Network Topology Inference with Partial Information

Brett Holbert, Srikar Tati, Simone Silvestri, *Member, IEEE*, Thomas La Porta, *Fellow, IEEE* and Ananthram Swami, *Fellow, IEEE*

***Abstract*—Full knowledge of the routing topology of the Internet is useful for a multitude of network management tasks. However, the full topology is often not known and is instead estimated using topology inference algorithms. Many of these algorithms use Traceroute to probe paths and then use the collected information to infer the topology. We perform real experiments and show that in practice routers may severely disrupt the operation of Traceroute and cause it to only provide partial information. We propose iTop, an algorithm for inferring the network topology when only partial information is available. iTop constructs a virtual topology, which overestimates the number of network components, and then repeatedly merges links in this topology to resolve it towards the structure of the true network. We perform extensive simulations to compare iTop to state of the art inference algorithms. Results show that iTop significantly outperforms previous approaches and its inferred topologies are within 5% of the original networks for all considered metrics. Additionally, we show that the topologies inferred by iTop significantly improve the performance of fault localization algorithms when compared to other approaches.**

***Index Terms*—Topology inference, Partial information, Fault localization.**

I. Introduction

Knowledge of the routing topology of the Internet is of fundamental importance for a wide variety of network management tasks. For example, the design of overlay networks [1] and the localization of failures [2] can significantly benefit from accurate and thorough knowledge of the network topology [3], [4]. This information is often unavailable for various reasons such as unplanned changes, network dynamics and heterogeneous network ownership. As a result, the network topology must often be inferred by means of topology inference algorithms.

Previous work on topology inference is typically based on either *network tomography* or *Traceroute*. Both methods place special nodes called *monitors* at the edge of the network and use these monitors to probe the network. Network tomography approaches [3], [5], [6] are only able to infer a simplified representation of the actual network-layer topology [6]. Traceroute based techniques [7], [8], [9], [10], [11], [12], [13] use the Traceroute utility [14] to collect path information. Traceroute sends hop-limited packets from a source monitor to a destination in a network. The source collects responses from intermediate routers and constructs a Traceroute *trace*. The Network Operation Center (NOC) collects the traces from all monitors and uses this information to infer the topology. Traceroute based techniques are more effective in inferring the routing topology than network tomography approaches.

Most of the previous approaches based on Traceroute [7], [8], [9] assume that internal routers fully comply with the information collection process. However, in practice router configurations, privacy policies, and firewalls may prevent some routers from correctly cooperating, as we show through real experiments in this paper. We classify the behavior of internal routers with respect to Traceroute probes in three categories: *responding*, *anonymous* and *blocking* routers. Responding routers correctly participate in Traceroute operations. Anonymous routers do not send responses back to the source but do forward requests to other routers in the path. Blocking routers drop all Traceroute packets and do not send responses, preventing the collection of any further information about the path. We refer to blocking and anonymous routers collectively as *non-cooperative* routers.

Our experiments show that with the most common versions of Traceroute more than 10% of the routers in a path are anonymous. Furthermore, more than 30% of the Traceroute probes do not reach the destinations due to blocking routers, and according to recent studies this number can be as high as 90% [15]. As a result, most of the previously proposed topology inference algorithms based on Traceroute can fail in real scenarios, as they assume the full cooperation of internal routers.

Despite the severe impact of non-cooperative routers on the information gathered by Traceroute, only a few works consider the problem of topology inference with partial information [10], [11], [12], [13]. As our results show, even the best performing inference algorithms largely underestimate the negative effect of non-cooperative routers, and are not able to accurately infer the network topology.

In this paper we propose a novel approach called *iTop* to infer the network topology in the presence of both blocking and anonymous routers. The inference process of iTop is guided by a novel classification of the routers based on the partial information given by Traceroute. This classification is proposed to reduce the inherent ambiguity in the traces which prevents the identification of some routers. iTop operates in three phases.

B. Holbert, S. Tati and T. La Porta are with the Department of Electrical and Computer Engineering, Pennsylvania State University, University Park, PA. E-mail: {bdh5027, tati, tlp }@cse.psu.edu

S. Silvestri is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO. E-mail: silvestris@mst.edu

A. Swami is with the Army Research Laboratory, Adelphi, MD. E-mail: a.swami@ieee.org

This work was supported by the Defense Threat Reduction Agency under grant HDTRA1-10-1-0085.

Manuscript received December 12, 2014.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

In the first phase, iTop uses the partial path knowledge to construct an initial *Virtual Topology* where the gaps created by non-cooperative routers are filled with virtual routers and links. Routers in the virtual topology are classified depending on their behavior with regards to Traceroute. The virtual topology contains redundant elements with respect to the real topology. This redundancy is reduced by means of merging operations. These operations are executed on the basis of *merging options* calculated in the second phase, which are derived from consistency conditions and router classification. Finally, in the third phase iTop iteratively merges the links in the virtual topology according to the merging options provided by the previous phase.

We compare iTop to existing state-of-the-art algorithms for topology inferencing with partial path information [10], [11] using a thorough series of simulations on both realistic and random networks. Results show that for all of the considered metrics iTop significantly outperforms previous approaches. In particular, the topologies inferred by iTop are within 5% of the real topologies with respect to relevant metrics such as the number of nodes and edges, the degree and betweeness centrality distributions, and a recently proposed metric, the joint degree distribution [16].

We also consider fault localization as an application to evaluate and compare topology inference techniques. We propose a set of metrics to evaluate the output of fault localization algorithms when applied to an inferred topology. We use a popular fault localization algorithm called Max-Coverage [2]. Results show that iTop outperforms other topology inference approaches. It provides an accuracy close to the accuracy that would be achieved by knowing the real topology at the expense of a small increase in the number of false positives.

To summarize, our contributions are:

- We perform real experiments showing that the Traceroute tool is only able to collect partial information due to the presence of non-cooperative routers.
- We introduce iTop, a new approach designed for topology inference with partial information.
- We compare iTop to state-of-the-art inference algorithms through extensive evaluation. Results show that iTop significantly outperforms previous approaches and closely infers the real topologies for all the considered metrics.
- We propose metrics for evaluating fault localization algorithms when applied to inferred topologies. We demonstrate that these algorithms perform best when using the topologies inferred by iTop out of all considered approaches.

A preliminary version of iTop appeared in [17]. In this paper we provide the results of real experiments to motivate our work. In addition, we study the complexity of iTop, and we formally prove its advantages in the common scenario of hub topologies. Furthermore, we performed experiments in realistic and random networks, and we apply topology inference in the context of failure localization.

II. Motivating experiments with Traceroute

Traceroute [14] is a well-known tool for path discovery in IP-based networks. The tool sends hop-limited packets from a source towards a destination. Traceroute works in rounds, where in the k-th round packets are sent with a Time to Live (TTL) value of k. Intermediate routers on the path decrement the TTL and alert the source if the packet expires before reaching the destination. By collecting these error messages, the source discovers the routers on the path to the destination.

Three main variants of Traceroute are commonly used: ICMP-, UDP- and TCP-based. These differ in the type of exchanged packets [15]. ICMP Traceroute sends *ICMP Echo Request* packets towards the destination and intermediate routers reply with an *ICMP Time Exceeded* or *Time to Live Expired in Transit* packet when the TTL reaches zero. When the destination receives the packet it replies with an *ICMP Echo* packet. UDP- and TCP- based Traceroute work in a similar manner, but use different types of exchanged packets. UDP Traceroute transmits UDP packets to an invalid destination port value, while TCP Traceroute sends a TCP SYN packet to a well-known port, such as the default port of a web server.

Due to these differences, the three variants of Tracerouce possess varying capabilities to acquire path information to the destination. For example, intermediate routers can be configured to not reply to or completely discard ICMP packets. Furthermore, firewalls may filter packets to unknown ports or ports that do not belong to established TCP connections.

In this section we describe real experiments that we performed to show that all Traceroute variants only allow the collection of partial path information. Our results confirm the experimental analysis provided in [15].

A. Experimental setup and trace analysis

In order to characterize the gathered information, we classify network routers into three categories with respect to their behavior regarding Traceroute probes. *Responding* routers are routers that correctly take part in Traceroute and reply to the source as described above. *Anonymous* routers do not send a reply to the source when the TTL of a Traceroute probe expires, but do forward Traceroute requests and replies coming from other routers. Finally, *blocking* routers neither respond to the source as the TTL expires nor forward Traceroute probes and replies coming from other nodes.

The experiments were executed as follows. Traces were collected by using ICMP, UDP, and TCP Traceroute to probe the paths to a set of 100 destination websites from a source located on the Pennsylvania State University, University Park campus. For UDP and TCP Traceroute, traces were collected using the default destination port numbers. We also collected traces using other ports and observed similar results. The destinations were selected to include a wide variety of sites. In particular, we used well-known sites from various locations around the world such as *cnn.com, bbc.co.uk*, and *europa.eu*. The maximum path length was set to 30, so each individual trace is of the form $(x_1, x_2, \ldots, x_n)$ for $n \leq 30$, where each x_i contains either an identifier of the router located i hops from the source if a response was received or the indicator $*$ for no response otherwise.

If a response was received for router x_i then it is obviously a responding router. Any router x_j which did not provide a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

response is classified as anonymous if there is some responding router x_i for which $i > j$. A path contains a blocking router if there exists a x_k, such that for each $j \geq k$, x_j equals $*$. Note that we cannot conclude that x_k is blocking, as the same trace can be generated by a series of anonymous routers immediately preceding a blocking router.

Trace type	ICMP	UDP	TCP
Avg. number of responding routers	14	9.5	8
Percentage of anonymous routers	12%	2.6%	18.7%
Percentage of unreachable destinations	34%	90%	67%

Table I
SUMMARY OF THE EXPERIMENTAL RESULTS.

B. Experimental results

Table I shows the number of responding routers, the percentage of anonymous routers per path[1], and the percentage of unreachable destinations. ICMP Traceroute performs better than the other versions, as ICMP packets are not discarded as often as UDP and TCP packets. As a result, this version is able to identify more responding routers and reach more destinations. However, it should be noted that the percentage of unreachable destinations may be underestimated, due to some routers spoofing the source address of the ICMP response [15]. These results highlight that all versions are significantly affected by the presence of anonymous and blocking routers, as even with ICMP Traceroute 12% of routers in a path are anonymous and 34% of the destinations cannot be reached. As a result, a large portion of the topology remains unobserved. These results motivate the need for a topology inference algorithm that specifically takes into account the limitations of information provided by Traceroute.

III. NETWORK MODEL

We refer to the real topology as the *Ground Truth* (GT) topology. This is represented by an undirected graph $G_{\mathsf{GT}} = (V_{\mathsf{GT}}, E_{\mathsf{GT}})$, where V_{GT} is the set of nodes in the network and E_{GT} is the set of links connecting them. A set of nodes in V_{GT} is designated as monitors in the network. Such monitors are hosts external to the network and do not need to be owned, or run in cooperation with, the network providers. As a result, they are very inexpensive general purpose machines and do not incur an expensive infrastructure deployment. Similar to previous works on topology inference [10], [11], we assume that the routing algorithm uses shortest paths and that a single fixed path is used between each pair of monitors. Our approach can be easily extended to different routing strategies. We further assume that pairs of monitors have a mechanism to measure the hop distance of the path between them. For example, the hop distance can be measured using the TTL field of UDP packets exchanged between monitors. By setting the TTL to a large known value, monitors can measure how much it has decreased when the packet is received and calculate the number of hops traversed. These packets are not Traceroute packets and are exchanged between cooperative hosts on open ports. As a result, they are not discarded by intermediate routers

As previously described, we assume that anonymous routers forward Traceroute packets but do not send responses as the TTL expires, while blocking routers discard and ignore any Traceroute packets they receive. We assume that routers are consistent in their behavior with respect to Traceroute probes during the monitor information collection.

We assume that traces are pre-processed at the NOC through standard alias resolution techniques [7], [8], [9] before the execution of topology inference algorithms.

Our objective is to infer a topology from the partial information collected by monitors, which is as close as possible to the GT topology.

IV. ITOP APPROACH

In this section we present iTop, our topology inference approach for partial path information. iTop makes use of Traceroute to probe the network and infers the topology from the gathered traces. We do not assume a specific version of Traceroute, since iTop can work with any of the versions described in Section II.

iTop operates in three phases. In the first phase, it analyzes the traces and constructs the *virtual topology* $G_{VT} = (V_{VT}, E_{VT})$, which is a vastly overestimated topology compared to the ground truth. During the construction of the virtual topology, iTop classifies nodes in the virtual topology on the basis of their observed behavior with respect to Traceroute probes. In the second phase, iTop determines the *merge options* for each link in the virtual topology. These options indicate pairs of links in the virtual topology that can be merged while preserving consistency with respect to characteristics of the ground truth topology observed in the traces and the node classifications assigned in the previous phase. In the third phase, iTop infers the merged topology $G_{MT} = (V_{MT}, E_{MT})$ by iteratively merging pairs of links based on their merge options and removing any merge options made invalid as a result.

A. Virtual Topology Construction

The NOC collects the information gathered by all monitors and constructs the virtual topology as follows. Consider two monitors m_1 and m_2 connected by the path $m_1, v_1, v_2, \ldots, v_{n-1}, m_2$. In order to maximize the gathered information, both monitors take part in the information collection process by probing the path. The monitors first estimate their mutual hop distance $d(m_1, m_2)$ and then execute Traceroute. The gathered traces and hop distances are sent to the NOC.

Node classification. The NOC analyzes the traces to infer the virtual topology and partition the nodes into classes. These classes are introduced in order to guide the merging process of iTop and reflect the router behavior as observed by the Traceroute probes. We define five classes of routers. Classes R, A and B refer to responding, anonymous and blocking routers, respectively. The available information may not be enough to enable the classification of some nodes into one of these categories. For this reason, we include two additional, more

[1]The percentage of anonymous routers is estimated as the ratio between the number of anonymous and responding routers discovered in a path, since the presence of blocking routers may prevent the discovery of some routers.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

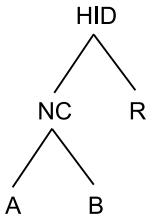


Figure 1. Node class hierarchy.

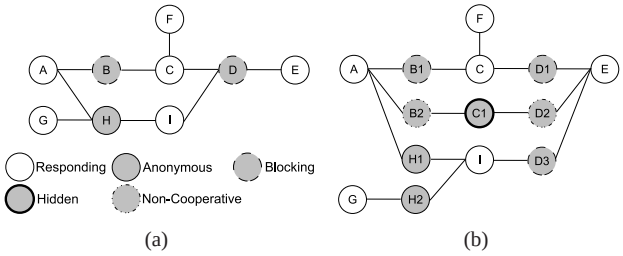


Figure 2. Example: GT (a) and virtual topologies (b).

generic, classes: *non-cooperating*, NC, and *hidden*, HID. NC nodes can be either anonymous or blocking, while HID routers can be responding, anonymous or blocking. As a result, node classes form a hierarchy, as shown in Figure 1. Leaf classes are definite while internal classes are more generic. When processing the collected traces, the NOC marks each router as belonging to one of these classes as described in the following.

Anonymous routers. Consider the case in which a monitor m_1 uses Traceroute to probe the path to m_2 and successfully receives a response from m_2. Since a reply was received, the NOC can conclude that the path does not contain any blocking router. As described in Section 2, the trace from m_1 will be of the form $(m_1, x_1, ..., x_{n-1}, m_2)$ where $d(m_1, m_2) = n$ and each x_i either identifies a router v_i that is responding or is a * to denote no response. As there are no blocking routers in the path, all routers corresponding to a * in the trace must be anonymous. The NOC adds a node in the virtual topology for each responding router observed in the trace and connects them accordingly. It marks these nodes as responding and combines multiple instances of the same responding routers reported by other monitors to avoid duplication of observed components. The anonymous routers and the links connecting them are also added to the virtual topology. A node is added for each * in the trace and it is marked as anonymous.

Blocking routers. If the path does contain at least one blocking router then the trace acquired by m_1 will not contain a response from m_2. In this case the NOC can combine the traces obtained by m_1 and m_2 to create the trace $(m_1, x_1, \ldots, x_i, *, \ldots, *, x_{n-j}, \ldots, , x_{n-1}, m_2)$ where x_i and x_{n-j} are the last responses received by m_1 and m_2, respectively. The NOC treats the trace fragments $(x_1, \ldots, x_i)$ and $(x_{n-j}, \ldots, x_{n-1}, m_2)$ as described above, adding responding and anonymous routers in the virtual topology. The NOC uses the hop distance $d(m_1, m_2)$ to infer that there are $n - j - i - 1$ unobserved routers in the trace fragment $(x_{i+1}, \ldots, x_{n-j-1})$. The NOC adds these routers to the virtual topology and connects them to fill the gap in the path between the routers already added. How these nodes are marked depends upon the number of unobserved routers. If there is only one router between x_i and x_{n-j}, that is $i+1 = n-j-1$, then that router must be blocking and is marked as such. Otherwise, the NOC can only conclude that there is at least one blocking router in the fragment, but cannot uniquely identify it. As a result, it marks x_{i+1} and x_{n-j-1} as non-cooperative while all routers between them, if any, are marked as hidden since there is no way to determine whether they are responding or not. This process is repeated for all communicating monitor pairs.

The fully constructed VT topology overestimates the network because it contains multiple anonymous, blocking, and hidden nodes which are the same router in the GT topology. Since these routers are all represented in the traces by a * there is no simple way to determine which ones are the same. Therefore they are assumed to be separate nodes until merged in the third phase of iTop.

The complexity of the virtual topology construction phase depends on the size of the set $\mathcal{M}$ of probed monitor pairs. For each pair $(m_i, m_j) \in \mathcal{M}$, iTop analyzes the trace between them and updates the current VT graph accordingly. The length of the trace is linear with respect to the distance $d(m_i, m_j)$. Since graphs updates can be performed in constant time, the overall complexity of this phase is $O(|\mathcal{M}|)$.

An example

Figures 2 (a-b) show an example of a GT topology and the corresponding virtual topology. The legend in Figure 2 (a) indicates the type of each node as it exists in the GT topology and is marked by the NOC in the VT topology. In this example we assume that nodes A, E, F, G, and I act as monitors and the following pairs of monitors communicate: A-E, A-F, A-G, A-I, E-F, E-G, and G-I. All paths are shortest distance, and the path from A to E goes through B, C, and D.

In the path A-F, the blocking router B drops all Traceroute packets sent from A. As a result, A receives no reply from C or F. In the other direction, F can detect the segment F-C, but B prevents the collection of further information. By using the measured hop distance, F and A can determine that the path is composed of exactly three hops. Therefore, the NOC determines that there is only one blocking router between them and adds the path A-B1-C-F to the virtual topology, with B1 marked as blocking.

In the path A-E, the blocking routers B and D prevent the identification of the responding router C. The monitors measure a path length of four, and the NOC adds the path A-B2-C1-D2-E to the virtual topology. Both B2 and D2 are marked as non-cooperative, while C1 is marked as hidden. This is repeated for all paths in the network. The resulting virtual topology is shown in Figure 2 (b).

Note that the virtual topology contains a larger number of nodes and links than the GT topology. Some of the nodes in Figure 2 (b) correspond to the same nodes in the GT topology. In the case of nodes B, D, and H, this is because they are non-cooperative, thus the traces for paths they occur in only contain a * for their response. C is located between blocking routers

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

B and D so it also appears in the VT as a hidden node.

B. Merge Options

In order to infer G_{MT} from G_{VT}, iTop identifies the valid merge options for each link e_i in E_{MT}, i.e., the set of links with which e_i can be merged. We introduce three conditions which have to be satisfied for a merge option to be valid. These conditions check the consistency of a merge option with the information gathered from the traces and with the node classification provided in the previous phase.

The set $\mathsf{M_i}$ denotes the set of links which are valid merge options for link e_i, and initially $\mathsf{M_i} = \emptyset$ for each $e_i \in E_{MT}$. The sets of merge options are then used during the merging phase to determine which merges occur and in what order. We define the following conditions for merge options, which are checked for each pair of links.

Trace Preservation: Since paths do not contain loops, a link will never appear twice in the same path. A merge option between two links satisfies the trace preservation if these links do not appear together in any path. More formally, let $p_1, \ldots, p_{|\mathcal{M}|}$ be the probed paths, where each path is a set of links. Trace preservation is verified for two links e_i and e_j if $\nexists p_k$ s.t. $e_i \in p_k$ and $e_j \in p_k$.

Distance Preservation: The distance between two monitors in G_{VT} is consistent with the traces. The merging process keeps this consistency by preventing any merges which would decrease this distance. Formally, let $G_{MT}^{e_i,e_j}$ be the topology resulting by merging two links e_i and e_j. Such links verify the distance preservation if for each pair of communicating monitors m_1, m_2 the distance $d_{G_{MT}^{e_i,e_j}}(m_1, m_2) = d_{G_{VT}}(m_1, m_2)$.

Link Endpoint Compatibility: A merge option between two links is valid if there is a way to combine their endpoints without violating the hierarchy in Figure 1. Intuitively, the merging process can only increase the specialization of a node.

Table 2 shows the types of endpoints that compatible links can have. Entries in this table can be reversed, for example R-A is the same as A-R. If two links have incompatible endpoints then the entry is marked with an "-". Otherwise the entry contains the endpoint classes of the link that would result if the links were merged. The entries where two routers with class R are combined are valid only if the responding router is the same in both links. How the endpoint types of the merged links are decided is further described in the merging phase section.

Referring to the example VT topology in Figure 2 (b), it is possible to see how combinable endpoints supplement distance and trace preservation. Those two rules alone provide no restrictions on the classes of link endpoints that can be combined, so merging all vertically aligned links, such as (A,B1), (A,B2), (A,H1), and (G,H2), would be possible without checking endpoint compatibility. Table 2 shows that R-B and R-A links should not have a merge option, so this will prevent (A,B1) from having a merge option with (A,H1). Further, the fact that the identities of responding nodes are known will prevent (A,H1) and (G,H2) from satisfying endpoint compatibility, but will allow (H1,I) and (H2,I) to satisfy all three merge option requirements.

The merging option phase analyzes all link pairs in VT and for each pair it verifies if merging such links would violate the conditions described above. Trace preservation can be checked in constant time by keeping a table in which we store for each link the paths in which it appears. Distance preservation for a link pair can be verified by calculating the new distances that would result by merging the link in the pair, and by comparing them with the previous distances. The Floyd-Warshall algorithm can be used, with a worst-case complexity of $O(|V_{VT}|^3)$. Finally, link endpoint compatibility can be checked in constant time by using the entries of Table 2. As a result, the overall complexity of the merging option phase is $O(|E_{VT}|^2 \times |V_{VT}|^3)$.

C. Merging Links

The next phase merges the links in the virtual topology to derive the iTop topology G_{MT}. Initially, $G_{MT} = G_{VT}$. The merging phase reduces G_{MT} by iteratively merging pairs of links based upon the existing merge options. Each merge combines two links in E_{MT}, combining their endpoints and reducing the number of components in the network accordingly. When no merge options remain, the merging phase is complete and the iTop topology is finalized.

Algorithm 1: iTop Merging Phase

Input: Initial iTop Topology $G_{MT} = (V_{VT}, E_{VT})$,
Merge Options $\mathsf{M_i}$ for each link $e_i \in E_{MT}$
Output: Merged iTop Topology $G_{MT} = (V_{MT}, E_{MT})$

1 **while** $\exists e_i \in E_{MT}$ *s.t.* $\mathsf{M}_i \neq \emptyset$ **do**
2 $e_i = argmin_{e_i \in E_{MT}} |\mathsf{M}_i|$;
3 $e_j = argmin_{e_j \in \mathsf{M}_i} |\mathsf{M}_j|$;
  `// Endpoint compatibility check`
4 **if** $C(e_i, e_j) = true$ **then**
    `// Link merging`
5 Merge(e_i, e_j);
6 **else**
7 $\mathsf{M}_i = \mathsf{M}_i \backslash \{e_j\}$;
8 $\mathsf{M}_j = \mathsf{M}_j \backslash \{e_i\}$;
9 **return** $G_{MT} = (V_{MT}, E_{MT})$

Algorithm 1 shows the pseudo-code of the merging phase. In each step, iTop chooses two links and attempts to merge them. Several alternatives are possible to determine the order in which links are merged, which influence the resulting final topology. Since links with few merging options have fewer merging possibilities, they are more likely to be the same link in the ground truth topology. On the basis of this observation, we first select the link e_i with the fewest merging options, and then link e_j which has the fewest merging options out of the links with which e_i can be merged (Alg. 1, lines 2-3). We experimented with several alternative heuristics and the one described above provides the best results.

Endpoint compatibility check
Before merging two links, their endpoint compatibility is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

	R-R	R-A	R-B	R-NC	A-A	A-HID	NC-NC	NC-HID	HID-HID	A-NC	B-NC	A-B
R-R	-	-	-	-	-	-	-	-	R-R	-	-	-
R-A	-	**R-A**	-	**R-A**	-	R-A	-	R-A	R-A	-	-	-
R-B	-	-	**R-B**	**R-B**	-	-	-	R-B	R-B	-	-	-
R-NC	-	**R-A**	**R-B**	**R-NC**	-	-	-	R-NC	R-NC	-	-	-
A-A	-	-	-	-	A-A	A-A	A-A	A-A	A-A	A-A	-	-
A-HID	-	R-A	-	-	A-A	A-HID	A-NC	A-HID	A-HID	A-NC	A-B	A-B
NC-NC	-	-	-	-	A-A	A-NC	NC-NC	NC-NC	NC-NC	A-NC	B-NC	A-B
NC-HID	-	-	-	-	A-A	A-HID	NC-NC	NC-HID	NC-HID	A-NC	B-NC	A-B
HID-HID	R-R	R-A	R-B	R-NC	A-A	A-HID	NC-NC	NC-HID	HID-HID	A-NC	B-NC	A-B
A-NC	-	-	-	-	A-A	A-NC	A-NC	A-NC	A-NC	A-NC	A-B	A-B
NR-NC	-	-	-	-	-	A-B	B-NC	B-NC	B-NC	A-B	B-NC	A-B
A-B	-	-	-	-	-	A-B	A-B	A-B	A-B	A-B	A-B	A-B

Table II
COMPATIBLE ENDPOINT CLASSES AND RESULTING CLASSES AFTER MERGING. BOLDED ENTRIES ARE VALID ONLY IF BOTH OF THE MERGED RESPONDING ROUTERS ARE THE SAME.

rechecked by the function $C()$, (Alg. 1, line 4), according to Table 2. This check is necessary as previous merges may have changed the links' endpoint classes since the initial check during the merge option phase.

Additionally, each path containing one of the two links is checked to make sure it will retain a coherent ordering of link endpoints should the merge occur. This coherence check is necessary because iTop is designed to keep the endpoint classes as generic as possible during the merging phase. Furthermore, as explained in the Link merging section below, iTop does not commit to one specific direction if there are two alternatives to combine the endpoints of the links being merged until this is implied by the merge operation. Since each link has two endpoints, there are two different ways in which the endpoints can be combined and both may result in the same amount of generality without violating distance preservation. A path consisting of links $e_1, \ldots, e_n$ is coherent if there exists a mapping of node classes $c_1, \ldots, c_{n+1}$ such that the endpoint classes of each e_i are combinable with classes c_i and c_{i+1}, according to the node class hierarchy. The mapping should contain specific responding routers instances, instead of just the responding class, because each responding router is identifiable.

For the VT topology in Figure 2 (b), consider checking the coherence of path A-B2-C-D2-E if the link (B1,C), of type R-B, is merged with (B2,C1), which has type NC-HID. This merge will result in a link with endpoint classes R-B. The node class ordering Node A, B, Node C, B, Node E shows that this path is coherent.

If the link types are not compatible or the merge will cause an incoherent path then the function $C(e_i, e_j)$ in the pseudo-code of Algorithm 1 returns *false*. In this case, links are removed as merging options for each other (Alg. 1, lines 7-8) and the algorithm repeats the step to choose a new pair of links. Otherwise, if the link types are compatible and all paths are coherent, the link e_i is merged with e_j as described in the following.

Link merging

If two links e_i and e_j pass the compatibility check, they are merged by the function $Merge(e_i, e_j)$ (Alg. 1, line 5). For ease of exposition we describe the merging of e_j into e_i, as the same result would be obtained by the opposite merging. All paths containing e_j are modified to contain e_i in its place and the set M_i is changed so that $\mathsf{M}_i = \mathsf{M}_i \cap \mathsf{M}_j$. Any links which could have been merged with both e_i and e_j retain their merge option with e_i and the merge option for e_j is removed. Any links which had a merge option with either e_i or e_j but not both have that option removed. This ensures that e_i can only be further merged with links that before were valid merges for both e_i and e_j. The link e_j is then removed from E_{MT}.

$Merge(e_i, e_j)$ also changes the endpoint classes of e_i according to Table 2, where the endpoint classes of e_i and e_j are matched to the first row and to the first column of the table, respectively. The classes are changed according to the hierarchy in Figure 1. The link with the most specialized endpoint (i.e., lower level in the hierarchy) determines the class of the resulting endpoint. We keep the endpoints as generic as possible during the merging operations in order to facilitate further merging.

Since links have two endpoints, there might be two alternatives to merge them. In most scenarios, one of these alternative is generally ruled out by the classes of the endpoints, the coherence of paths, or the links sharing that endpoint. If none of these cases apply, iTop does not immediately commit to either way of combining them. This is done to avoid specifying the classes of link endpoints beyond what is implied by each merge. As a result, iTop has more freedom in what merges can be performed in later iterations. Only when no further merges can be carried out and there are still links for which both combinations are valid is one of the two ways chosen randomly.

Figures 3 (a-c) show three of the steps in the merging phase for the example shown in Figure 2. The virtual topology and the number of merging options for each link are depicted in Figure 3 (a). For the first merging steps, one of the links with a single merging option is picked and combined with its only available option. In this example, (A,B1) is merged with (A,B2), (B1,C) is merged with (B2-C1), and (C-D1) is merged with (C1-D2). There is only one way in which the classes of the endpoints of these links can be combined, thus iTop commits to combining the endpoints. The resulting partially merged network and the classes of the merged nodes are shown in Figure 3 (b). Note that the endpoint classification prevents several erroneous merging from occurring. As an example, (A,B1) is not merged with (A,H1), since B1 and H1 have two incompatible classes, blocking and anonymous, and thus they must be two different nodes in the GT.

There are two possible merges remaining. (H1,I) is merged

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

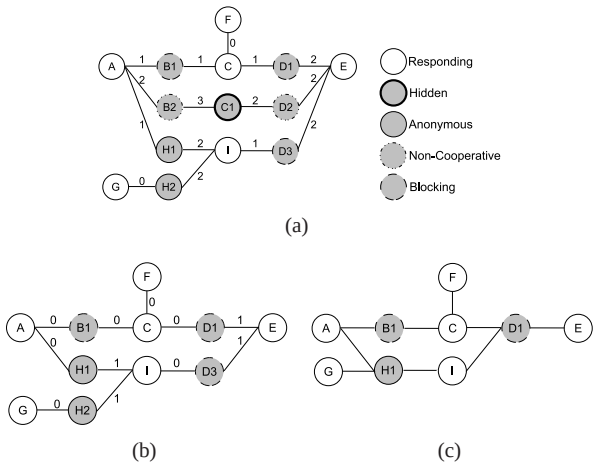


Figure 3. Example: G_{MT} topologies before (a), during (b), and after (c) the merging process.

with (H2,I) and (D1-E) is merged with (D3-E). This results in the merged topology shown in Figure 3 (c), which correctly represents the ground truth topology depicted Figure 2 (a).

Note that depending on the order in which links with the same number of options are merged, iTop may infer a topology slightly different from the GT topology. Our results in Section VI show that the topologies inferred by iTop closely match the ground truth, even with networks that are significantly larger than in this example.

The complexity of the merging link phase can be derived from Algorithm 1. The while loop (lines 1-8) is executed at most $O(|E_{VT}|^2)$ times, since there are at most $O(|E_{VT}|^2)$ merging options overall and at each iteration we remove at least one option. At each iteration, we can select the links to merge in $O(|E_{VT}|)$ time (lines 2-3). Recalling that $\mathcal{M}$ is the set of probed monitor pairs, the endpoint compatibility check (line 4) has a complexity $O(|\mathcal{M}|)$, since it requires checking all paths in which the selected links appear and these are at most $|\mathcal{M}|$. Finally, merging two links can be performed in constant time using appropriate data structures for paths and merging options. As a result, the overall complexity of the merging link phase is is $O(|E_{VT}|^2 \times |\mathcal{M}|)$.

V. Related approaches

In this Section we describe the approaches we consider for performance comparison with iTop. We selected these approaches as they provide the best performance in terms of accuracy of the inferred topologies. As discussed in Section VIII, other works either target a tradeoff between accuracy and complexity of the inference process [12], or require additional data which may often not be available [13].

Merging Nodes (MN) [10] is a Traceroute-based approach to infer the network topology in the presence of anonymous routers. MN collects the path information between monitors and uses it to construct an initial *induced topology*. Similar to the virtual topology, the induced topology contains several duplicated components which are reduced by performing merging operations.

MN iteratively merges nodes in the induced topology. In each iteration it builds an *equivalence class* which is constructed by incrementally adding one node at a time, chosen from the nodes that have not yet been merged. A node is added into the class if it can be merged with every node already in the class. The iteration terminates when the class cannot be extended further, and all of the nodes it contains are merged into a single node in the resulting MN topology.

According to MN, two nodes can be merged only if they never appear in the same path and if the resulting topology will not shorten the minimum distance between any two nodes as observed in the induced topology.

Since MN is designed only for anonymous routers, the construction of the induced topology does not take into account the lack of information caused by blocking routers and therefore may result in a disconnected network.

Isomap [11] is another Traceroute-based approach, which considers the presence of both anonymous and blocking routers. It constructs an *initial topology* that contains a virtual router for each anonymous router observed in the traces. Unlike iTop, no virtual router is added to the network when a blocking router is detected on a path. Instead a link is added between the last responding routers identified in the traces on either side of the unobserved section. Intuitively, Isomap's initial topology may underestimate the ground truth topology, as the portion of the network hidden by blocking routers is not considered at all.

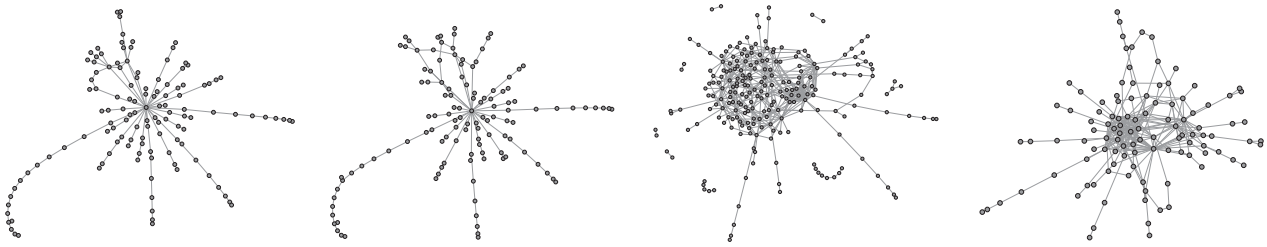
Isomap has two merging phases: *initial pruning* and *router merging*. The initial pruning merges virtual nodes that share the same neighbors. In the router merge phase, each router is represented as a point in a multi-dimensional space using hop distance or round trip time as a distance metric. A mapping to a lower dimensional space is performed using the algorithm proposed in [18]. The merging process is governed by two thresholds, Δ_1 and Δ_2. Two nodes are merged together if in the lower dimensional mapping the distance between two virtual routers is less than Δ_1 or if their distance is less than Δ_2 and they share at least one common neighbor.

MN, Isomap and iTop share a common design framework, according to which an initial topology is built from the traces, and merges are then performed to produce the inferred topology. Nevertheless, compared to previous approaches, iTop provides a much deeper trace analysis by means of virtual topology construction and node classification. These enable the derivation of merging options and operations that result in more accurate topologies, even if several routers in the network are non-cooperative.

To better highlight the advantages of iTop, in the following, we show that iTop provides better performance than MN and Isomap, in the case of hub topologies (Figure 5 (a)), where the hub is a *blocking* router. We consider this topology since hubs are common in the Internet [19], and it is likely that they are blocking due to the experienced high traffic.

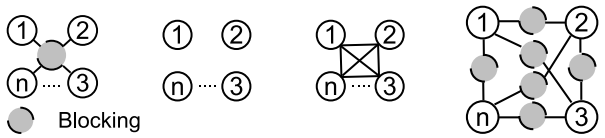
Theorem V.1. *Consider a hub GT topology with n peripheral nodes, where each peripheral node acts as a monitor and the hub is a blocking router. With respect to GT, the topology inferred by MN has* 1 *less node and n less edges, the topology*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management



(a) (b) (c) (d)

Figure 4. An example of inferred topologies with 10% anonymous and 10% blocking routers: GT (a), iTop (b), MN (c), and Isomap (d).



(a) (b) (c) (d)

Figure 5. Theorem V.1: Hub GT topology (a), topology inferred by MN (b) and Isomap (c). Virtual topology of iTop (d).

inferred by Isomap has 1 *less node and* $\frac{n(n-1)}{2} - n$ *extra edges, while the topology inferred by iTop perfectly matches GT.*

Proof. MN does not consider blocking routers. As a result, discarded probes may cause the induced topology not to include some nodes and edges. For the hub topology considered here, the blocking router discards all probes as soon as they are sent, hence the induced topology only includes the monitors (Figure 5 (b)). This topology is the final topology of MN, since no merges are possible, and it has 1 less node and n less edges with respect to GT.

For each probe discarded by a blocking router, Isomap adds an edge between the routers immediately before and after such a router. Hence, in this case it adds an edge between each pair of monitors, as shown in Figure 5 (c). Since no merge is possible because Isomap only merges anonymous routers, the inferred topology has $\frac{n(n-1)}{2} - n$ extra edges and one less node (the blocking router) with respect to GT.

iTop construct the virtual topology by adding an extra router and corresponding edges for each pair of monitors, as show in Figure 5 (d). The only merge options that do not violate distance preservation, trace preservation and link endpoint compatibility are those between the edges incident to each monitor. As a result, iTop performs such merges and eventually infers a topology that perfectly matches GT. □

We now compare the complexity of MN, Isomap and iTop. According to [11], MN has a complexity $O(|V_{GT}|(|V_{GT}| + |V_R| + |V_A|)^2|V_A|)$ and Isomap $O((|V_{GT}| + |V_R| + |V_A|)^3)$, where V_R and V_A are the set of responding and anonymous routers, respectively. The complexity of iTop is dominated by the calculation of the merging options, hence the complexity is $O(|E_{VT}|^2 \times |V_{VT}|^3)$. As a result, iTop has a complexity slightly higher than MN and Isomap, but the gap reduces in sparse networks, as is the case of real Internet topolgies [19]. As shown in Section VI, the slight increase in complexity translates in significant improvements in terms of quality of the inferred topologies.

VI. Topology evaluation

In this section we compare the performance of iTop, MN, and Isomap through simulations on both realistic and random networks. In both cases we deploy 40 monitors as edge nodes randomly. We consider 10 monitor deployments for each type of network and average the obtained results.

We consider random and realistic *starting networks*. Given a starting network and a monitor placement, the GT topology is obtained from the union of the paths between monitors. In each GT topology, a fraction of the nodes are randomly designated as anonymous and blocking routers. On the basis of this assignment, MN, Isomap, and iTop determine the induced, initial and virtual topologies, respectively, which are used as input for the subsequent merging operations.

In our experiments we use hop distance as the distance metric for Isomap as the potential instability of round trip time may negatively affect merging, as described in [11]. For Isomap we reduce to a 5-dimensional space as in [11] and set the thresholds to $\Delta_1 = 10$ and $\Delta_2 = 4\Delta_1$. Note that an optimal setting of the thresholds highly depends on the specific topology and monitor placement. We simulated several scenarios and selected a setting which performs well in the majority of cases.

A. Realistic Networks

Here we compare the merged topologies produced by iTop, MN, and Isomap for realistic topologies. We use the Autonomous System (AS) topologies from both the Rocketfuel [20] and the CAIDA [21] projects, which represent IP-level connections between backbone/gateway routers of several ASes from major Internet Service Providers (ISPs) around the globe. Due to space limitations we only show results for the CAIDA networks, as similar results are observed with the Rocketfuel topologies. The starting CAIDA network from which the GT topologies are derived consists of 250 nodes and 290 edges.

We consider two scenarios regarding the types of routers which are present in the network. In the first scenario networks include both anonymous routers and blocking routers. This is a realistic setting as shown by our experiments in Section 2, and it is this case for which iTop and ISOMAP are designed. Since MN is only intended for anonymous routers, the second

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

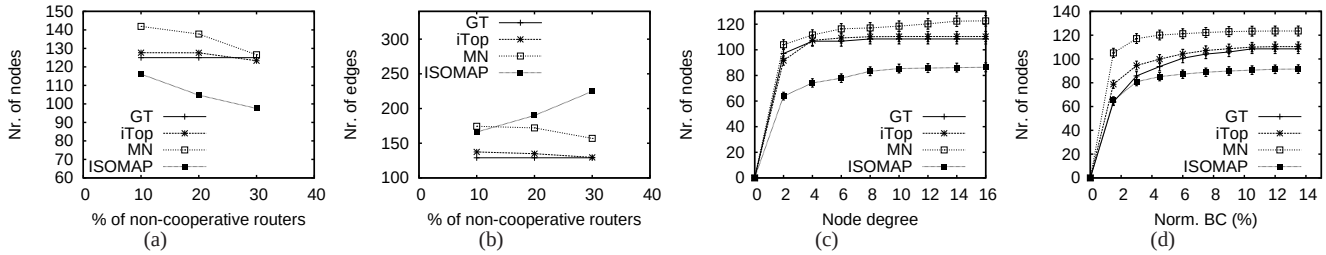


Figure 6. Realistic networks with anonymous and blocking routers: number of nodes (a), number of links (b), cumulative distribution of node degree (c), and betweeness centrality, (d).

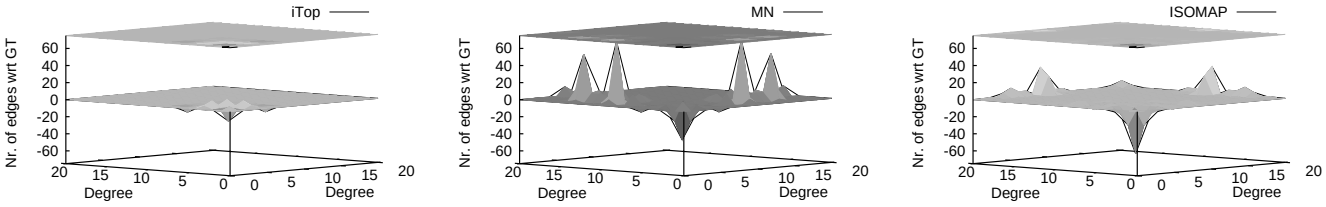


Figure 7. Joint degree distribution with respect to GT: iTop (a), MN (b), and Isomap (c).

scenario has only responding and anonymous routers and does not include any blocking ones.

Scenario 1: Anonymous and Blocking

In Figures 4 (a-d) we show an instance of GT topologies and the resulting topologies inferred by iTop, MN and Isomap[2]. This portrays a case in which 10% of the routers in the network are anonymous and another 10% of the routers are blocking.

The most notable feature of the GT topology (Figure 4 (a)) is the presence of a central hub router which connects several branches of the network. The topology inferred by iTop (Figure 4 (b)) clearly reflects the existence of this hub and resembles the original topology. MN (Figure 4 (c)) significantly overestimates the network due to its notion of distance preservation, as preserving the distance between *any* two known nodes in the network prevents a significant number of merges. Furthermore, the inferred topology is disconnected because MN does not take into account the presence of blocking routers when constructing the induced topology from the traces. Isomap (Figure 4 (d)) is able to perform more merges than MN, but the structure of the network is still significantly different from the GT topology. In addition, it underestimates several parts of the network, as shown by the branches that are shorter than in the GT topology. This is due to the fact that the construction of the induced topology uses only a single link to represent portions of a path hidden between two blocking routers.

Figures 6 (a) and (b) show the average number of nodes and links, respectively, in the inferred topologies as the fraction of non-cooperative nodes in the networks increases. We consider the number of non-cooperative nodes to be half anonymous and half blocking. These figures highlight that iTop outperforms other approaches, as the number of nodes and links are closer to the ground truth with iTop in all considered cases. MN and Isomap both overestimate the number of links because they perform merging operations on nodes whereas iTop merges links. This makes it less likely that MN and ISOMAP will combine two links, as both endpoints of the links must be merged together in order for the links to be merged. MN tends to achieve results closer to the GT topology as we increase the percentage of non-cooperative routers. This does not imply a better inferred topology. On the contrary, the increased number of blocking routers results in a larger portion of the network being hidden, and therefore not present in the induced topology. As a consequence, as we increase the fraction of non-cooperative routers, the topology inferred by MN contains more disconnected components and each of those components is overestimated with respect to the GT topology. Isomap underestimates the number of nodes in the network as the initial topology does not contain the portions of the network hidden by blocking routers.

In Figure 6 (c) we show the cumulative distribution of node degree[3]. iTop infers a better topology which closely approximates the degree distribution of the GT topology. MN has more nodes with a higher degree, which are not present in the ground truth. Isomap underestimates the number of nodes, and as a result the cumulative distribution significantly deviates from the distribution for the GT topology. These results highlight the substantial structural differences between the topologies inferred by MN and Isomap with respect to the ground truth.

[2]These topologies are represented using the Force Atlas layout algorithm of the Gephi tool [22].

[3]Note that we show the absolute number of nodes on the y-axis in order to better highlight the differences in the inferred topologies. These differences may have been hidden by normalization.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

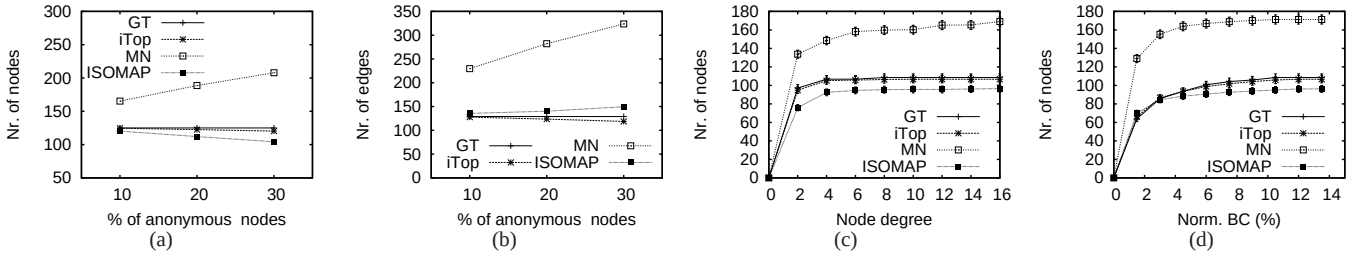


Figure 8. Realistic networks with only anonymous routers: number of nodes (a), number of links (b), cumulative distribution of node degree (c), and betweeness centrality (d).

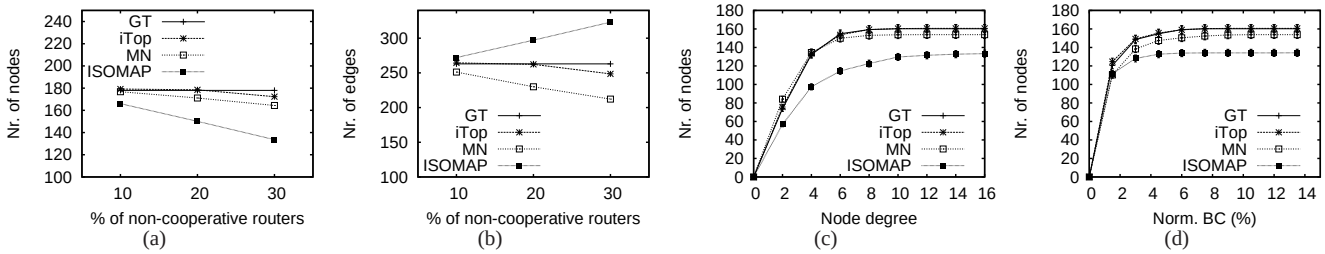


Figure 9. Random networks with anonymous and blocking routers: number of nodes (a), number of links (b), cumulative distribution of node degree (c), and betweeness centrality (d).

Figure 6 (d) shows the cumulative distribution of the normalized betweeness centrality. The normalized betweeness centrality of a node is calculated as the percentage of shortest paths in the network that go through that node. This metric indicates the degree of structural similarity of the inferred topologies with respect to the GT topology. As the figure shows, MN overestimates and Isomap underestimates the real distribution. This is a consequence of the respective overestimation and underestimation of the GT topology that they perform. In comparison, iTop closely matches the ground truth, highlighting structural similarity of its inferred topology to the GT topology.

These results show that the topologies inferred by iTop are within 5% of the GT topologies with regard to all of the considered metrics.

In order to further compare the structure of the inferred topologies to the GT topology, we consider the *Joint Degree Distribution* (JDD), which has been recognized as a meaningful metric for topology comparison [16]. Given a network $G = (V, E)$, the JDD counts the number of links that connect a node with degree x to a node of degree y for $x, y \in [0, |V|-1]$. In Figures 7 (a-c) we show 3D representations of the relative JDD for the inferred topologies with respect to the JDD of the GT topology. Flatness in these diagrams indicates that the JDD of the inferred topology is close to that of the GT topology. As depicted in Figure 7 (a), the topology inferred by iTop results in a flat relative JDD, once again indicating a good inference of the GT topology. Since MN is not able to correctly infer hub nodes, it has more nodes with a high degree connected to nodes with low degree, as Figure 7 (b) shows. These nodes should have been merged together to correctly infer the hub. The JDD of the Isomap topology highlights the fact that this algorithm underestimates the nodes with low degree with respect to GT, as shown in Figure 7 (c). The presence of blocking routers causes several nodes located on the branches of the realistic topology to not be represented in the induced topology (see Figure 4 (d)). Furthermore, Isomap is not able to correctly infer the hub, so the inferred topology has more nodes with high degree than the GT does.

Scenario 2: Anonymous Routers Only

In this scenario we consider the presence of only anonymous routers in order to study the performance of iTop in the setting for which MN is designed. Results are shown in Figures 8 (a-d).

Although MN is designed to operate in this setting, it significantly overestimates the number of nodes and links in the network as shown in Figures 8 (a-b). MN ensures the shortest distances between any two nodes are preserved, thereby preventing many correct merges from occurring. In comparison, Isomap and iTop better infer the real topology, although Isomap performs some extra merging operations which result in a slightly lower number of nodes than are present in the GT topology.

Figures 8 (c-d) show the the degree distribution and normalized betweeness centrality distribution, respectively. These metrics highlight that MN does not perform enough merging operations, resulting in an inferred topology that significantly differs from the GT topology. The distributions of iTop and Isomap inferred topologies better match the distribution of the GT topology, but Isomap slightly underestimates the ground truth.

In this scenario iTop still achieves results which are within 5% of the GT for all of the considered metrics. Note that Isomap performs better in this scenario because the absence of blocking routers enables the construction of a better initial topology that contains all nodes and links in the network. However, real scenarios are characterized by the presence of both anonymous and blocking routers, as shown by our

experiments in Section II and the experiments in [15]. In these real contexts Isomap performs poorly as previously shown.

B. Random networks

The simulated random networks are generated by ensuring that a single connected component is present. First a tree with a given number of nodes is randomly created and then additional links are added randomly to increase the network connectivity. The starting random network from which the GT topologies are derived contains 200 nodes and 350 edges before the addition of monitors. We consider random networks in order to study the performance of the algorithms with GT topologies having significantly different structures than are present in the realistic networks. Due to space limitations we only consider the case with both blocking and anonymous routers. Results are shown in Figures 9 (a-d).

The random topologies are characterized by degree distributions that are more even than those of the realistic networks, as shown in Figure 9 (c). As a result, there is more variation in the links that the paths contain, and blocking routers cause a larger portion of the network to be unobservable. Since MN and Isomap do not specifically handle the effects of these blocking routers, the induced and initial topologies from which they start merging underestimate the random GT topologies more than they do realistic GT topologies. For this reason, MN underestimates the number of nodes and links (Figure 9 (a-b)). Isomap also underestimates the number of nodes, but it significantly overestimates the number of links because it performs merging operations on nodes. iTop outperforms the other two approaches as its inferred topology closely matches the number of nodes and links in the GT topology even as the number of non-cooperative routers increases.

Figures 9 (c-d) show the distribution of node degree and normalized betweeness centrality. In this case, iTop also outperforms MN and Isomap, showing a closer match to the GT topology for these distributions as well.

These results show that even with random networks the topologies inferred by iTop are within 5% of the GT topologies.

VII. Application: fault localization

In this section we evaluate iTop, MN and Isomap by considering fault localization as an application of inferred topologies. Link failures are common in modern networks due to maintenance procedures, hardware malfunctions, energy outages, or disasters [23] and may cause degradation in performance. Fault localization techniques [2], [24] generally assume complete knowledge of the network topology. When this knowledge is not available, failures are diagnosed on the inferred topologies. Intuitively, a better inferred topology enables a fault localization algorithm to achieve performance closer to what can be obtained when the full GT topology is known.

In this paper, we consider the Max-Coverage (MC) algorithm [2], one of the most referenced approaches. In the following section we first describe MC and then introduce the performance metrics used to evaluate the output of MC in the context of inferred topologies. Finally, we present the experimental results.

A. Max-Coverage

Algorithm MAX-COVERAGE

Input: Network topology $G = (V, E)$, symptoms S
Output: List of failed links FL

$FL = \emptyset$;
$A = E$;
while $S \neq \emptyset$ **do**
  $e^* = \text{argmax}_{e \in A} |\texttt{explained}(G, S, e)|$;
  $FL \leftarrow e^*$;
  $A = A \setminus \{e^*\}$;
  $S = S \setminus explained(G, S, e^*)$;
return FL

MC is a greedy algorithm for diagnosing network link failures. Its input is the network topology, represented as a graph $G = (V, E)$, and a set of symptoms S. Symptoms represent disconnections between network monitor pairs. Each symptom represents a pair of monitors that cannot communicate because of link failures in the network.

The pseudo-code for MC is shown in Algorithm MAX-COVERAGE. The function `explained` (G, S, e) returns the set of symptoms in S that are explained by the failure of the link e. A symptom is explained by the failure of e if such a failure causes the disconnection between the monitors that generated the symptom. The list FL contains the links which MC has identified as the cause of the observed symptoms and is initialized as an empty set. In each iteration of the while loop, MC selects a link which explains the most unexplained symptoms and adds it to FL. The loop terminates when all symptoms are explained by the links in FL. The algorithm then outputs the list FL.

In some cases there may be a tie between two or more links that all explain the most unexplained symptoms. When this occurs, MC randomly selects one of the tied links to add to FL.

B. Performance metrics

When MC is applied to an inferred topology, the resulting list FL may contain links that do not correspond to only a single link in the GT topology. One or more links in FL may represent one or multiple real links at the same time, due to possible erroneous inferencing. Therefore, to evaluate the performance of MC in terms of real links, we propose different definitions of basic performance metrics, such as *accuracy* and *false positives*. Moreover, we propose an additional new metric called *redundancy*.

When using MC, if two or more links occur in the same set of paths in a network topology then they will always explain the same set of symptoms. We refer to these links as *indistinguishable* links. MC treats indistinguishable links by breaking the tie randomly and adding one of them to FL. This may result in an increase in the number of false positives and a decrease in accuracy.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2451032, IEEE Transactions on Network and Service Management

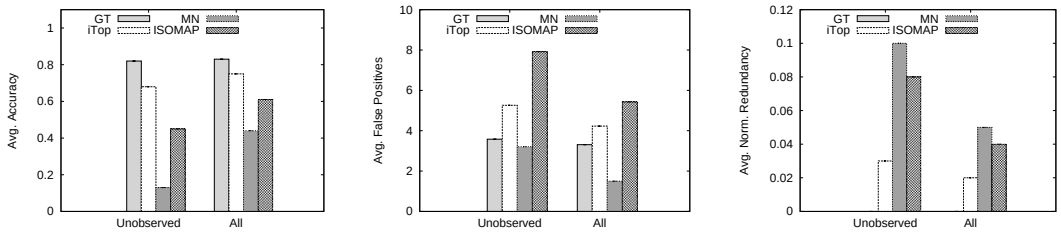


Figure 10. Average accuracy (a), false positives (b) and redundancy (c) of MC with GT and inferred topologies.

The effect of indistinguishable links is much more noticeable in inferred topologies than in the GT topology, because one or more inferred links may represent one or multiple real links. Based on this observation, given a failed list FL we define an extended list FL_{EXT} which includes all indistinguishable links corresponding to the links present in FL. Hence $FL_{\mathsf{EXT}} = \bigcup_{e \in FL} \left|\{q \in E \text{ s.t. } P(e) = P(q)\}\right|$ where for a link e, $P(e)$ refers to the set of paths between monitors in which q occurs in the inferred topology.

In order to take into account the possible many-to-many relationship between links in FL_{EXT} and the real links in the GT topology, we define the *hypothesis list*, $\mathsf{H} = \{Q, D\}$, as a multi-set. Q is a set of real links that are represented by the links in FL_{EXT}, and $D : Q \rightarrow \mathbb{N}$ gives the multiplicity for each link in Q. More formally, $Q = \bigcup_{e \in FL_{\mathsf{EXT}}} rl(e)$ where $rl(e)$ is the set of real links represented by e in the inferred topology. The function $D()$ for a real link $q \in Q$ is defined as $D(q) = |\{e \text{ s.t. } e \in FL_{\mathsf{EXT}} \wedge q \in rl(e)\}|$ Hence, the size of the hypothesis list H can be defined as $|\mathsf{H}| = \sum_{q \in Q} D(q)$.

Given the set of *Actual Failed Links* $AFL \subseteq E_{\mathsf{GT}}$, we define the following performance metrics:

Accuracy (ACC): $\frac{|Q \cap AFL|}{|AFL|}$. Accuracy represents the fraction of real links that fail and are represented by at least one link of the inferred topology in the output of MC.

Redundancy (RED): $\sum_{q \in Q \cap AFL} \big(D(q) - 1\big)$. Some of the real links in AFL may appear multiple times in the output of MC for an inferred topology. The redundancy metric measures the number of redundant times failed links are reported in MC.

False positives (FP): $\sum_{q \in Q \setminus AFL} D(q)$. False positives count the number of links which have not failed, but are present in the output of MC.

According to the above definitions, it can be seen that $|\mathsf{H}| = ACC \times |AFL| + RED + FP$.

C. Results

To test the performance of the inferred topologies with MC, we simulate link failures on the realistic network shown in Figure 4 (a). We consider a scenario with 10% anonymous and 10% blocking routers. The topologies inferred by iTop, MN and Isomap are shown in Figures 4 (b), 4 (c) and 4 (d), respectively.

We consider two failure scenarios. In the first scenario, we randomly fail an *unobservable* link in the ground truth topology. An unobservable link is one for which at least one endpoint is a non-cooperative router. The unobservable nature of the failed links makes this scenario particularly meaningful for the performance of failure localization algorithms, as additional probing cannot be used to confirm and refine the returned hypothesis list. In the second scenario, we randomly fail a generic link in the network, no matter the classes of its endpoints. For each scenario we averaged the results over 100 trials.

Tests in which multiple independent failures occurred simultaneously were also performed, but are not displayed due to space constraints. The same trends between the different inferred topologies result when more failures occur.

Figure 10 (a) shows the accuracy of MC when applied to GT and the inferred topologies. The good inference provided by iTop is reflected in the accuracy achieved by MC. iTop outperforms the other approaches and achieves accuracy close to that of GT. MN in particular suffers in the unobserved scenario as blocking routers cause only a fraction of the unobserved links to appear in the initial topology of MN. Isomap does not provide a sufficiently precise estimation of the ground truth to enable MC to correctly localize the failure, resulting in a lower accuracy.

The number of false positives is shown in Figure 10 (b). iTop shows only a slight increase in the number of false positives with respect to GT, highlighting that our approach enables MC to provide a hypothesis list that only contains a few links erroneously reported as failed. MN shows a lower number of false positives, but this is highly counterbalanced by the low accuracy that this approach provides, as Figure 10 (a) shows. Isomap incurs a high number of false positives. This is due to the construction of the induced topology, which represents any portion of the network included between two blocking routers with a single link. This aggregate representation enables MC to only identify failures at a coarse granularity, resulting in a large number of false positives.

Figure 10 (c) shows the redundancy, as defined in Section VII-B, normalized by the accuracy. This metric represents the amount of redundant information reported by MC when the algorithm correctly reports a failed link. Note that this metric is meaningful only for inferred topologies, as redundancy is caused by multiple representations of the same link. The plot reaffirms that the topologies inferred by MN are inappropriate for failure localization. The hypothesis list contains a large amount of redundant information for the few times that MC is able to localize the failed link. In comparison, iTop and Isomap show low redundancy. In particular, the correctness of the merging process of iTop limits multiple representations of the same link in the inferred topology, resulting in the lowest redundancy.

These results show that not only does iTop infer topologies which closely match the ground truth, but it can also be effectively used in real network management applications such as failure localization.

VIII. Related work

Previous works on topology inference are typically based on either network tomography or Traceroute. Network tomography approaches [3], [5], [6] probe the network to collect end-to-end measurements of additive metrics such as delay and packet loss. These approaches do not require any cooperation from internal nodes, as a result they are not negatively affected by the presence of non-cooperative routers. However, these solutions can only infer the *logical routing topology*, which is a simplified representation of the actual network-layer topology. Furthermore, they incur a high communication overhead from their probing [6].

Most Traceroute based techniques [25], [26], [27] address the problem of alias resolution, in which a single router may have multiple interfaces with different IP addresses. These approaches assume that all routers in the network are cooperative and thus do not consider the limitation of partial information caused by the presence of non-cooperative routers.

Recent approaches [28], [29], [30], [31] focus on a variety of problems which may arise by using Traceroute. In particular, the work [28] studies the effect of per-flow load-balancers. Similarly, [29] considers the presence of off-path addresses, while [30] investigates MPLS tunnels obscured in the traces. These approaches also do not consider non-cooperative routers. Nevertheless, their solutions are orthogonal to ours, and can be integrated in iTop.

Other related works [31], [32] adopt Traceroute to determine IPv6 router availability and propose packet-prober mechanism like Scamper [32] for active measurement of the Internet. These approaches do not specifically address topology inference.

Only few works consider the problem of topology inference with partial information [10], [11], [12], [13]. This problem was introduced in [10], but only considered anonymous routers. Blocking routers are addressed in [11]. Recent works on topology inference with partial information either try to reduce the complexity of the inference process [12] or try to supplement Traceroute traces with additional information [13]. In particular, the authors of [12] reduce the complexity by identifying patterns in the traces that indicate certain structures in the real network. While this has lower runtime complexity than previous solutions [10], [11], it provides less accuracy [12]. Finally, [13] supplements Traceroute information with data from IP record route. This results in a more accurate inferred topology, however record routes may not be available and are not standardized.

In this paper, we compared iTop to the works proposed in [10] and [11], since they outperform other approaches in terms of accuracy of the inferred topologies, and do not require additional information.

The problem of network failure localization has been widely studied [33], [34], [35]. These approaches assume complete knowledge of the network topology. Only recently, has the need for proper topology inference algorithms to improve the failure localization been highlighted in [4]. In this paper, we study the effects of non-cooperative routers on fault localization by applying MC [34] to inferred topologies. We show that the topologies inferred by iTop enable MC to achieve performance close to the case in which the full topology is known.

IX. Conclusions

Detailed knowledge of the network topology is the basis of multiple network management tasks. Topology inference techniques have been proposed to derive the network topology from path information collected using the Traceroute protocol. However, often only partial information can be acquired since some routers may not participate in the protocol and instead behave as anonymous or blocking.

In this paper, we propose an algorithm called iTop to infer the network topology in the presence of non-cooperative routers. iTop initially constructs an overestimated virtual topology, which is then refined through merging operations guided by a set of link compatibility rules.

We compare iTop to previously proposed approaches. Experiments show that iTop outperforms existing approaches by providing a more accurate estimation of the real topology. Furthermore, the topologies provided by iTop enable successful localization of faults even when only partial information is available.

References

[1] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes." *IEEE Communications Surveys and Tutorials*, vol. 7, no. 1-4, pp. 72–93, 2005.

[2] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "Detection and localization of network black holes," *IEEE INFOCOM*, pp. 2180–2188, 2007.

[3] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang, "Efficient and dynamic routing topology inference from end-to-end measurements," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 1, pp. 123–135, 2010.

[4] B. Holbert, S. Tati, S. Silvestri, T. La Porta, and A. Swami, "Effects of partial topology on fault diagnosis," *IEEE MILCOM*, 2013.

[5] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast topology inference from measured end-to-end loss," *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 26–45, 2002.

[6] X. Zhang and C. Phillips, "A survey on selective routing topology inference through active probing," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, 2012.

[7] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, 2004.

[8] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *IEEE INFOCOM*, vol. 3, 2000, pp. 1371–1380.

[9] P. Bedford, A. Bestavros, J. Byers, and M. Crovella, "On the marginal utility of deploying measurement infrastructure," Boston University Computer Science Department, Tech. Rep., 2000.

[10] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology inference in the presence of anonymous routers," *IEEE INFOCOM*, 2003.

[11] X. Jin, W.-P. Yiu, S.-H. Chan, and Y. Wang, "Network topology inference based on end-to-end measurements," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2182–2195, 2006.

[12] M. H. Gunes and K. Sarac, "Resolving anonymous routers in internet topology measurement studies," in *IEEE INFOCOM*, 2008.

[13] R. Sherwood, A. Bender, and N. Spring, "Discarte: A disjunctive internet cartographer," *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, pp. 303–314, 2008.

[14] V. Jacobson, "Traceroute software," *Lawrence Berkeley Lab*, 1998.
[15] M. Luckie, Y. Hyun, and B. Huffaker, "Traceroute probe method and forward ip path inference," *ACM SIGCOMM conference on Internet measurement*, pp. 311–324, 2008.
[16] P. Mahadevan *et al.*, "The internet as-level topology: three data sources and one definitive metric," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 17–26, 2006.
[17] B. Holbert, S. Tati, S. Silvestri, T. La Porta, and A. Swami, "Network topology inference with partial path information," *IEEE ICNC*, 2015.
[18] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[19] M. Newman, *Networks: an introduction*. Oxford University Press, 2010.
[20] U. of Washington, "Rocketfuel: An isp topology mapping engine," 2002. [Online]. Available: http://www.cs.washington.edu/research/networking/rocketfuel/
[21] T. C. A. for Internet Data Analysis (CAIDA), "Macroscopic internet topology data kit (itdk)," 2013. [Online]. Available: http://www.caida.org/data/active/ internet-topology-data-kit/
[22] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," *AAAI Conference on Weblogs and Social Media*, 2009.
[23] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," *IEEE INFOCOM*, vol. 4, pp. 2307–2317, 2004.
[24] S. Tati, S. Rager, B. J. Ko, G. Cao, A. Swami, and T. La Porta, "Netcsi: A generic fault diagnosis algorithm for large-scale failures in computer networks," *IEEE SRDS*, pp. 167–176, 2011.
[25] B. Cheswick, H. Burch, and S. Branigan, "Mapping and visualizing the internet," *USENIX*, 2000.
[26] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *ACM SIGMETRICS*, 2005, pp. 327–338.
[27] J. Ni, H. Xie, S. Tatikonda, and R. Y. Yang, "Efficient and dynamic routing topology inference from end-to-end measurements," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 123–135, 2010.
[28] M. Luckie, A. Dhamdhere, k. claffy, and D. Murrell, "Measured Impact of Crooked Traceroute," *ACM SIGCOMM Computer Communication Review*, 2011.
[29] M. Luckie and k. claffy, "A Second Look at Detecting Third-Party Addresses in Traceroute Traces with the IP Timestamp Option," *Passive and Active Measurement Whowrkshop*, 2014.
[30] B. Donnet, M. Luckie, P. Mrindol, and J. Pansiot, "Revealing MPLS tunnels obscured from traceroute," *ACM SIGCOMM Computer Communication Review*, 2012.
[31] R. Beverly, M. Luckie, L. Mosley, and k. claffy, "Measuring and Characterizing IPv6 Router Availability," *Passive and Active Measurement Whowrkshop*, 2015.
[32] M. Luckie, "Scamper: a scalable and extensible packet prober for active measurement of the internet," *ACM SIGCOMM*, 2010.
[33] S. Kandula, D. Katabi, and J.-P. Vasseur, "Shrink: A tool for failure diagnosis in ip networks," in *MineNet*, 2005.
[34] R. R. Kompella, J. Yates, A. Greeberg, and A. C. Scnoeren, "Ip fault localization via risk modeling," in *NSDI '05: Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation*, 2005, pp. 57–70.
[35] S. Tati, B. J. Ko, G. Cao, A. Swami, and T. La Porta, "Adaptive algorithms for diagnosing large-scale failures in computer networks," *IEEE DSN*, pp. 1–12, 2012.

Brett Holbert received his B.S. in Computer Science from University of Maryland, College Park in 2010. He received his M.S. in Computer Science at the Pennsylvania State University, University Park, PA. As a member of the Institute for Networking and Security Research his research focuses on topology inference in partially responding computer networks. Additional research interests include network fault diagnosis and failure recovery.

Srikar Tati received the bachelors of technology (B Tech Hons.) degree at the Indian Institute of Technology, Kharagpur in 2006, the masters of Sciences (MS) degree at Pennsylvania State University in 2008, and the PhD degree at Pennsylvania State University. He is currently working in Ericsson Silicon Valley. His research interests include broad areas of design, modeling and analysis of network systems, large distributed systems, wireless networks etc. As part of his dissertation, he worked on various problems that arise due to challenges in failure and performance diagnosis algorithms in computer networks. He previously worked on few problems in wireless networks. He published his work in conferences such as IEEE ICDCS, DSN, ICNP, SRDS etc.

Simone Silvestri graduated with honors and received his PhD in computer science at Sapienza University of Rome. He is now an Assistant Professor at the Computer Science Department of Missoury University of Science and Technology. Before joining MS&T, he was a Post-Doctoral research associate at the Computer Science and Engineering Department of Pennsylvania State University. He was also a visiting scholar at the Electrical and Electronic Engineering Department, Imperial College, London. He served as program committee member of several international conferences. His research interests lie in the area of network management, sensor networks and interdependent networks.

Thomas F. La Porta is a Distinguished Professor in the department of computer science and engineering at Penn State University, where he is the Director of the Institute of Network and Security Research. Prior to joining Penn State in 2002, he was with Bell Laboratories since 1986 as Director of the Mobile Networking Research Department. He is an IEEE Fellow, Bell Labs Fellow, received the Bell Labs Distinguished Technical Staff Award in 1996, and an Eta Kappa Nu Outstanding Young Electrical Engineer Award in 1996. He was the founding Editor-in-Chief of the IEEE Transactions on Mobile Computing, and served as Editor-in-Chief of IEEE Personal Communications Magazine. His research interests include mobility management, signaling and control for wireless networks, mobile data and sensor systems, and network security.

Ananthram Swami received the B.Tech. degree from Indian Institute of Technology (IIT), Bombay; the M.S. degree from Rice University, Houston, TX, and the Ph.D. degree from the University of Southern California (USC), Los Angeles, all in electrical engineering. He has held positions with Unocal Corporation, USC, CS-3, and Malgudi Systems. He was a Statistical Consultant to the California Lottery, developed a Matlab-based toolbox for non-Gaussian signal processing, and has held visiting faculty positions at INP, Toulouse, France, and Imperial College, London. He is with the U.S. Army Research Laboratory (ARL) where he is the ST for Network Science. His work is in the broad area of network science, with emphasis on wireless communication networks.