The Theory of Network Tracing

H. B. Acharya
Department of Computer Science
University of Texas at Austin

M. G. Gouda
Department of Computer Science
University of Texas at Austin

Student author :
H. B. Acharya

Abstract

A widely used mechanism for computing the topology of any network in the Internet is Traceroute. Using Traceroute, one simply needs to choose any two nodes in a network and then obtain the sequence of nodes that occur between these two nodes, as specified by the routing tables in these nodes. Thus, each use of Traceroute in a network produces a trace of nodes that constitute a simple path in this network. In every trace that is produced by Traceroute, each node occurs either by its unique identifier or by the anonymous identifier "$*$". In this paper, we introduce the first theory aimed at answering the following important question. Is there an algorithm to compute the topology of a network N from a trace set T that is produced by using Traceroute in N, assuming that each edge in N occurs in at least one trace in T, and that each node in N occurs by its unique identifier in at least one trace in T? Our theory shows that the answer to this question is "No" in general. But if N is a tree, or is a ring with an odd number of nodes, then the answer is "Yes". On the other hand, if N is a ring with an even number of nodes, then the answer is "No", but if N is a "mostly regular" ring with an even number of nodes, then the answer is "Yes".

1 Introduction

Traceroute is arguably the most popular mechanism for computing the topology of a network in the Internet [3] and [13]. Each use of Traceroute between any two nodes, say nodes x and y, in a network produces a sequence, also called trace, of nodes that occur, in the network, between x and y as determined by the routing tables in the network nodes. It follows that each trace, that is produced from using Traceroute between nodes x and y, does correspond to a simple path between x and y in the network. On the other hand, if the network has multiple simple paths between nodes x and y, then only one of these paths corresponds to the trace that is produced from using Traceroute between x and y.

Traceroute can be used to compute the topology of a network N in the Internet as follows [3] :

1. Identify the nodes that occur at the perimeter of network N. We refer to these nodes as the terminal nodes of N.
2. Use Traceroute between any two terminal nodes of N to produce the trace of nodes that occur between these two terminal nodes (as determined by the routing tables in the nodes of N).
3. Put all traces, that are produced in Step 2, together in order to compute the topology of network N.

There are three problems that can hinder computing the correct topology of network N in Step 3. These three problems are as follows:

(a) *Incomplete coverage:* It is likely that the set of traces produced in Step 2 do not cover every edge or every node in the network. And when this happens, the computed topology in Step 3 will not be correct.

(b) *Aliasing of Node Identifiers:* A node in a network may have two or more (unique) identifiers and so such a node may occur by its different identifiers in different traces in the trace set produced in Step 2. This may cause the node to be regarded as multiple nodes and the computed topology in Step 3 to be incorrect.[7]

(c) *Node Anonymity:* If a node that occurs in a trace is busy, when this trace is being produced in Step 2, then the node may decide not to bother announcing its unique identifier in the produced trace. If this happens, then the node will occur in the trace by an anonymous identifier "$*_i$", where i is a positive integer, rather than by the node's unique identifier. This in turn may cause the computed topology in Step 3 to be incorrect.[12]

In this paper, we focus on the problem of node anonymity, and develop the first theory aimed at answering the question: "Is there an algorithm to compute the topology of a network from a given trace set of this network, even when some nodes occur by anonymous identifiers, rather by their unique identifiers, in some traces in the given trace set?" As we show below, our theory produces a number of surprising and unexpected answers to this question. (Note that node anonymity is a serious problem: the topology generated from the Internet by the iPlane tool, in [8], includes over 9M anonymous nodes (1M after the initial pruning) along with 230K known nodes.)

In order to develop our theory, we need first to dispose of the other two problems mentioned above: incomplete coverage and aliasing of node identifiers. We achieve this goal by basing our theory on the following two assumptions:

(i) *Unique Identifiers:* Each node in the network has exactly one unique identifier.

(ii) *Complete Coverage:* Each edge in the network occurs in at least one trace in the given trace set of the network. Also, each node in the network occurs by its unique identifier in at least one trace in the given trace set of the network (but the node may appear by an anonymous identifier in other traces in the given trace set).

Note that the assumption of complete coverage can be somewhat realized if one designates a large fraction of the nodes in the network (being traced) to be terminal nodes. (Recall Step 1 above.) To some extent, some missing links can be guessed by using the techniques in [9].

The problem of node anonymity in a given trace set has been discussed in an ad-hoc manner using unsubstantiated heuristics in [8], [10], and [14]. Similarly, the problem of aliasing of node identifiers has been discussed in an ad-hoc manner using unsubstantiated heuristics in [5], [6], and [11]. Previous work related to the subject matter of this paper is discussed in Section 7.

In the next section, we will define terms such as trace and trace set formally, so as to be able to develop our mathematical treatment of the problem.

2 Network Tracing

A network N is a connected, undirected graph where nodes have unique identifiers. Every node in a network is designated either *terminal* or *non-terminal*. Also, every node is either *regular* or *irregular*.

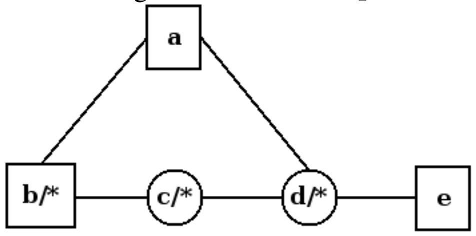
A *trace* t is *generable from* a network N iff t is a sequence of nodes in N that represents a simple path between two distinct terminal nodes in N. If a node that *occurs* in t is regular, then the node occurs in t by its unique identifier in t. Otherwise (if the node is irregular), the node occurs in t by an anonymous identifier $*_i$, where i is a unique integer in the trace. The first and last nodes of t must occur by their unique identifiers in t.

As an example, consider network N_1 in Figure 1. Note that, for convenience, we adopt the following notation in all our graphical representations of networks.

1. A terminal node is represented by a box.
2. A non-terminal node is represented by a circle.
3. A regular node x is labeled by its unique identifier "x".
4. An irregular node x is labeled by the label "x/*", where x is the unique identifier of the node.

Consider the following node sequences taken from network N_1 in Figure 1.

Figure 1: Network N_1



- The node sequence $(a, *_4, c, *_5)$ is not a trace that is generable from network N_1 : one of the two terminal nodes in the sequence does not occur by its unique identifier.
- The node sequence $(a, b, *_1, *_1, e)$ is not a trace that is generable from network N_1 : a trace corresponds to a simple path, hence nodes may not repeat. In this sequence, the node $*_1$ occurs twice, hence there is a loop. This makes the sequence an invalid trace.
- The node sequence $(b, *_1, e)$ is not a trace that is generable from network N_1: this is because there is no path of length 2 connecting b and e in network N_1.
- The node sequence $(a, *_2, *_4, d)$ is not a trace that is generable from network N_1: one of the two terminal nodes in the sequence, node d, is not a terminal node in N_1.
- The node sequence $(a, *_1, *_2, *_3, e)$ is a trace that is generable from network N_1.

Note that a trace t that is generable from a network N is a sequence of nodes that corresponds to a simple path in N. Thus, there are two ways to write the sequence of nodes in t. For example, t can be written as $(e, *_1, *_2, *_3, a)$, or it can be written as $(a, *_3, *_2, *_1, e)$. We regard the differences between these two ways of writing t as immaterial. Later on, when we mention that a trace is of the form $(x, \ldots, y)$, we mean that the trace could also be of the form $(y, \ldots, x)$.

For a trace t, we adopt the notation $|t|$ to indicate the number of edges in trace t. For example, if $t = (a, *_3, *_2, *_1, e)$, then $|t| = 4$.

A *trace set* T is *generable from* a network N iff T satisfies the following five conditions :

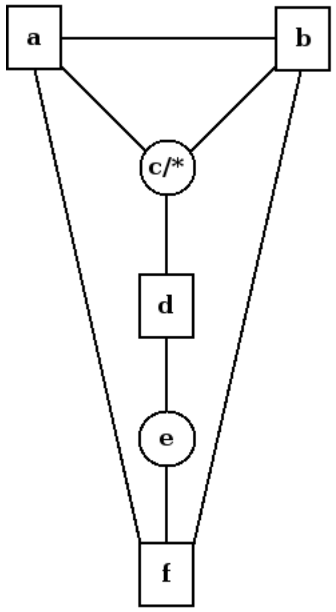
1. T is a set of traces, each of which is generable from N.
2. For every pair of terminal nodes x, y in N, T has exactly one trace between x and y.
3. Every edge in N occurs in at least one trace in T.
4. The unique identifier of every node in N occurs in at least one trace in T.
5. T is *consistent*: for every two distinct nodes x and y, if x and y occur in two or more traces in T, then the exact same set of nodes must occur between x and y in every trace in T where both x and y occur.

Two comments concerning condition 5 in this definition, the consistency of T, are in order. First, if a trace set T has two traces of the form $(x, *_2, z)$ and (u, x, y, z), then from the consistency condition, we can conclude that node $*_2$ is in fact node y.

Second, if a trace set T has a trace of the form $(x, *_2, z)$, then from the consistency condition, T cannot have a trace of the form $(u, x, *_5, y, z)$. This is because the number of nodes between x and z in the first trace is 1, and their number in the second trace is 2, in violation of the consistency condition.

The *network tracing problem* is to design an algorithm that takes as input a trace set T that is generable from a network, and produces a network N such that T is generable from N and not from any other network.

Figure 2: Network N_2



3 Impossibility of Network Tracing

In this section, we show that the network tracing problem is not solvable for general networks.

Theorem 1. *There is no algorithm that takes as an input a trace set T that is generable from a network, and produces as output a network N such that:*

- *T is generable from N, and*
- *T is not generable from any other network.*

Proof. (By contradiction) Assume that such an algorithm exists. The following trace set T is generable from network N_2 in Figure 2.

$$\begin{aligned}&\{(a,b),(a,*_1,d),(a,f),\\&(b,c,d),(b,f),\\&(d,e,f)\}\end{aligned}$$

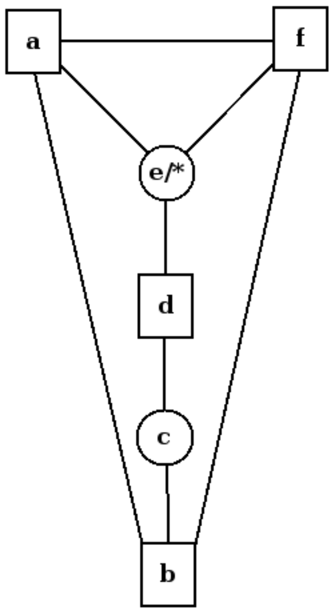
If T is given as an input to this algorithm, the algorithm will compute network N_2 in Figure 2 as the output. This implies that T is not generable from any other network, which contradicts the fact that T is also generable from network N_3 in Figure 3.

□

Theorem 1 shows that the network tracing problem is not solvable for general networks. However, as shown below, the problem is solvable for special classes of networks. One such class of networks is the class of regular networks defined next.

A network N is called *regular* iff every node in N is regular. The next theorem states that the network tracing problem is solvable for regular networks.

Figure 3: Network N_3



Theorem 2. *There is an algorithm that takes as an input a trace set T that is generable from a regular network, and produces as output a regular network N such that:*

- *T is generable from N, and*
- *T is not generable from any other regular network.*

Proof. The proof of this theorem follows from the fact that every node occurs by its unique identifier in every trace in a trace set that is generable from a regular network. □

In the next two sections, we consider two other special classes of networks, trees and rings, and discuss whether the network tracing problem is solvable for these two classes.

4 Tracing of Tree Networks

A network N is called a *tree* if N is acyclic. In this section, we show that the network tracing problem is solvable for tree networks.

Theorem 3. *There is an algorithm that takes as an input a trace set T that is generable from a tree network, and produces as output a tree network N such that:*

- *T is generable from N, and*
- *T is not generable from any other tree network.*

Proof. (By construction) We prove Theorem 3 by describing the algorithm that is mentioned in the theorem. The algorithm consists of the following eight steps:

1. Initially, tree N is empty.

2. Apply procedure *Leaf*, discussed below, to compute, from T, the unique identifier of each leaf node in N.

3. Apply procedure *Parent*, discussed below, to compute from T, the unique identifier of the parent of each leaf node in N.

4. For every node y that is the parent of a leaf node x, add to tree N an (undirected) edge between nodes x and y.

5. For every node y that is the parent of a leaf node x, replace in T each trace of the form $(x, *_i, \ldots)$ by a trace of the form $(x, y, \ldots)$.

6. Shorten the traces in T by replacing in T each trace of the form $(x, y, \ldots)$, where x is a leaf node, by the trace $(y, \ldots)$ and by discarding from T each trace that has only one node or is empty.

7. Repeat the algorithm, starting from Step 2, on the trace set T, that results from Step 6, provided that the resulting set T is non-empty.

8. The algorithm outputs N and terminates when the resulting T from Step 6 is empty.

Next, we specify the two procedures *Leaf* and *Parent* that are used in Steps 2 and 3, respectively, of the above algorithm.

The correctness of procedure *Leaf* follows from the observation that each leaf node in N occurs as a terminal node in some trace in T, but the converse is not necessarily true. Procedure *Leaf* is specified as follows:

```
procedure Leaf
    for each terminal node $y$ in any trace in $T$
        if $T$ has three traces $t = (x, \ldots, y), t' = (y, \ldots, z), t'' = (x, \ldots, z)$, such that $|t| + |t'| = |t''|$
        then $y$ is a non-leaf node in $N$
        else $y$ is a leaf node in $N$
    end
```

The correctness of procedure *Parent* follows from the observation that the parent of each leaf node in N must occur by its unique identifier in some trace in T. Procedure *Parent* is specified as follows:

```
procedure Parent
    for each leaf node $x$ in $N$,
        if $T$ has a trace of the form $(x, y \ldots)$, or $T$ has two traces of the form $(x, *_i, z)$ and $(z, y, \ldots)$
        then the unique identifier of the parent of node $x$ is $y$
    end
```

□

5 Tracing of Ring Networks

A network N is called an *odd ring* iff N consists of one cycle that has an odd number of nodes. It is called an *even ring* if it consists of one cycle that has an even number of nodes. In this section, we discuss whether the network tracing problem is solvable for odd or even ring networks.

Theorem 4. *There is an algorithm that takes as an input a trace set T that is generable from an odd ring network, and produces as output an odd ring network N such that:*

- *T is generable from N, and*
- *T is not generable from any other odd ring network.*

Proof. (By construction) We prove Theorem 4 by describing the algorithm that is mentioned in the theorem. The algorithm consists of the following five steps:

1. Construct an unlabeled ring N with n nodes, where n is the number of unique identifiers that occur in the traces in T. The algorithm terminates when each node in N is labeled by a distinct unique identifier from those that occur in the traces in T.

2. Choose any trace $t = (a, \ldots, b)$ in T. Label any node in N with the unique identifier "a", and label the node in N, that is reachable by traversing $|t|$ edges clockwise starting from node a, with the unique identifier "b".

3. For every pair of traces $t' = (a, \ldots, c)$ and $t'' = (b, \ldots, c)$ in T,

   if $|t| = |t'| + |t''|$ or $|t'| = |t| + |t''|$

   then label the node in N, that is reachable by traversing $|t'|$ edges clockwise starting from node a, with the unique identifier "c".

   else label the node in N, that is reachable by traversing $|t'|$ edges counter-clockwise starting from node a, with the unique identifier "c".

4. Note that by the end of Step 3, every unique identifier of a terminal node in a trace in T is used to label one node in ring N.

5. Consider any trace $t' = (x, \ldots, y)$ in T, and note that $|t'|$ cannot be equal to $n/2$ since $|t'|$ is a positive integer, and n is odd. Consequently, one can determine whether any trace $t = (x, \ldots, y)$ goes, either clockwise or counter-clockwise, from node x to node y. Thus, if trace t has a unique identifier z that has not yet been used to label any node in N, then one can identify the node in N that should be labeled with z.

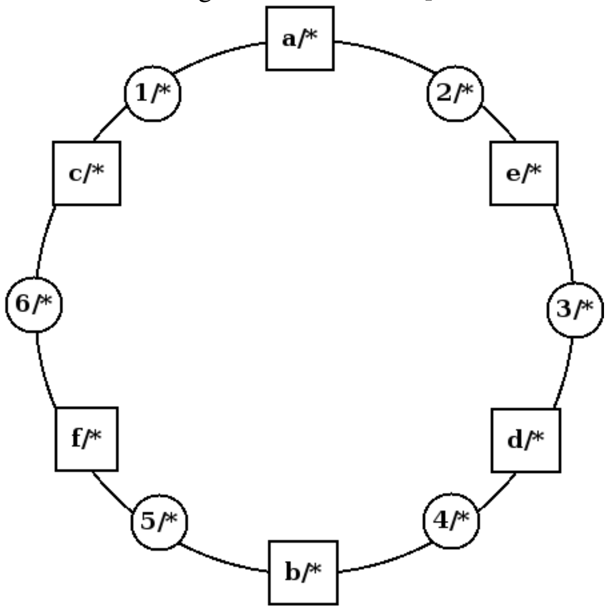
□

In the light of Theorem 4 (that the network tracing problem is solvable for odd ring networks), the next theorem (that the network tracing problem is not solvable for even ring networks) is some sort of a surprise.

Theorem 5. *There is no algorithm that takes as an input a trace set T that is generable from an even ring network, and produces as output an even ring network N such that:*

- *T is generable from N, and*
- *T is not generable from any other even ring network.*

Figure 4: Network N_4



Proof. (By contradiction) Assume that such an algorithm exists.The following trace set T is generable from network N_4 in Figure 4. (Note that each node in network N_4 is irregular. For convenience, the unique identifiers of the terminal nodes in N_4 are $a, b, c \ldots$ and the unique identifiers of the non-terminal nodes in N_4 are $1, 2, 3 \ldots$.)

$$\begin{aligned}&\{(a, 1, *_1, 6, *_2, *_3, b), (a, *_4, c), (a, *_5, *_6, *_7, d), (a, *_8, e), (a, *_9, *_{10}, *_{11}, f),\\ &(b, *_{12}, *_{13}, *_{14}, c), (b, *_{15}, d), (b, *_{16}, *_{17}, *_{18}, e), (b, *_{19}, f),\\ &(c, *_{20}, *_{21}, 2, *_{22}, 3, d), (c, *_{23}, *_{24}, *_{25}, e), (c, *_{26}, f),\\ &(d, *_{27}, e), (d, *_{28}, *_{29}, *_{30}, f),\\ &(e, *_{31}, *_{32}, 4, *_{33}, 5, f)\}\end{aligned}$$

If T is given as an input to this algorithm, the output produced is network N_4. This implies that T is not generable from any other even ring network. This contradicts the fact that T is generable from network N_5 in Figure 5. □

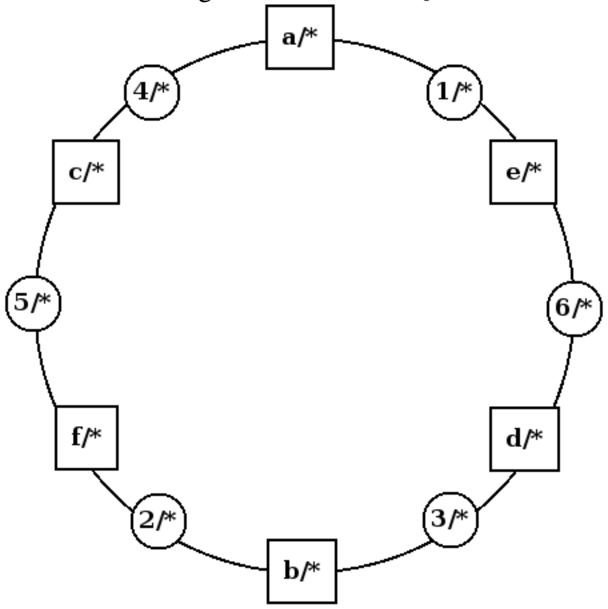
6 Tracing of Mostly-regular Networks

The network tracing problem for even rings can be solved given the additional constraint that each node (regular or irregular) has at most one irregular neighbor. We call a network, where each node has at most one irregular neighbor, a *mostly-regular network.*

Theorem 6. *There is an algorithm that takes as an input a trace set T that is generable from a mostly-regular even ring network, and produces as output a mostly-regular even ring network N such that:*

- *T is generable from N, and*
- *T is not generable from any other mostly-regular even ring network.*

Figure 5: Network N_5



Proof. It is sufficient to identify both neighbors of each node to specify the ring completely.

Assume a trace $(a, \ldots, w/*, *, y, z, \ldots, b)$

The anonymous node is x. both its neighbors are not irregular, as the ring is guaranteed to be mostly-regular.

Now if x is non-terminal and y is regular, all traces with x also have y, so there must be a trace of form $(\ldots, x, y, \ldots)$.

If x is a terminal and y is regular, then in the traces where it is a terminal x is not anonymous. Some trace passing through x must cover the edge $x - y$; this trace is to b. By consistent routing, when a trace is obtained from x to b, it must have the form $x, y, \ldots, b$.

Hence the regular neighbor of x can always be identified. further, both neighbors of a regular node y can always be identified - the case where both are regular is obvious; even when a node is irregular, its position next to a regular node can be identified, as above.

Now as x is irregular, the other neighbor of y, ie z, must be regular. The other neighbor of z, say z_1 (whether regular or irregular) is also known to be the neighbor of z (as above : the regular neighbor of an irregular node can be identified).

By similar reasoning to the case of x (ie, w is terminal/non-terminal) w is identified as being 2 hops from y. w and z_1 are the only two nodes with this description, and the position of z_1 is known. Hence the position of w is the only other one.

The irregular neighbor of x can always be identified. Hence in all cases, both neighbors of any node can be identified. The topology can be reconstructed. $\square$

Encouraged by Theorem 6, one may have hoped that the network tracing problem is solvable for the whole class of mostly-regular networks. Unfortunately, as shown by the next theorem, this turns out not to be the case.

Theorem 7. *There is no algorithm that takes as an input a trace set T that is generable from any mostly-regular network, and produces as output a mostly-regular network N such that:*

- *T is generable from N, and*

- *T is not generable from any other mostly-regular network.*

Proof. (By contradiction) Assume that such an algorithm exists. The following trace set T is generable from the mostly-regular network N_2 in Figure 2.

$$\begin{aligned}\{&(a,b),(a,*_1,d),(a,f),\\&(b,c,d),(b,f),\\&(d,e,f)\}\end{aligned}$$

If T is given as an input to this algorithm, the output produced is network N_2. This implies that T is not generable from any other mostly-regular network. This contradicts the fact that T is also generable from the mostly-regular network N_3 in Figure 3.

□

7 Discussion and Related Work

Anonymous router resolution is an inherent problem in traceroute based topology mapping studies. Most of the early work in the area ignores or circumvents the problem. In [3], authors avoid the problem by stopping a trace toward a destination on encountering an anonymous router on the path; this approach obviously discards useful information. In [2], authors handle anonymous routers by replacing them either with arcs connecting the known routers at two ends, or with unique identifiers to treat them as separate nodes, producing inaccurate maps. The "sandwich" approach used in [1], merges a chain of anonymous nodes, "sandwiched" between the same pair of known nodes, with each other - thereby losing resolution.

There have been three attacks on the anonymous router resolution problem. Yao et al. formulate it as an optimization problem [14]: building the smallest possible topology by combining anonymous nodes with each other under the constraints of trace preservation and distance preservation. They prove that the optimum topology inference under these conditions is NP-complete, then propose a heuristic to minimize the constructed topology by identifying anonymous nodes that, when merged, satisfy the two conditions. This is an $O(n^5)$ algorithm; also, its constraint of distance preservation states that the anonymous router resolution process should not reduce the length of the shortest path between any two nodes in the resulting topology map. This assumes, besides stable and symmetric routing, the additional constraint that routing is always along the shortest path.

Jin et al. propose two heuristics to address the problem in [10]. The first one, an ISOMAP based dimensionality reduction approach, uses link delays or node connectivity as attributes in the dimensionality reduction process. This is still a $O(n^3)$ algorithm; further, they ignore the difficulty of estimating individual link delays from round trip delays in path traces [4]. The second, a simple neighbor matching heuristic, is $O(n^2)$ but suffers from accuracy problems: it may introduce a high rates of both false positives and false negatives. Gunes et al. propose their own heuristics in [8] and show good performance, strictly better than $O(n^3)$ for five heuristics they apply in succession.

This paper addresses the problem and provides a theoretical basis for stating which instances of trace set can be used to compute exactly one network, and which cannot. We give a metric for reduction - the irregularity number - and bounds on algorithms such as in the above papers. We also give polynomial-time exact algorithms for several network cases of interest.

8 Concluding Remarks

We have made two contributions in this paper. First, we formally stated the network tracing problem. Second we identified network classes for which this problem is solvable and network classes for which the problem is unsolvable. In particular, we showed that the problem is solvable for the following network classes:

1. regular networks	(Theorem 2)
2. tree networks	(Theorem 3)
3. odd ring networks	(Theorem 4)
4. mostly regular even ring networks	(Theorem 6)

We also showed that the problem is not solvable for the following network classes:

1. general networks	(Theorem 1)
2. even ring networks	(Theorem 5)
3. mostly regular networks	(Theorem 7)

The research in this paper can be extended by weakening the network tracing problem and so making it solvable for many more classes of networks. As an example, a weak version of the network tracing problem can be stated as follows: "Is there an algorithm that takes as an input a trace set T that is generable from a network, and produces a "small" set $\{N_1, \ldots, N_k\}$ of networks such that

- T is generable from each network in set $\{N_1, \ldots, N_k\}$, and
- T is not generable from any network not in set $\{N_1, \ldots, N_k\}$?"

In fact, one can view the results of this paper as solving this weak network tracing problem when the value of k is 1. Solving this problem, when the value of k is 2 or 3 or ..., remains open and merits further research.

Finally, recall that our theory of network tracing has so far been based on the two assumptions of unique identifiers and complete coverage (as discussed in Section 1). It would be interesting to explore ways to relax these two assumptions while maintaining the effectiveness and elegance of the theory.

References

[1] S. Bilir, K. Sarac, and T. Korkmaz. Intersection characteristics of end-to-end internet paths and trees. *Network Protocols, 2005. ICNP 2005. 13th IEEE International Conference on*, pages 378–390, Nov. 2005.

[2] A. Broido and K. C. Claffy. Internet topology: connectivity of ip graphs. *Scalability and Traffic Control in IP Networks*, 4526(1):172–187, 2001.

[3] B. Cheswick, H. Burch, and S. Branigan. Mapping and visualizing the internet. In *ATEC '00: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 1–12, Berkeley, CA, USA, 2000. USENIX Association.

[4] D. Feldman and Y. Shavitt. Automatic large scale generation of internet pop level maps. *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, Dec 4 2008.

[5] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3:1371–1380, Mar 2000.

[6] M. Gunes and K. Sarac. Analytical ip alias resolution. *IEEE International Conference on Communications, 2006.*, 1:459–464, June 2006.

[7] M. Gunes and K. Sarac. Importance of ip alias resolution in sampling internet topologies. *IEEE Global Internet Symposium, 2007*, pages 19–24, May 2007.

[8] M. Gunes and K. Sarac. Resolving anonymous routers in internet topology measurement studies. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1076–1084, April 2008.

[9] M. H. Gunes and K. Sarac. Inferring subnets in router-level topology collection studies. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 203–208, New York, NY, USA, 2007. ACM.

[10] X. Jin, W.-P. K. Yiu, S.-H. G. Chan, and Y. Wang. Network topology inference based on end-to-end measurements. *Selected Areas in Communications, IEEE Journal on*, 24(12):2182–2195, Dec. 2006.

[11] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1):2–16, 2004.

[12] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. In search of path diversity in isp networks. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 313–318. ACM, 2003.

[13] F. Viger, B. Augustin, X. Cuvellier, C. Magnien, M. Latapy, T. Friedman, and R. Teixeira. Detection, understanding, and prevention of traceroute measurement artifacts. *Computer Networks*, 52(5):998 – 1018, 2008.

[14] B. Yao, R. Viswanathan, F. Chang, and D. Waddington. Topology inference in the presence of anonymous routers. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 1:353–363 vol.1, March-3 April 2003.