This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Improving the Efficiency of End-to-End Network Topology Inference

Xing Jin W.-P. Ken Yiu 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, kenyiu, gchan}@cse.ust.hk

***Abstract*— We consider inferring the underlay topology among a group of hosts by traceroute-like end-to-end measurement tools. 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, there is still high measurement redundancy in Max-Delta. That is, a router may be repeatedly visited in different traceroutes. In this paper, we integrate a previously proposed Doubletree algorithm into Max-Delta to reduce such redundancy. We study two key issues in the integration, i.e., the selection of h (a parameter of Doubletree) and the distribution of the global stop set of Doubletree. We have conducted extensive simulations on Internet-like topologies to evaluate the proposed scheme. The results show that Doubletree can significantly reduce the measurement redundancy and the bandwidth consumption in Max-Delta while introducing a small penalty in the measurement accuracy.**

I. Introduction

With the rapid growth of the Internet, 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. Note that this problem is different from most of the previous works on Internet measurements such as Skitter, Mercator and Rocketfuel [8]–[10]. All these works focus on Internet-level or ISP-level topology inference. Their measurement results often involve hundreds of thousands of routers and links, and their major concern is how to discover a *complete* network topology including all the routers and links. On the contrary, we are only interested in the topology among a certain group of hosts, where the group size is often on the scale of hundred or thousand. Our target is to achieve good tradeoff between the topology accuracy and the measurement cost.

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

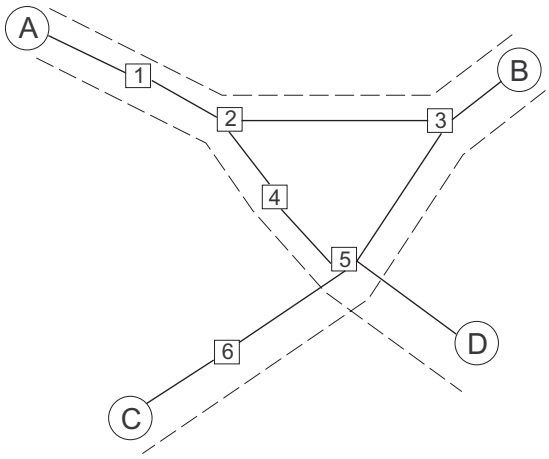


Fig. 1. Measurement redundancy in Max-Delta.

Max-Delta has been proposed to efficiently infer the underlay topology among a group of hosts [6], [11]. It uses a server to collect traceroute results from hosts and to select representative paths for hosts to traceroute. Max-Delta has been shown to be able to infer a highly accurate topology with a few traceroutes. However, there is still high measurement redundancy in its traceroute results. That is, a router may be visited multiple times in different traceroutes. We show an example in Fig. 1, where A, B, C, D are hosts and $1-6$ are routers. The dashed lines indicate the overlay paths among hosts. Suppose that Max-Delta selects paths $A-B$, $B-C$ and $A-D$ to traceroute. It can hence discover all the underlay links by only three traceroutes and there is no need to further traceroute paths $A-C$, $B-D$ or $C-D$. However, in such

1-4244-0353-7/07/$25.00 ©2007 IEEE

Max-Delta traceroutes, routers 1, 2, 3 and 5 have all been visited twice. The measurement redundancy hence exists. As shown in our simulations later, this redundancy is significantly high in large-scale traceroutes.

To reduce such redundancy, we integrate the Doubletree algorithm [12]–[14] into Max-Delta. In Doubletree, a traceroute starts at some intermediate router between the source and the destination. 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. For example, in Fig. 1, suppose that paths $A-B$ and $B-C$ have been measured. When measuring path $A-D$ (assuming that A is the traceroute source and D is the destination), the traceroute starts at some router between A and D, say, router 4. A bidirectional probing then starts: (1) The probing proceeds backwards towards the source A. The next router discovered is 2. Since router 2 has been discovered when measuring path $A-B$, the backward probing stops and does not continue probing router 1. Therefore, router 1 is probed only once in the whole inference process, instead of twice as in Max-Delta. (2) The probing also proceeds towards the destination D. The next router discovered is 5. Similarly, as router 5 has been discovered before, the forward probing then stops.

In this paper, we study two key issues in the integration of Max-Delta and Doubletree. The first one is how to select h, a parameter of Doubletree. h is the number of underlay hops between the starting point of the traceroute and the traceroute source. It determines the efficiency of redundancy reduction. We propose to use the partially discovered topology to select h. The second issue is that Doubletree needs to distribute a global stop set among the hosts, which introduces more network traffic. We hence compare the bandwidth consumption in Max-Delta and the integrated scheme. Our simulation results show that the integration of Doubletree can significantly reduce the measurement redundancy and the bandwidth consumption in Max-Delta while introducing a small penalty in the accuracy.

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 Max-Delta and Doubletree schemes. In Section III we discuss the integration of Max-Delta and Doubletree. In Section IV we present illustrative simulation results on Internet-like topologies. We finally conclude in Section V.

II. Review on Max-Delta and Doubletree

A. Max-Delta

Max-Delta has been proposed to efficiently infer the underlay topology among a group of hosts [6], [11]. In the scheme, hosts utilize light-weight tools such as GNP [15] or Vivaldi [16] 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, there is high measurement redundancy in Max-Delta inference. We hence integrate Doubletree into Max-Delta to reduce such redundancy.

B. Doubletree

Donnet et al. note that large-scale traceroute measurements have high intra- and inter-monitor redundancies [12]–[14]. 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 destination also have overlaps. They then propose a Doubletree algorithm to reduce such redundancies. Given a source and a destination, the traceroute starts at some intermediate router between them. The starting point is h hops away from the source, where h is a system parameter. Clearly, in a naive traceroute, h is equal to 1. The probing then proceeds towards the destination and backwards towards the source. Both the backward and forward probings use stop sets. The one for backward probing, called the *local stop set*, consists of all the routers that have been discovered by the source. The forward probing uses the *global stop set*, which consists of all the $(router, destination)$ pairs from all the sources. A pair enters the global stop set if a source visited the router while tracerouting the corresponding destination. In either case, the probing stops whenever a member of the stop set is met.

In this paper, we integrate the Doubletree algorithm into Max-Delta to reduce the measurement redundancy. Namely, after destination selection by Max-Delta, a traceroute starts and stops under Doubletree's supervision. We study two key issues in the integration. Firstly, the parameter h determines the tradeoff between reducing intra- and inter-monitor redundancies, and hence the overall efficiency of redundancy reduction. Donnet et al. have proposed some instructions for selecting h in [13]. We apply the instructions in the integrated scheme to select a proper h. Secondly, in Doubletree, the global stop set is shared among all the monitors. When the monitors are few (e.g., 24 monitors in Doubletree experiments), the overhead for sharing is not large. But in Max-Delta, each host is a

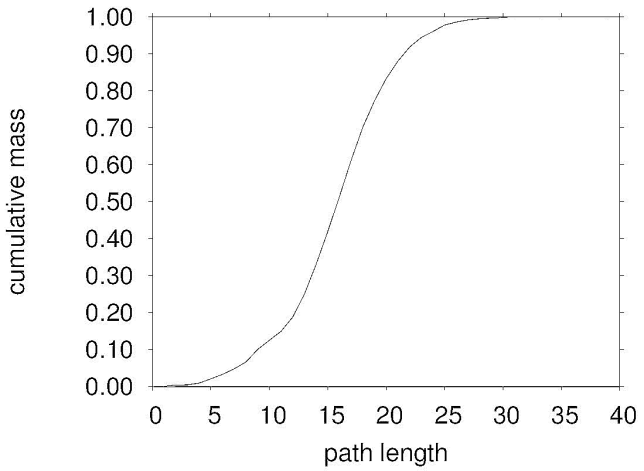


Fig. 2. Length of paths for a Skitter monitor (from [13]).

monitor. Sharing of the global stop set among all the hosts (e.g., hundreds of hosts) may be costly. We hence endeavor to reduce the bandwidth consumption for such sharing and further quantitatively study the bandwidth consumption of the integrated scheme.

III. Reducing Measurement Redundancy

We integrate Doubletree into Max-Delta to reduce the measurement redundancy in traceroutes. The integrated scheme works as follows. In each iteration, each host needs to traceroute one path. After it starts and completes the traceroute under Doubletree's supervision, the host reports the traceroute result to the server. The server then combines all the traceroute results and selects traceroute targets in the next iteration. The server also distributes the newly discovered members in the global stop set to all the hosts. We study two key issues in the integration, i.e., the selection of h and the sharing of the global stop set.

A. Selecting Proper h

The selection of the probing distance h is crucial. According to [13], a small h leads to high intra-monitor redundancy while a large h leads to high inter-monitor redundancy. In [13], each monitor sets its own value of h in terms of the probability p that the path between the monitor and a random destination is of no more than h hops. For example, Figure 2 shows the cumulative mass function of p for a Skitter monitor (Figure 5 in [13]). In order to restrict the probability that the first ICMP (Internet Control Message Protocol) message of a traceroute (with Time-to-Live, or TTL=h) hits the destination to a certain value, say, 10%, the monitor should start probing at $h = 10$ hops. In [13], the authors recommend to set p to 0.05. They also suggest to estimate the distribution of p by sampling a few randomly selected paths. However, this sampling method is not feasible here. In Max-Delta, only a very small sampling size is allowed, which may lead to poor estimation accuracy of p. This is because the number of traceroutes conducted by each host in Max-Delta is small (e.g., $10 - 14$ in a group of 256 hosts) and it is infeasible to have a sampling cost larger than the real measurement cost.

TABLE I

Link ratio achieved by Max-Delta in different iterations (%).

Iteration	1	3	5	7	9	11	13	15	17	19
$N = 256$	74.3	82.3	86.7	91.8	94.8	96.8	97.1	97.4	97.8	98.1
$N = 1024$	65.1	74.2	83.3	85.2	86.2	87.8	88.4	89.0	90.2	91.1

We note that Max-Delta can discover a majority of the underlay topology in its first iterations. Table I shows the link ratio (defined in Section IV) achieved by Max-Delta in the first iterations on Transit-Stub topologies. In a group of 256 hosts, 74.3% underlay links can be discovered after the first iteration of traceroutes. After 19 iterations, over 98.1% links are discovered. When $N = 1024$, 65.1% links are discovered after the first iteration. After 19 iterations, over 91.1% links are discovered. We hence estimate h for each host as follows. In the first iteration, each host randomly selects a target to traceroute as in Max-Delta, and reports the result to the server. Given a certain host, the server computes the paths between it and all the other hosts in the currently discovered topology (using shortest path routing). When the discovered topology contains the majority of the links on the underlay, a shortest path between two hosts in the discovered topology is supposed to approximate the actual path between them. Note that the partially discovered topology is not complete yet. Missing of inter-host links clearly leads to overestimation of path length in the discovered topology. We hence discard a certain number of paths with the most hops and use the rest results to compute the distribution of p. Our simulation results show that it is good to discard around $20\% - 30\%$ paths with the most hops.

B. Sharing the Global Stop Set

The distribution of the global stop set may incur heavy traffic. We need to quantitatively evaluate the resource consumption in the integration. We use *resource usage* to evaluate the network resource consumed for topology inference. Given a traffic between two points, the resource usage is computed as the size of packets delivered times the delay between the two points.

We first analyze the traffic generated by a traceroute. Traceroute is implemented with ICMP messages. Each time, the source sends an IP datagram with a certain TTL value to the destination. Each router that receives the datagram decrements the TTL by one and then continues forwarding the datagram. When a router receives an IP datagram whose TTL is 1, it throws away the datagram and returns an ICMP "time exceeded" error message to the source. The message contains the router's name, IP address and round-trip time to the source. In another case, if the datagram arrives at the destination with an unused port number (usually larger than $30,000$), the destination host returns an ICMP "port unreachable" error message to the source host. Therefore, in traceroute, the source sends a series of IP datagrams with increasing TTL to the destination and each datagram can identify one router in

TABLE II
RESOURCE USAGE FOR DISTRIBUTING ONE BYTE DATA AMONG 500 HOSTS (UNIT: $byte \times ms$).

Topology	1	2	3	4	5
Direct Access to Server (A)	172096	192935	190848	153270	188308
20-ary tree (B)	180170	193461	204809	168332	195338
Relative gap= $100(B-A)/A$	4.69	0.27	7.32	9.83	3.73

the path. The whole router-level path is then identified. An outgoing UDP datagram in traceroute contains 12 bytes of user data, 8 bytes of UDP header, 20 bytes of IP header for a total of 40 bytes. The size of the returned datagram changes. The returned ICMP message contains 20 bytes of IP header, 8 bytes of ICMP header, 20 bytes of IP header of the datagram that caused the error, 8 bytes of UDP header for a total of 56 bytes. For each TTL value, three ICMP messages are sent. We hence can accordingly compute the resource usage consumed by a traceroute.

We now analyze the distribution process of the global stop set. We slightly modify the format of the stop set in our simulations. In the global stop set, each member is a router IP instead of a $(router, destination)$ pair. Clearly, this reduces the size of the global stop set. For a small group of hosts, each host can directly exchange data with the server. Namely, the server distributes the global stop set among the hosts without overlay relaying. This is also the case for Max-Delta, where a host directly reports its traceroute result to the server and obtains the next traceroute target from the server. When the group size is large, the server may not be able to simultaneously set up so many connections. In this case, we can build an overlay tree to aggregate traceroute results for the server and to distribute data from the server. There are many ways to build overlay trees [6], [17], [18]. For simplicity, in our simulations we build a complete k-ary overlay tree. Table II shows the resource usage for distributing one byte data among 500 hosts. We use five different Transit-Stub topologies, each with 3200 routers and around 20000 links. On each topology, we run the simulations for 20 times and show the average over the results. From the table, the above two approaches have similar resource usage. The relative gap between them is always below 10%. Therefore, in the simulations in Section IV we only evaluate the direct server-access approach.

IV. ILLUSTRATIVE NUMERICAL RESULTS

In this section we evaluate the proposed scheme through simulations on Internet-like topologies.

A. Simulation Setup

We generate a number of *Transit-Stub* topologies with GT-ITM [19]. Each topology is a two-layer hierarchy of transit networks and stub networks. We then randomly put 500 hosts into the network (i.e., $N = 500$). A host is connected to a unique stub router with 1ms delay, while the delays of core links are given by the topology generator. We use the following metrics to evaluate an inference scheme.

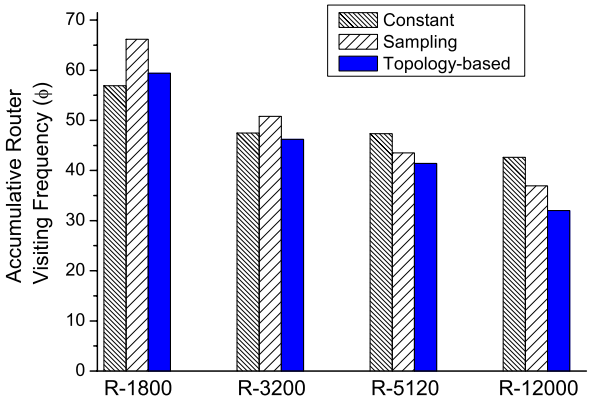
- *Router visiting frequency (ϕ)*, defined as the number of occurring times of a router in a set of traceroute paths.
- *Link ratio (β)*, defined as the ratio of the number of links in the inferred topology to the total number of links in the actual underlay topology [11].
- *Router ratio (γ)*, defined as the ratio of the number of routers in the inferred topology to the total number of routers in the actual underlay topology [11].
- *Resource usage*, defined in Section III-B.
- *ICMP-message reduction ratio (θ)*, defined as the ratio of the number of ICMP messages reduced by Doubletree to the total number of ICMP messages by normal traceroutes.

B. Results

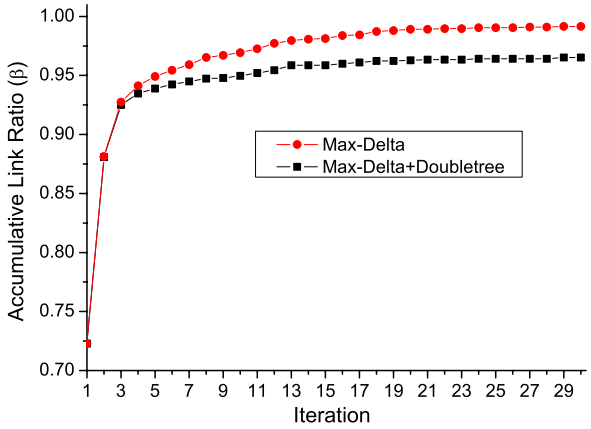
In the following, we call the scheme proposed in the paper an integrated scheme. We first evaluate the different mechanisms for selecting h. We generate four types of Transit-Stub topologies with different network sizes. The numbers of routers (or links) in the topologies are 1800, 3200, 5120 and 12000 (or around 9000, 20000, 32000 and 170000), respectively. In the following, these four types of topologies are denoted as $R-1800$, $R-3200$, $R-5120$ and $R-12000$, respectively. For each type, we generate 5 topologies. We then randomly put 500 hosts into the network and compute the average number of hops in the inter-host paths. After repeating the above process for 10 times, we get average numbers of hops on these four types of topologies as 7.5, 8.9, 9.7 and 14.2, respectively. We then test three mechanisms for selecting h. The first one sets h to a constant 4. The second one estimates h as in Doubletree, with a sampling on 20 paths. The last one estimates h as in Section III-A, with 25% paths with the most hops discarded. These three mechanisms are denoted as "*Constant*", "*Sampling*" and "*Topology-based*", respectively.

Figure 3(a) shows the accumulative router visiting frequencies at the 30th iteration achieved by different h-selection mechanisms. On $R-1800$ and $R-3200$, *Constant* has good performance. However, it does not perform well on $R-5120$ and $R-12000$. It shows that *Constant* is not flexible to different networks. *Sampling* always achieves higher ϕ than the *Topology-based* approach. Note that it also incurs additional sampling cost, which is not negligible as compared to the measurement cost. It is hence not a good choice. The *Topology-based* approach achieves good performance on all the four types of topologies. It does not incur any additional cost and is hence applicable for our inference scheme.

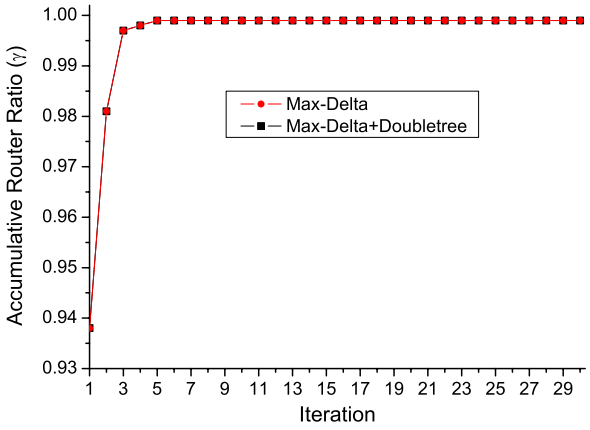
Figure 3(b)-(f) show the performance of Max-Delta and the integrated scheme on $R-3200$, with $N = 500$. Figure 3(b) shows the accumulative link ratios achieved by the schemes. Max-Delta performs slightly better than the integrated one. As the iteration number increases, β achieved by Max-Delta converges towards 0.992 while that of the integrated scheme converges towards 0.965. On the other hand, these two schemes achieve almost the same router ratio, as shown



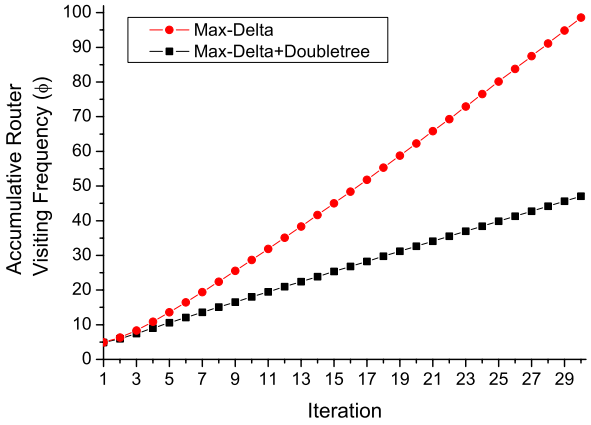
(a) Selection of h;



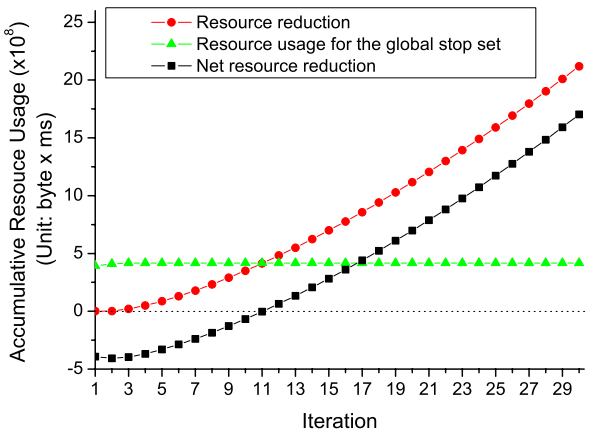
(b) Accumulative link ratio;



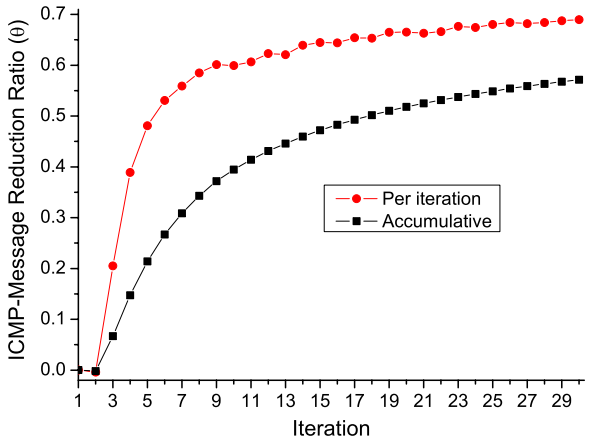
(c) Accumulative router ratio;



(d) Accumulative router visiting frequency;



(e) Accumulative resource usage;



(f) ICMP-message reduction ratio;

Fig. 3. Reducing measurement redundancy ($N = 500$).

in Fig. 3(c). These two figures show that the stop rules of Doubletree have slight impact to the measurement accuracy.

Figure 3(d) compares the accumulative router visiting frequencies achieved by the schemes. In Max-Delta, ϕ quickly increases with the iteration number. In the first iteration, ϕ is only 4.9. But after 30 iterations, the accumulative ϕ increases to 98.5. The integration of Doubletree can significantly reduce the router visiting frequency. Using the integrated scheme, the accumulative ϕ after 30 iterations is only 47.1, which is less than half of 98.5.

Figure 3(e) shows the resource usage in the integrated scheme. In particular, we show the reduction in the resource usage due to the reduction in ICMP messages and the resource usage for the sharing of the global stop set. The difference between these two values is the net resource reduction. We elaborate more on computing the resource usage for the global stop set. As shown in Fig. 3(c), over 99% routers are discovered in the first three iterations. In fact, in these three iterations, the numbers of routers discovered are around 980, 50 and 20, respectively. We estimate the size of the global stop set for the first iteration as follows. We randomly select 79 nodes from PlanetLab and conduct pair-wise traceroutes among them [20]. 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 $78\times79 = 6162$ ones), 1950 links, 946 known routers and some anonymous routers. From the result, we find that the average size of a router IP is around 12.5 bytes, and a compression over $600 - 900$ router IPs can achieve a compression ratio of $3 - 7$. We assume a compression ratio of 5 in our simulations. Therefore, a global stop set with M routers is of size $12.5 \times M/5 = 2.5M$ bytes. For the following iterations, we assume a smaller compression ratio of 2, as the numbers of newly discovered router IPs in these iterations are much smaller.

As Fig. 3(e) shows, the resource usage for the global stop set converges after three iterations. On the other hand, the resource reduction due to the reduction in ICMP messages keeps increasing as the iterations continue. This is because when the iterations continue, more and more routers are discovered and Doubletree stops more repeated probings. Clearly, the more complete topology being discovered, the more reduction being achieved. From the figure, starting from the 12th iteration, the net resource reduction becomes positive. Later on, more resource reduction can be achieved.

Figure 3(f) shows the ICMP-message reduction ratio in the integrated scheme. The curve denoted "Per iteration" shows the result in each individual iteration. Note that in the 2nd iteration, θ is negative. This is because the probing distance h is too large and the first ICMP message of a traceroute has reached the destination. In this case, more ICMP messages are sent out than that in a normal traceroute. In the following iterations, θ quickly increases to around 0.6. Therefore, significant reduction in the number of ICMP messages has been achieved. The curve denoted "Accumulative" shows the accumulative ICMP-message reduction ratio. After 30 iterations, we can achieve an overall 57% reduction in the number of ICMP messages. Clearly, the integration of Doubletree is efficient and effective.

V. Conclusion

We consider how to infer the underlay topology among a group of hosts through end-to-end measurement tools such as traceroute. Max-Delta has been proposed to infer a highly accurate topology among hosts with a few traceroutes. However, there is still high measurement redundancy in Max-Delta. In this paper, we integrate the Doubletree algorithm into Max-Delta to reduce such redundancy. The simulation results show that Doubletree can efficiently reduce the measurement redundancy and the resource consumption in the inference with a small penalty in the measurement accuracy. Such reduction of redundancy is important in large-scale network measurements.

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] Skitter. [Online]. Available: http://www.caida.org/tools/measurement/skitter/

[9] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *Proc. IEEE INFOCOM'00*, March 2000, pp. 1371–1380.

[10] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM'02*, Aug. 2002, pp. 133–145.

[11] 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.

[12] B. Donnet, T. Friedman, and M. Crovella, "Improved algorithms for network topology discovery," in *Proc. PAM'05*, March 2005.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM'02*, Aug. 2002, pp. 205–217.

[18] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," in *Proc. IEEE INFOCOM'02*, June 2002, pp. 1366–1375.

[19] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM'96*, March 1996, pp. 594–602.

[20] PlanetLab. [Online]. Available: http://www.planet-lab.org