickly accomplished.

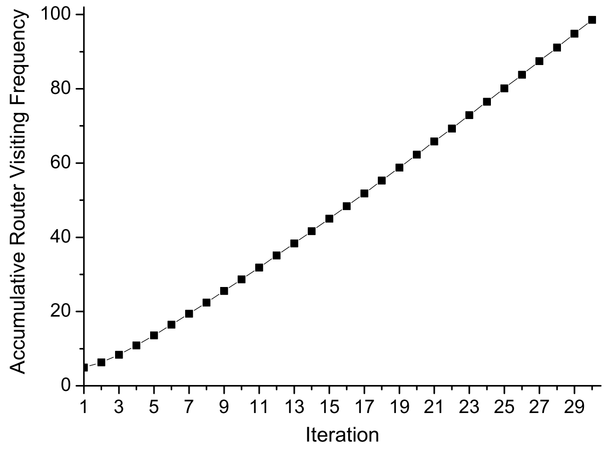
Figure 3(a) shows the accumulative router visiting frequencies achieved by Max-Delta. Router visiting frequency is defined as the number of occurring times of a router in a set of traceroute paths. From the figure, the router visiting frequency quickly increases with the iteration number. In the first iteration, the router visiting frequency is only 4.9. But after 30 iterations, the accumulative router visiting frequency increases to 98.5. It shows that a router appears many times in different traceroutes. On the other hand, in our PlanetLab measurements, there are totally 82212 known routers in the 5589 paths. However, there are only 946 different routers. It means that each router averagely appears $82212/946 = 86.9$ times. This confirms our simulation results. Therefore, a lookup table for routers is important.

Figure 3(b) shows the number of routers discovered in different inference iterations. The above line shows the number of routers reported in each iteration. In an iteration, each host usually traceroutes one path (in some case, a host may not be able to find a traceroute target and does not conduct any traceroute). In the first iteration, each host randomly selects a path to traceroute, and the total number of routers is 4899. In the next iteration, the total number of routers is only 1613. This is because Max-Delta preferentially selects a path with a large Δ value, which corresponds to a small $Euclidean$ distance and hence a short path. In other words, Max-Delta preferentially selects the shortest paths. Therefore, in the following iterations, we see an increase in the total number of routers reported. The line below in Fig. 3(b) shows the accumulative number of different routers in different iterations. After the first 4 iterations, over 99% routers have been discovered. Therefore, routers can be easily and quickly discovered. Since the number of different routers is much smaller than the total number of routers, a lookup table can significantly reduce the size of traceroute results.

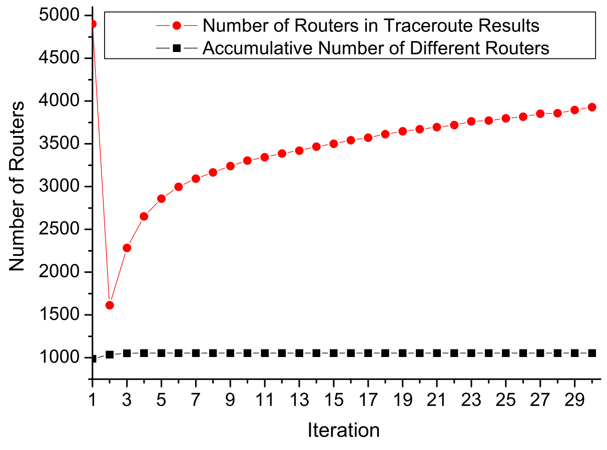
We compare the sizes of traceroute results received by a host with and without the lookup table in Fig. 3(c). The size of traceroute results is estimated as follows. The formats of raw traceroute results and compact traceroute results follow that in Fig. 1. In the PlanetLab measurements, we have 946 different routers. The representation of a router (including router name and router IP) averagely consists of 41.5 letters or digits. With a lookup table, each router can be represented by 4 digits (4 digits can represent at most $10,000$ routers). From the figure, the lookup table can averagely reduce the traceroute size by 50.2%. Clearly, this approach is efficient and effective.

V. Conclusion

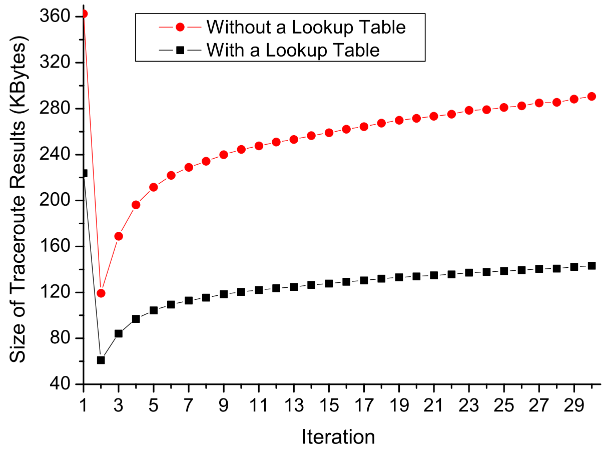
Max-Delta has been proposed to infer a highly accurate topology among a group of hosts with a low number of traceroutes. Considering that Max-Delta is centralized and not scalable, we propose a distributed scheme for scalable



(a) Accumulative router visiting frequency;



(b) Number of routers;



(c) Size of traceroute results;

Fig. 3. Reducing bandwidth consumption in inference ($N = 500$).

topology inference in this paper. In our scheme, hosts form an overlay tree to exchange traceroute results. A host can independently select paths to traceroute with no need of central scheduling. As compared to Max-Delta, our scheme distributes the computational loads for path selection to all the hosts, and only requires a host to exchange data with a few other hosts. Simulation results show that our data delivery tree has a low diameter, and that the proposed lookup table can significantly reduce bandwidth consumption in inference.

References

[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. ACM SOSP'01*, Oct. 2001, pp. 131–145.
[2] Y. H. Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE JSAC*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.
[3] 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.
[4] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *Proc. ACM NOSSDAV'02*, May 2002, pp. 127–136.
[5] J. Han, D. Watson, and F. Jahanian, "Topology aware overlay networks," in *Proc. IEEE INFOCOM'05*, March 2005, pp. 2554–2565.
[6] X. Jin, Y. Wang, and S.-H. G. Chan, "Fast overlay tree based on efficient end-to-end measurements," in *Proc. IEEE ICC'05*, May 2005, pp. 1319–1323.
[7] Traceroute. [Online]. Available: http://www.traceroute.org/
[8] X. Jin, W.-P. K. Yiu, S.-H. G. Chan, and Y. Wang, "Network topology inference based on end-to-end measurements," *IEEE JSAC*, vol. 24, no. 12, pp. 2182–2195, Dec. 2006.
[9] S. Y. Shi, J. S. Turner, and M. Waldvogel, "Dimensioning server access bandwidth and multicast routing in overlay networks," in *Proc. ACM NOSSDAV'01*, 2001, pp. 83–91.
[10] S. Y. Shi and J. S. Turner, "Routing in overlay multicast networks," in *Proc. IEEE INFOCOM'02*, June 2002, pp. 1200–1208.
[11] 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'02*, 2002, pp. 11–20.
[12] M. Coates, A. Hero, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, May 2002.
[13] D. G. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan, "Topology inference from BGP routing dynamics," in *Proc. ACM SIGCOMM IMW'02*, Nov. 2002, pp. 243–248.
[14] F. Wang and L. Gao, "On inferring and characterizing Internet routing policies," in *Proc. ACM SIGCOMM IMC'03*, Oct. 2003, pp. 15–26.
[15] Skitter. [Online]. Available: http://www.caida.org/tools/measurement/skitter/
[16] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *Proc. IEEE INFOCOM'00*, March 2000, pp. 1371–1380.
[17] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM'02*, Aug. 2002, pp. 133–145.
[18] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *Proc. ACM SIGMETRICS'05*, June 2005, pp. 327–338.
[19] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Deployment of an algorithm for large-scale topology discovery," *IEEE JSAC*, vol. 24, no. 12, pp. 2210–2220, Dec. 2006.
[20] T. S. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proc. IEEE INFOCOM'02*, June 2002, pp. 170–179.
[21] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. ACM SIGCOMM'04*, Aug. 2004, pp. 15–26.
[22] PlanetLab. [Online]. Available: http://www.planet-lab.org
[23] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM'96*, March 1996, pp. 594–602.