A Theory of Network Tracing

Hrishikesh B. Acharya[1] and Mohamed G. Gouda[1,2]

[1] The University of Texas at Austin, USA
[2] The National Science Foundation, USA
{acharya,gouda}@cs.utexas.edu

Abstract. Traceroute is a widely used program for computing the topology of any network in the Internet. Using Traceroute, one starts from a node and chooses any other node in the network. Traceroute obtains the sequence of nodes that occur between these two nodes, as specified by the routing tables in these nodes. 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 network 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? We prove that the answer to this question is "No" if N is an even ring or a general network. However, it is "Yes" if N is a tree or an odd ring. The answer is also "No" if N is mostly-regular, but "Yes" if N is a mostly-regular even ring.

1 Introduction

Traceroute is arguably the most popular mechanism for computing the topology of a network in the Internet [1] and [2]. Executing Traceroute between any two nodes, say nodes x and y, in a network produces a sequence of node identifiers that corresponds to a simple path between x and y in the network. This sequence of node identifiers is usually referred to as a *trace* between x and y.

Traceroute can be used to compute the topology of a network N in the Internet as follows [1] :

1. Identify the "terminal" nodes in network N (preferably at the perimeter of N for good coverage).
2. Execute Traceroute between every pair of terminal nodes of N, identified in Step 1, to produce traces of nodes that occur between each pair (as per the routing tables in the nodes of N).
3. Put the traces produced in Step 2 together in order to compute the topology of network N.

It turns out that this procedure for using Traceroute to compute the topology of network N has a problem. As observed in [3], [4], and [5], some of the nodes in the

R. Guerraoui and F. Petit (Eds.): SSS 2009, LNCS 5873, pp. 62–74, 2009.

traces produced in Step 2 occur by anonymous identifiers, rather than by their unique identifiers. This causes Step 3 to compute many candidate topologies, rather than one unique topology, for network N.

To solve this problem, Yao et al. [3] have suggested that Step 3 compute only the topology with the smallest number of anonymous nodes, subject to some constraints (trace preservation and distance preservation). This suggestion has two problems of its own. First, the choice, that the topology of network N be the one with the smallest number of anonymous nodes, is an arbitrary one. Second, it turns out that the problem of computing the network topology with the smallest number of anonymous nodes, from a given set of traces, is NP-complete. In order to solve this second problem, Jin et al. [4] and Gunes et al.[5] propose several heuristics (with complexity polynomial in the number of unique identifiers) that can be used to compute a network topology with a "small" number of anonymous nodes. Clearly, these heuristics cannot always compute a network topology with the smallest possible number of anonymous nodes.

In this paper, we take a different approach to the problem of computing one unique topology for network N from a given trace set T that is generated by executing Traceroute over N. Our approach is based on the assumption that the given trace set T satisfies a number of "conditions". The assumption, that the given trace set T satisfies these conditions, is made with the hope that the computed topology for network N is unique. These conditions can be summarized as follows: (Formal statements of these conditions are given in section 2).

- *Unique node identifiers:* Each node in network N has exactly one unique identifier, and if this node occurs in a trace in T, then it occurs in this trace either by this unique identifier, or by an anonymous identifier.
- *Complete coverage:* Each edge in network N occurs in at least one trace in the trace set T. Also, each node in N occurs by its unique identifier in at least one trace in T.
- *Stable and symmetric routing:* The routing tables in the nodes of network N indicate exactly one route between any two nodes in N.

These conditions may appear to be too strong to hold in practice. However, it is straightforward to show that if the given trace set T does not satisfy any one of these conditions, then more than one candidate topology for network N can be computed from T.

For example, assume that the given trace set T does not satisfy the first condition: a node occurs by identifier a in one trace in T, and occurs by a second identifier b in another trace in T. In this case, one can infer at least two candidate topologies for network N : in one topology, a and b indicate one node in N, and in the other, they indicate two distinct nodes in N.

Our main result in this paper is negative. This means that, even when the given trace set T satisfies the above (admittedly strong) conditions, it is not always possible to compute a unique topology for network N from T. Thus, adopting the above conditions has the effect of strengthening our primary (negative) results.

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 node identifiers that represents a simple path between two terminal nodes in N. A regular node occurs in t by its unique identifier. An irregular node occurs in t either by its unique identifier, or by the anonymous identifier $*_i$, where i is a unique integer in t. The first and last nodes of t occur by their unique identifiers in t.

We adopt the following notation in our graphical representations.

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 "$x/*$".

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 this trace could also be of the form $(y, \ldots, x)$.

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

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 at least one trace $(x, \ldots, y)$.
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 occur between x and y in every trace in T where both x and y occur.

Two comments concerning Condition 5 in this definition 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 Condition 5, 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 Condition 5, 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 Condition 5.

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.

3 Impossibility of Network Tracing

Obviously, the network tracing problem is solvable for *regular* networks, those where every node is regular. However, it turns out that the problem is *not* solvable for general networks. In fact, if a network is permitted to have just one irregular node, then the network tracing problem is unsolvable, as shown by the following theorem.

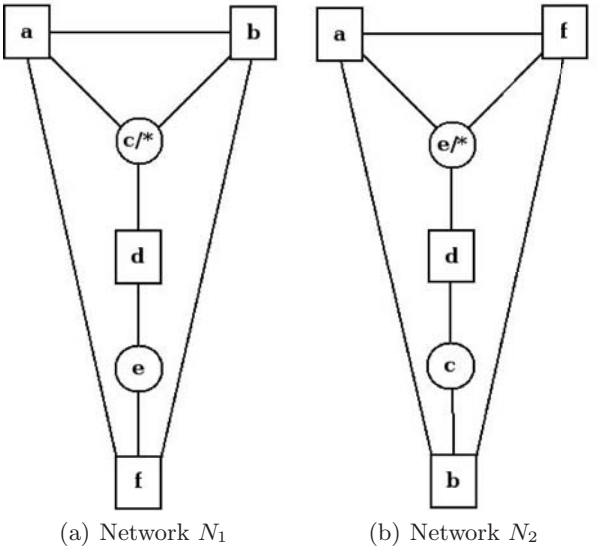
Theorem 1. *There is no algorithm that takes as an input a trace set T that is generable from a network with one irregular node, and produces as output a network N with one irregular node such that:*

- *T is generable from N, and*
- *T is not generable from any other network with at least one irregular node.*

Proof. (By contradiction) Assume that such an algorithm exists. The following trace set T_1 is generable from network N_1 in Figure 1.

$$\begin{aligned} T_1 = \{&(a, b), (a, *_1, d), (a, f), \\ &(b, c, d), (b, f), \\ &(d, e, f)\} \end{aligned}$$

If T_1 is given as an input to the assumed algorithm, the algorithm produces network N_1 as output. This implies that T_1 is not generable from any other network, which contradicts the fact that T_1 is also generable from network N_2 in Figure 1. □



(a) Network N_1 (b) Network N_2

Fig. 1. Theorem 1

Theorem 1 provides a strong negative result for the network tracing problem. Nevertheless, we identify, in the next three sections, classes of networks for which the network tracing problem is solvable:

1. Tree networks in Section 4.
2. Odd rings in Section 5.
3. Mostly regular even rings in Section 6.

4 Tracing of Tree Networks

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

Theorem 2. *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 2 by providing the algorithm mentioned. 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 occurs 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 $(z, y, \ldots)$
where z is a leaf node in N
then the unique identifier of the parent of node x is y
end

□

5 Tracing of Ring Networks

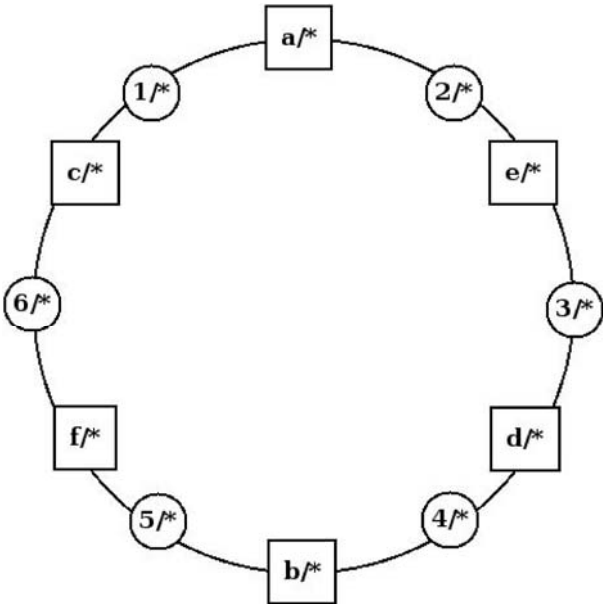
In this section, we discuss the solvability of the network tracing problem for ring networks. Surprisingly, we show that the problem is solvable for odd rings (i.e. cycles with an odd number of nodes), but not solvable for even rings (i.e. cycles with an even number of nodes).

Theorem 3. *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:*

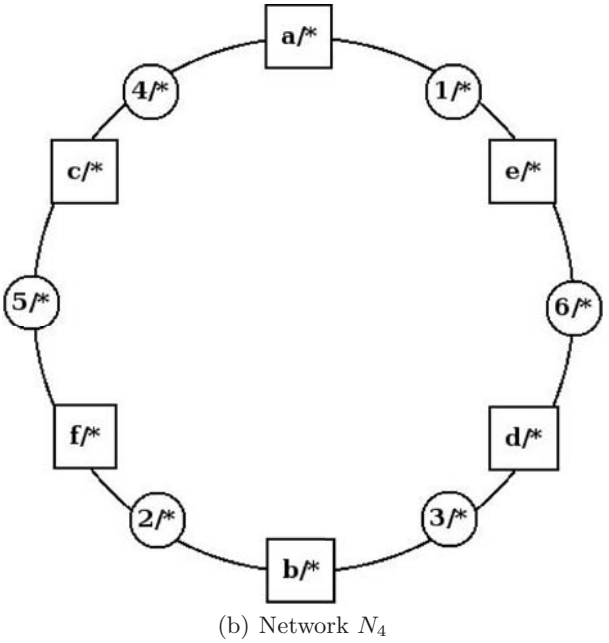
- *T is generable from N, and*
- *T is not generable from any other odd ring network.*

Proof. (By construction) We prove Theorem 3 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.



(a) Network N_3



(b) Network N_4

Fig. 2. Theorem 5

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 counterclockwise 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 counterclockwise, 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. □

Theorem 4. *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.*

Proof. (By contradiction) The proof proceeds as for Theorem 1, using the observation that the following trace set T_2 is generable from two even ring networks, N_3 and N_4, in Figure 2.

$$\begin{aligned} T_2 = \{&(a, 1, *_1, 6, *_2, *_3, b), (a, *_4, c), (a, *_5, *_6, *_7, d), (a, *_8, e), (a, *_9, *_{10}, *_{11}, f), \\ &(b, *_{12}, *_{13}, *_{14}, c), (b, *_{15}, d), (b, *_{16}, *_{17}, *_{18}, e), (b, *_{19}, f), \\ &(c, *_{20}, *_{21}, 2, *_{22}, 3, d), (c, *_{23}, *_{24}, *_{25}, e), (c, *_{26}, f), \\ &(d, *_{27}, e), (d, *_{28}, *_{29}, *_{30}, f), \\ &(e, *_{31}, *_{32}, 4, *_{33}, 5, f)\} \end{aligned}$$

□

6 Tracing of Mostly-Regular Networks

A network, where each node has at most one irregular neighbor, is called *mostly-regular*. The following theorem shows that the network tracing problem is solvable for mostly-regular even rings.

Theorem 5. *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.*

Proof. (By construction). The algorithm is given in our technical report [6]. □

Encouraged by Theorem 5, 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 6. *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) Suppose such an algorithm exists. As any network with exactly one irregular node is clearly mostly-regular, this algorithm takes any trace set generable from a network N with one irregular node, and returns N (and only N). This contradicts Theorem 1. □

7 The Weak Network Tracing Problem

The reason that the network tracing problem is not solvable in most cases, one may argue, is that the given trace set T is required to be generable from one, and only one, network N. One may hope, then, that if this strict requirement is somewhat relaxed, then the resulting weak version of the network tracing problem becomes solvable in many cases. The *weak network tracing problem* can be stated as follows:

"Design an algorithm that takes as input a trace set T, that is generable from a network, and produces a small set $\{N_1, .., N_k\}$ of networks such that T is generable from each network in this set and not from any network outside this set."

The requirement that the produced set $\{N_1, .., N_k\}$ be small means, mathematically, that the cardinality k of this set be a constant rather than a function of the number of unique node identifiers in the given trace set T.

There are both practical and theoretical reasons for this requirement. From a practical point of view, the smaller the produced set, the better. From a theoretical point of view, allowing the cardinality of the produced set to be a function of n, the number of unique identifiers, makes the weak network tracing problem trivially solvable (by exhaustive enumeration, setting each $*_i$ to each unique identifier).

Unfortunately, the following theorem shows that the weak network tracing problem is not solvable in general.

Theorem 7. *There is no algorithm that takes as an input a trace set T that is generable from a network, and computes a set of networks $\{N_1 \ldots N_k\}$ such that:*

- *T is generable from every network in the set $\{N_1 \ldots N_k\}$,*
- *T is not generable from any other network, and*
- *k is a constant whose value is not a function of the number of node identifiers in T.*

Proof. We prove this theorem by exhibiting an infinite sequence of trace sets TS_6, $TS_8, TS_{10} \ldots$ such that each trace set TS_n satisfies the following three conditions:

- TS_n has 1 anonymous node identifier.
- TS_n has n unique node identifiers.
- TS_n is generable from any one of $\frac{n-2}{2}$ distinct networks.

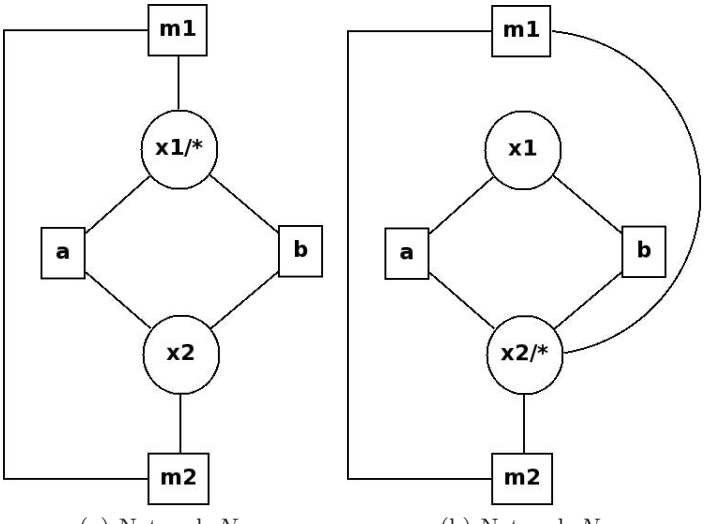
Consider the first trace set in the sequence.

$$\begin{aligned} TS_6 = \{&(a, x_1, b), (a, *_1, m_1), (a, x_2, m_2), \\ &(b, *_2, m_1), (b, x_2, m_2), \\ &(m_1, m_2)\} \end{aligned}$$

This trace set is generable from the two networks N_5 and N_6 in Figure 3.

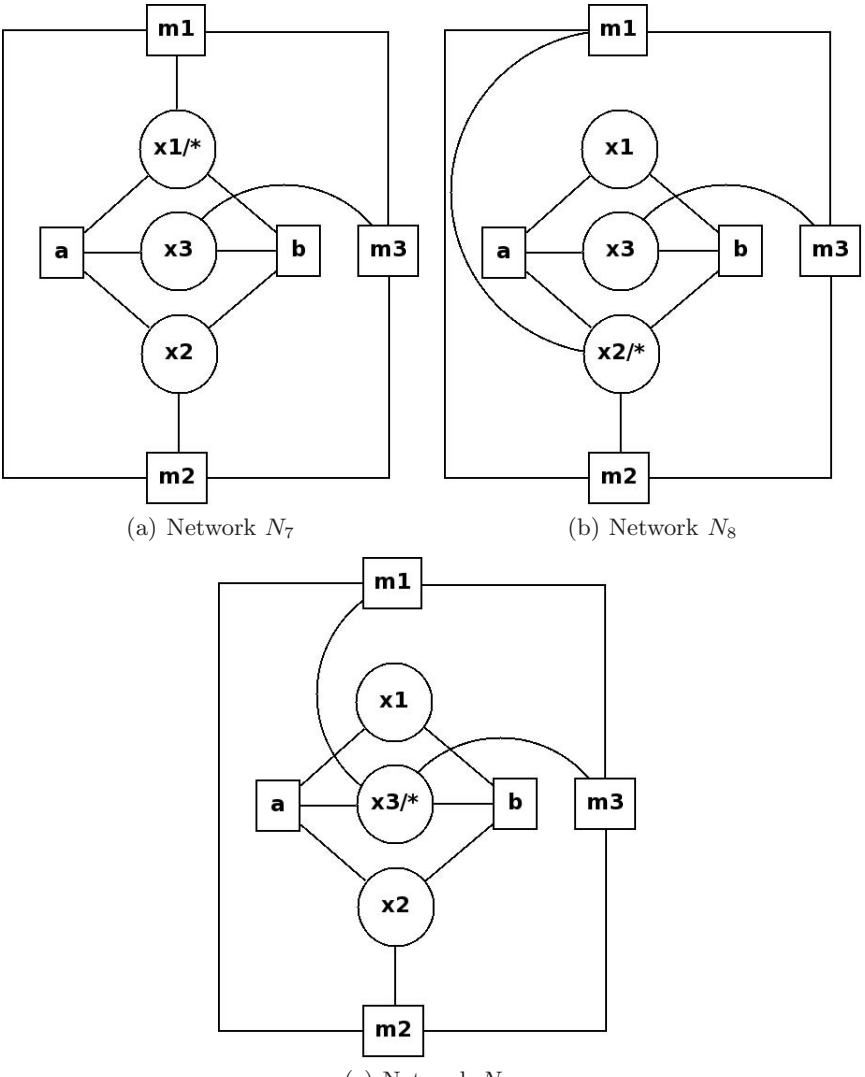
We now add two nodes, x_3 and m_3, to the trace set TS_6 to form the trace set TS_8 and increase the number of possible networks (from which the trace set TS_8 is generable) by 1. Node x_3 connects m_3 to both a and b, so we add the traces (a, x_3, m_3) and (b, x_3, m_3). Also, m_3 is directly connected to every m_i, so we add (m_1, m_3) and (m_2, m_3). The resulting trace set TS_8 is as follows:

$$\begin{aligned} TS_8 = \{&(a, x_1, b), (a, *_1, m_1), (a, x_2, m_2), (a, x_3, m_3), \\ &(b, *_2, m_1), (b, x_2, m_2), (b, x_3, m_3), \\ &(m_1, m_2), (m_1, m_3), (m_2, m_3)\} \end{aligned}$$



(a) Network N_5 (b) Network N_6

Fig. 3. Two networks



(a) Network N_7 (b) Network N_8

(c) Network N_9

Fig. 4. Three networks

The trace set TS_8 is generable from any of the three networks N_7, N_8 and N_9 in Figure 4.

By repeating this procedure $k-2$ times, we produce a trace set that is generable from every member of a set of k distinct networks. As each step adds two new nodes, each of these networks has $2k+2$ nodes. The number of unique identifiers being n, we have $k = \frac{n-2}{2}$. □

8 Discussion and Related Work

There have been three attacks on the anonymity problem. Casting it as an optimization problem, Yao et al. [3] try building the smallest possible topology by combining anonymous nodes. They consider two constraints, trace preservation and distance preservation. Proving that optimum topology inference under these conditions is NP-complete, they propose a $O(n^5)$ heuristic that merges anonymous nodes, keeping the constraints invariant. Distance preservation requires that merging nodes never reduces the length of the shortest path between any two nodes in the computed network; this assumes not only consistency, but also shortest-path routing. Their study also assumes that anonymous nodes are strictly anonymous, and their unique identifiers are never revealed.

Jin et al. propose two heuristics to address the problem in [4]. The first is $O(n^3)$, uses link delays or node connectivity as attributes, and performs ISOMAP-based dimensionality reduction. It ignores the difficulty of estimating one-hop delays from RTTs in path traces [7]. The second, a simple $O(n^2)$ neighbor matching heuristic, has high rates of both false positives and false negatives. In [5], Gunes et al. apply five heuristics in succession and get performance strictly better than $O(n^3)$.

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.

9 Concluding Remarks

We have made three contributions in this paper. First, we formally state the network tracing problem. We then develop the theory by identifying some important network classes for which this problem is solvable, and some for which it is not. This includes some very surprising results.

We then extend this research using a weaker version of the network tracing problem, and show that it is not only not possible in general to take a trace set T and compute a single network N, such that T is generable from N and only N, it is also not possible to generate a small (with constant cardinality) set of networks such that T is generable only from members of this set.

In future work, we intend to investigate whether it is possible to relax our assumptions (consistent routing, unique identifiers, and complete coverage) while maintaining the effectiveness and elegance of the theory.

References

1. Cheswick, B., Burch, H., Branigan, S.: Mapping and visualizing the internet. In: ATEC 2000: Proceedings of the annual conference on USENIX Annual Technical Conference, Berkeley, CA, USA, pp. 1–12. USENIX Association (2000)

2. Viger, F., Augustin, B., Cuvellier, X., Magnien, C., Latapy, M., Friedman, T., Teixeira, R.: Detection, understanding, and prevention of traceroute measurement artifacts. Computer Networks 52, 998–1018 (2008)
3. Yao, B., Viswanathan, R., Chang, F., Waddington, D.: Topology inference in the presence of anonymous routers. In: Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2003, vol. 1, pp. 353–363. IEEE, Los Alamitos (2003)
4. Jin, X., Yiu, W.P.K., Chan, S.H.G., Wang, Y.: Network topology inference based on end-to-end measurements. IEEE Journal on Selected Areas in Communications 24, 2182–2195 (2006)
5. Gunes, M., Sarac, K.: Resolving anonymous routers in internet topology measurement studies. In: INFOCOM 2008. The 27th Conference on Computer Communications, pp. 1076–1084. IEEE, Los Alamitos (2008)
6. Acharya, H.B., Gouda, M.G.: Tr-09-02: The theory of network tracing. Technical report, University of Texas, Austin (2009), `http://www.cs.utexas.edu/research/publications/ncstrl/ncstrl2html.cgi`
7. Feldman, D., Shavitt, Y.: Automatic large scale generation of internet pop level maps. In: IEEE Global Telecommunications Conference (GLOBECOM), pp. 1–6 (2008)