Computer Networks 81 (2015) 178–200

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Graph Based Induction of unresponsive routers in Internet topologies

Hakan Kardes [a], Mehmet Hadi Gunes [b,*], Kamil Sarac [c]

[a] Inome Inc., United States
[b] University of Nevada Reno, United States
[c] University of Texas at Dallas, United States

ARTICLE INFO

Article history:
Received 8 July 2013
Received in revised form 27 December 2014
Accepted 16 February 2015
Available online 23 February 2015

Keywords:
Internet topology measurement
Internet mapping
Unresponsive router resolution

ABSTRACT

Internet topology measurement studies utilize traceroute to collect path traces from the Internet. A router that does not respond to a traceroute probe is referred as an unresponsive router and is represented by a '*' in the traceroute output. Unresponsive router resolution refers to the task of identifying the occurrences of '*'s that belong to the same router in the underlying network. This task is an important step in building representative traceroute-based topology maps and obtaining an optimum solution, i.e., minimal graph under trace and distance preservation conditions, is shown to be NP-complete. In this paper, we first analyze the nature of unresponsive routers and identify different types of unresponsiveness. We also conduct an experimental study to understand how the responsiveness of routers has changed over the last decade. We then utilize a novel graph data indexing approach to build an efficient solution to the unresponsive router resolution problem. The results of our experiments on both synthetic and sampled topologies show a significant improvement in accuracy and effectiveness over the existing approaches. Our experiments also demonstrate that applying IP alias resolution along with unresponsive router resolution results in a final topology closer to the original topology.

1. Introduction

Internet has become an important part of our daily interactions and keeps expanding around the world. As the largest man-made complex network, it has been growing with no central authority. Each network is built by a different AS for possibly different purposes, e.g., small local campuses to large transcontinental backbone providers. As each AS grows its own network based on local economic and technical objectives, a single or few ASes are not representative of the Internet.

Understanding Internet topology is valuable for commercial, social, and technical purposes. Knowledge of the network graph helps in understanding the large scale characteristics and dynamics of the Internet [1]. For instance, such graphs are needed in analyzing the topological characteristics of the Internet and designing topology generators that can produce realistic synthetic topologies [2]. Researchers have also indicated that the deployment of networks by content providers has a flattening effect on the hierarchical AS structure [3]. Knowledge of the underlying networks help in path exploration using new approaches [4] and optimize data delivery [5]. Additionally, analysis of the Internet topology is useful to develop failure detection measures, network planning, and optimal routing algorithms [6]. Researchers test new protocols and systems using simulations or emulations

* Corresponding author.
E-mail addresses: hkardes@cse.unr.edu (H. Kardes), mgunes@unr.edu (M.H. Gunes), ksarac@utdallas.edu (K. Sarac).

http://dx.doi.org/10.1016/j.comnet.2015.02.012
1389-1286/

[7,8], but more realistic results can be obtained when real topologies are utilized in the analysis [9,10].

Internet service providers (ISPs) keep their topology information confidential due to commercial and security reasons. This policy introduces a practical challenge for the research community and requires them to generate measurement probes to sample the Internet topology at various levels, including IP, link, router, point of presence (PoP), and autonomous system (AS) levels. Most measurement studies utilize *traceroute* [11] or its variations to collect a large number of path traces from topologically diverse set of vantage points. Traceroute relies on the TTL expiration mechanism of routers to elicit responses from routers between source and destination (see RFC 1393: Traceroute Using an IP Option).

After collecting the path traces, the information needs to be processed to build the corresponding topology map. In particular at link, router and PoP levels, we need to *(i)* verify correctness of the collected paths [12], *(ii)* identify underlying tunnels [13], *(iii)* resolve IP addresses belonging to the same router [14,15], *(iv)* infer IP addresses that are connected over a common subnet link [16,17], and *(v)* resolve unresponsive routers that are represented by '*'s in traceroute outputs [18]. The first task is due to the fact that certain traffic engineering practices may cause traceroute to return IP addresses that do not correspond to a real end-to-end path in the network. Augustin et al. [19] propose *Paris traceroute* tool to address this problem. The second task helps in identifying and addressing inaccuracies due to tunneling techniques such as MPLS. Researchers have developed approaches to identifying underlying tunnels [20,13]. The third task is due to the fact that routers have multiple IP addresses that may appear in different path traces. Without any resolution, each IP address is treated as a potentially different router in the initial graph and IP alias resolution is applied to identify the IP addresses which belong to the same router. Several tools exist to resolve alias IP addresses [14,15,21]. The third task, subnet resolution, identifies subnet relations among IP addresses and reveals connectivity that is not observed in collected path traces [16,22,17]. The last task emerges due to the fact that not all routers respond to traceroute probes during topology collection and is the main focus of this paper.

All of these tasks are important for obtaining accurate samples of the underlying network. The accuracy and completeness of these tasks significantly affects the accuracy of the resulting topology maps [23–25]. Hence, topology measurement studies should handle these tasks to obtain representative topology maps. Moreover, when handling these tasks, one needs to make decisions based on observations to infer the underlying topology. As the earlier decisions affect the later ones, obtaining the most likely topology under various conditions has been shown to be NP-hard [26]. Several approaches have been proposed to reduce the set of hypotheses in the decision making of the resolution tasks [27,18,14,28,20].

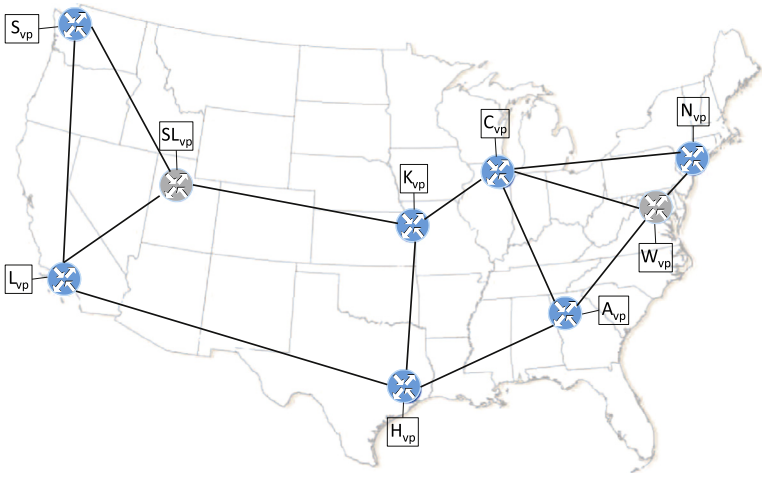
A router that is unresponsive to measurement probes is represented by a '*' in traceroute output and the same router may be probed multiple times in different traces. Even though routers are expected to generate an ICMP message indicating TTL expiration, it is not mandated (see RFC 1812: Requirements for IP Version 4 Routers). Moreover, different protocols have different success rates in eliciting a response [29]. Without any resolution, each occurrence of '*' is treated as a potentially different router in the graph. Unresponsive router resolution task focuses on identifying the '*'s belong to the same unresponsive router. In this paper, we use the term *unknown node* to refer to a '*' in the traceroute output and *unresponsive router* to refer to the actual router that is represented by this unknown node (i.e., by this '*'). Similarly, we use the term *known node* to refer to an observed IP address in the traceroute output and *responsive router* to refer to the actual router that is represented by this known node (i.e., by the IP address(es)).

Depending on the number of unresponsive routers and the topology collection scenario, the collected set of path traces may include a large number of '*'s. For instance, let us consider the Internet2 backbone presented in Fig. 1a where squares represent the vantage points attached to each router shown as a circle. Assume that the routers at *Washington, DC* and *SaltLake* are configured to be unresponsive and path traces are collected between all vantage points. Under these assumptions, the topology that is constructed from the 36 path traces between 9 vantage points would be as in Fig. 1b (with no resolution). The common approach of pruning (see Section 4 for details) will produce a graph as seen in Fig. 1c and leave artificial nodes that needs to be further processed.

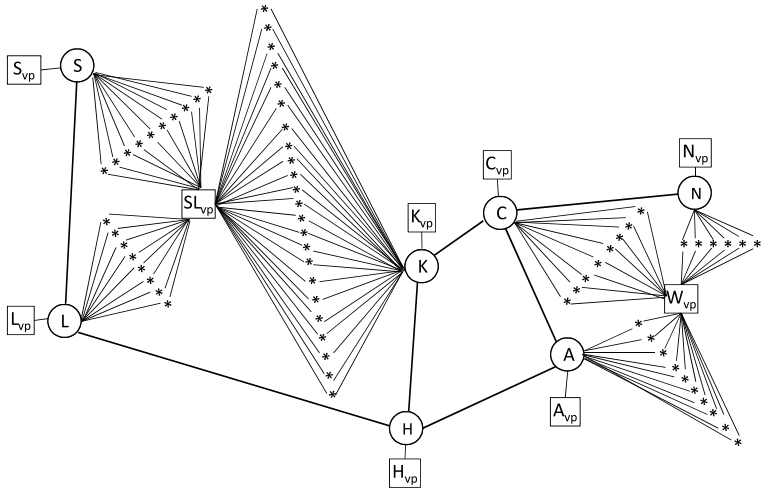
In this paper, we first analyze the nature of unresponsive routers and identify different types of unresponsiveness. We observe that temporary unresponsiveness is an important case that is omitted in previous studies. We also investigate the responsiveness of routers to active network measurements. Expanding our earlier work [29] on skitter [30] data sets, we present the prevalence of unresponsive routers in historical Ark [31] and iPlane [32] measurement data sets.

We then introduce a *Graph Based Induction* (GBI) approach, based on our earlier work [18], to resolve unresponsive routers in traceroute based topology mapping studies. In our work, we define a new induction approach from the practical context of the unresponsive router resolution problem and develop an efficient implementation. To this end, we first examine topology maps that are constructed from traceroute data with unresponsive routers and identify a number of graph structures that are formed among unknown nodes and their known neighbors. We then use *Structural Graph Indexing* (SGI) to detect these structures in the graph and reduce the unknown nodes (i.e., the occurrences of '*'s) into their corresponding unresponsive routers. We enhance our SGI [33] approach to efficiently query constructed topology graphs.

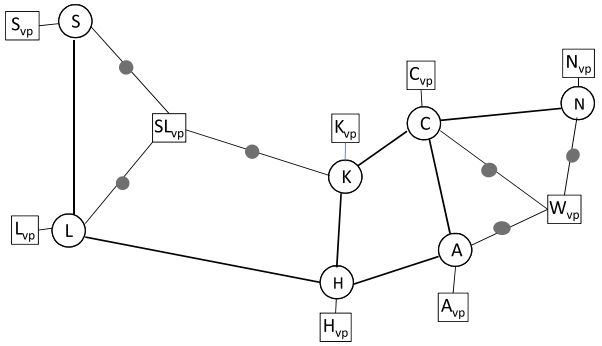
In our evaluations on synthetic topologies, we observe that GBI has better resolution than previously proposed Initial Pruning [34] and Neighbor Matching [35] approaches. Due to their high algorithmic complexity (see Section 2), we did not compare GBI with graph minimization [36], dimensionality reduction [37], and spectral



(a) Internet2 Backbone (grey routers are unresponsive)



(b) Induced Topology (traces between all vantage points)



(c) After Initial Pruning (of unresponsive nodes)

Fig. 1. Sample network.

embedding [38] approaches. We also showed that proper resolution of IP aliases improves the unresponsive resolution. In addition, we demonstrate the feasibility of our approach on datasets collected by *(i)* iPlane with 33 M traces, 9 M unresponsive and 300 K known nodes, *(ii)* Ark with 22.7 M traces, 11.4 M unresponsive and 1.2 M known nodes, and *(iii)* Cheleby [39] with 15 M traces, 7.2 M unresponsive and 1.2 M known nodes.

According to our study, the large volume of unresponsive routers in collected Internet graph significantly affects graph characteristics such as degree distribution, and clustering coefficient of the graph (see Section 5.1). In order to build more representative Internet maps, unresponsive routers should be properly resolved. However, the massive volume of unknown nodes in the collected data set introduces challenges in building efficient solutions. For instance, the iPlane datasets we use in our evaluations in Section 3.2 has almost 67% of nodes as unknown in average. Hence, timely analysis of large scale data sets, currently tens of millions of traces, requires efficient algorithms.

The rest of this paper is organized as follows: Section 2 summarizes the related work. Section 3 presents an experimental study to understand the responsiveness of routers to active probing. Section 4 introduces our *Graph Based Induction* approach. Section 5 presents our experimental evaluations. Finally, Section 6 concludes the paper.

2. Related work

2.1. Router unresponsiveness

Active probing has increasingly been used to observe different characteristics of the underlying network. Based on the increasing need for active measurements, the research community has developed large scale distributed measurement platforms (e.g., skitter [30], Ark [31], iPlane [32], Dimes [40], Cheleby [41], DipZoom [42], PlanetLab [8], and TraceNET [43]) that are commonly used to conduct measurement activities.

Most topology measurement studies utilize the well-known Internet debugging tool *traceroute* [11] or its variants [19,44] to collect a large number of path traces from a topologically diverse set of vantage points. Traceroute returns a path from a local system to a given remote system by tracing the routers in between. It uses TTL-scoped TCP, UDP or ICMP based probe packets to elicit ICMP error messages from the routers on the path. By collecting the source IP addresses from the incoming ICMP packets, traceroute returns the path as a sequence of IP addresses each representing a router between the local system and the remote destination.

Luckie et al. analyzed router responsiveness of TCP, UDP, and ICMP based traceroute methods [45]. They found that reachability of ICMP-based traceroute methods is higher. ICMP-based methods also collect evidence of a greater number of AS links. On the other hand, UDP-based methods infer the greatest number of IP links, despite reaching the fewest destinations. We had a similar analysis in [29] that is updated in Section 3. In [29], we conducted an experimental study to understand the responsiveness of routers to active probing both from a historical perspective and current practices. Our historical analysis revealed that router and end-system responsiveness reduced during the last decade. We also observed that ICMP based probes elicit the highest response rate while UDP based ones elicit the lowest.

Broido et al. report that around 1/3 of probed paths contain unresponsive, private, or invalid routers [46]. Most of the early work in the area ignored this problem or used simple heuristics to work around it. Cheswick et al. avoid the problem by stopping a trace toward a destination on encountering an unresponsive router on the path [47]. Thus, in this approach, they simply ignored the unresponsive router problem. However, this causes loss of potentially useful connectivity information as the path after an unresponsive router is ignored. Broido et al. handle unresponsive routers by replacing them either with arcs (to connect the known routers at two ends) or with unique identifiers to treat them as separate nodes [46]. This approach also results in either loss of path information from the probe result or inaccuracies in the resulting topology maps. Bilir et al. merges same length sequence of unknown nodes between the same pair of known nodes with each other [34] (called as *Initial Pruning* in Section 5.1). This approach provides limited resolution in the resulting topology maps. Hence, considerable amount of unresponsive routers remains unresolved.

Yao et al. formulate the unresponsive router resolution problem as an optimization problem [36]. Their goal is to build a minimum size topology by merging unknown nodes under two conditions: (i) *trace preservation*, i.e., there should not be any routing loop due to merging two unknown nodes and (ii) *distance preservation*, i.e, the unresponsive router resolution process should not reduce the length of a shortest path between any two nodes in the resulting topology map. They prove that the optimum topology inference under these conditions is NP-complete and propose a heuristic to minimize the constructed topology by identifying unknown nodes that, when merged, satisfy both conditions. They evaluate their approach on a sample 6Bone topology with only 1351 routers with few of them set as unresponsive. The main limitation of this approach is that it is not feasible for current data sets due to its high algorithmic complexity of $O(n^5)$ where n is the number of unknown nodes. Additionally, the claimed distance preservation condition does not reflect the routing practices where the observed paths are not always the shortest due to peering agreements between ASes. Moreover, as the authors indicate, this paper made the simplifying assumption that a router cannot have both a known and an unresponsive interface address. While this assumption was consistent with their experiment in the context of the 6Bone, actual routers may behave unresponsive through certain interfaces or under certain conditions (see Section refsec:types for details).

Later, Jin et al. introduced a dimensionality reduction approach that uses link delays or node connectivity as attributes in the dimensionality reduction process [37]. The main limitation of this approach is also that it is not feasible with an algorithmic complexity of $O(n^3)$ where n is the number of nodes. In addition, the link delay based approach is not practical as it ignores the difficulty of estimating individual link delays from round trip delays in path traces [35]. The authors also propose a simple neighbor matching heuristic with a smaller time complexity, i.e., $O(n^2)$. As the authors indicate in their paper, this approach introduces a higher rate of false positives and false negatives (see Section 5.1 for details).

Almog et al. proposed semi-supervised spectral embedding of all nodes followed by clustering of the unknown nodes in the projected space [38]. As the authors indicate, the approach should be improved to be practical in large-scale data sets. Moreover, it assumes that there is only one unresponsive router between two known routers. According to our analysis in Table 2, however, there are considerable amount of unresponsive router chains and the number of these chains has been increasing in recent years. Additionally, they use a *k*-means clustering approach to cluster the unresponsive routers which requires the knowledge of the number of unresponsive routers in the graph. Authors approximate the number of the unresponsive routers in the graph but, according to our study in Section 3, the number of unresponsive routers and their distribution depends on the topology collection approach. Thus, a general approximation technique which will work for different topology maps is a challenge for this approach.

Recently, Du et al. [48] developed an approach based on the results of single-source traceroute under the assumptions that every router supports source routing, every traceroute packet passes through a same path each time, and all the connections have the same bandwidth. The main weakness of this approach is that it is based on single-source trace route as topology collection systems use multiple vantage points.

Finally, Pignolet et al. [49] theoretically investigated the topology inference with unresponsive routers. Authors focused on the network properties of inferrable topologies using pairwise end-to-end measurements. They present a formal theory of topology inference by proposing basic axioms (i.e., assumptions on the trace set) that are used to guide the inference process without the cooperation of some nodes along these paths. Overall, they prove that (1) unresponsive nodes makes the topology inference process extremely difficult and (2) having complete set of path traces is helpful with correct inference of only global graph characteristics.

In this paper, we propose an approach that combines theoretical approach of graph minimization and clustering approach of data mining in Section 4.

2.2. Graph data mining

Graph data mining techniques are utilized in many application domains [50–53] to extract useful information from the graph representation of large data sets [54]. The need for mining large graphs in an efficient manner increases as researchers look into complex networks. Several studies have been carried out to make graph mining in an efficient manner using indexing techniques. Among various graph mining techniques, graph-based induction (GBI) technique shows similarity to our problem. Graph-based induction is a technique to find frequent substructures, a common problem in biological and chemical networks, [55]. It extracts typical patterns by stepwise identification of recurring node pairs and minimizes the graph size by replacing identified patterns with a node. In our work, we propose a similar induction technique to identify subgraphs of some common structures within the topology graph and use them to resolve unresponsive routers.

Graph indexing studies can be mainly divided as *path-based* and *structure-based* approaches. Path-based graph indexing approaches use path expressions as indexing features such as GraphGrep [56] and Daylight [57]. GraphGrep enumerates all paths in the graph up to the length *maxL*. Then, it looks for the graph whether it contains all paths up to *maxL* for a graph query q_i. A significant feature of the path-based approaches is that paths can be manipulated easier than general graphs. However, path is a simple structure loosing structural information of a graph, and hence false positive ratio of path-based methods might be very high [58]. In addition, the number of paths in a graph database increases exponentially, making path-based methods impractical for very large graphs. Alternatively, structure-based graph indexing approaches identify subgraphs to be indexed as in gIndex [58]. gIndex first searches for the frequent subgraphs in the graph, then indexes these frequent structures. An issue in this case is that frequent subgraph discovery increases complexity and exponential number of frequent fragments may exist under low frequency support. Therefore, in their study, authors limit the number of nodes to 10.

In this paper, we utilize a novel *Structural Graph Indexing (SGI)* approach to search and process queries efficiently even in very large graphs [33]. As indexing features, we use commonly observed graph structures: star, complete bipartite, and triangle.

3. Router responsiveness analysis

In this section, we analyze responsiveness of routers and end hosts to active probes from historical data sets. We assume a *trace* visits different vertices v_k in the underlying Internet graph and returns a list of *nodes* representing the interfaces (one for each vertex) as its output, i.e., $trace(v_i, v_j) = (i_x^{v_i}, \ldots, i_e^{v_p}, i_f^{v_r}, \ldots, i_y^{v_j})$. If a router $v_p \in trace(v_i, v_j)$ is an unresponsive router, the *trace* output will have a *unknown node*, $*_e^{v_p}$, instead of a *known node*, $i_e^{v_p}$. Note that IP alias resolution, i.e., identification of IP addresses belonging to the same router, should be handled during the resolution process as well [14,15,21]. In this study, we focus on resolving unknown nodes (e.g., $*_f^{v_p}$) in addition to known nodes (e.g., $i_e^{v_p}$) that are observed due to a single router v_p.

3.1. Unresponsive router types

In this section, expanding on our earlier work [29], we investigate router (un)responsiveness to active probes. As in Fig. 2 active probes are divided into two categories as (i) *direct probes* where the destination IP address in the probe packet is the intended destination as in ICMP ping or (ii) *indirect probes* where the destination IP address in the probe packet is some other destination as in traceroute when it probes an intermediate router during a trace.

In both direct and indirect probing, eliciting a packet from a probed node indicates the responsiveness of the node. On the other hand, not receiving a response to an

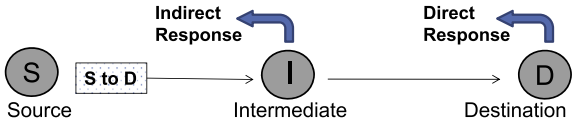


Fig. 2. Active probing.

active probe packet may have different interpretations. In the case of direct probing, the lack of a response message may not necessarily indicate node unresponsiveness as it may be that the probed node may be unreachable; may be disconnected or turned off; or either probe or the response packet may be filtered/dropped/lost in the network. In the case of indirect probing as in traceroute, the lack of a response message, in general, indicates node unresponsiveness if another responsive node appears later on within the same trace.

We classify unresponsive routers into two categories:

Permanent unresponsive: A router might be permanently unresponsive under two conditions. First, a system may be configured to ignore certain probe packets causing it to be unresponsive with such probing. Second, a border router may be configured to filter out *(i)* certain types of packets such as unsolicited UDP packets directed to a local host or *(ii)* outgoing ICMP responses originating from nodes within its local domain. Border filtering causes internal nodes to be seen as unresponsive.

Temporary unresponsive: A router might be temporarily unresponsive under several conditions. A system may apply ICMP rate limiting and become unresponsive if the rate of the incoming probes exceed a preset limit. Similarly, a system may ignore probe packets when it is congested but respond to them when it is not. Finally, packets may occasionally be dropped or lost due to routing or overflow. In either case, the router has altering responsiveness.

Moreover, a system may have a private (i.e., publicly unroutable) IP address that cannot guarantee node uniqueness. Such nodes can be either marked per AS they originate from or marked as an unknown node. Since the number of such nodes were very small in iPlane and Caida datasets, we excluded them from our study.

3.2. Historical data analysis

In this section, we use traceroute collected historical data sets to study router reaction to indirect probe messages. We downloaded publicly available historical traceroute data sets from CAIDA (collected by Skitter [30] and Ark [31] systems), and iPlane [59]. We utilized data sets that were collected during a single collection cycle in July of each year by the corresponding infrastructure (typically two days of measurement for CAIDA and one day of measurement for iPlane). Note that we discard the inaccurate path traces,i.e., path traces with loops, from the datasets. Additionally, we ignore unknown nodes at the end of traces as they might be due different reasons such as an unresponsive destination, a filtering firewall/gateway, or non-existing destination.

First, we look for a trend in the ratio of unknown nodes in the collected data in Table 1. In this table, *Srcs* shows the number of sources, *Traces* shows the number of traces, *Completed Traces* gives the percentage of traces that reached the final destination; *IPs* gives the number of known nodes (interfaces) within the data set before IP alias resolution; and *Unknown* gives the percentage of unknown

Table 1
Analysis of historical responsiveness.

		Initial					Unres. type	
Dataset	Year	# Srcs	# Traces (M)	Completed traces (%)	# IPs (M)	Unknown (%)	Perm. (%)	Temp. (%)
Skitter (CAIDA)	1999	5	3.5	86.5	0.2	37.2	100	0.0
	2000	14	14.8	83.5	0.7	45.1	100	0.0
	2001	17	13.4	73.6	2.1	42.2	100	0.0
	2002	20	19.1	50.4	1.5	33.9	100	0.0
	2003	23	24.3	54.3	1.9	29.6	100	0.0
	2004	23	22.9	53.0	2.4	39.1	100	0.0
	2005	22	21,0	46.4	6.8	81.9	96.9	3.1
	2006	19	18.4	37.2	6.4	81.5	96.5	3,5
	2007	18	17.5	30.6	4.9	84.8	95.6	4.4
	2008	11	10.7	23.2	2.8	76.8	92.8	7.2
Ark (CAIDA)	2007	8	6.4	3.11	0.69	9.0	86.8	13.2
	2008	23	14.0	6.72	0.95	11.9	86.2	13.8
	2009	36	22.2	7.51	1.16	15.6	88.7	11.3
	2010	43	24.5	8.71	1.22	18.8	89.8	10.2
	2011	53	28.5	9.41	1.30	19.7	88.6	11.4
	2012	53	29.7	10.52	1.39	19.8	87.4	12.6
	2013	66	30.6	12.45	1.37	21.9	82.7	17.3
iPlane	2006	190	17.1	79.4	0.21	60.1	63.2	36.8
	2007	187	20.0	66.6	0.24	64.0	67.4	32.6
	2008	180	24.0	68.3	0.29	74.3	80.3	19.7
	2009	200	27.0	60.0	0.28	71.5	78.0	22.0
	2010	239	34.8	51.8	0.29	70.1	63.3	36.7
	2011	201	29.1	51.4	0.29	69.9	65.1	34.9
	2012	147	18.6	70.2	0.31	61.6	66.0	34.0
	2013	143	20.7	65.9	0.34	63.5	65.0	35.0

nodes (interfaces) in the original data set with partial unresponsive router resolution as explained below. The next two columns give the classification of unknown nodes as percentage values.

In this table, we assume *Initial Pruning* is performed. That is, unknown nodes between the same known nodes are considered as the same. Suppose we have the following three traces:

A — B — X — ∗ — Y — C
D — E — X — ∗ — Z — ∗ — F
G — X — ∗ — Z — H — F

In these two traces, we have 19 responses (A, B, X, ∗, Y, C, D, E, X, ∗, Z, ∗, F, G, X, ∗, Z, H, F), and 15 unique nodes (A, B, X, $*_1$, Y, C, D, E, $*_2$, Z, $*_3$, F, G, $*_4$, H). For above traces, $*_2$ and $*_4$ are considered as same since both appear between *X*, and *Z*. So, *unknown ratio* is 21.4% (3/14 ∗ 100).

Additionally, for our analysis, we mark an unknown node (e.g., $*_3$) between two known nodes (i.e., Z and F) as temporarily unresponsive when we observe a parallel trace with an IP address (i.e., H) between the same pair of nodes (i.e., Z and F). This classification is approximate as it is difficult to know the actual cause of the lack of response to a probe packet from a remote system.

According to iPlane data sets, the ratio of path traces reaching their final destination fluctuates over the time. In average, 64% of traces do not reach to their final destination. In skitter data set, this ratio has significantly decreased over the time (87% in 1999, 23.2% in 2008). Although this ratio is relatively low in traces collected with Ark, it increases over the time (3% in 2007, 12% in 2013). Ark performs better than skitter in reaching a targeted IP address as it uses an improved destination list. The decrease in reachability and the low reachability values are most likely caused by the change in the default ICMP behavior by operating systems, proliferation of firewalls, and the inclusion of a destination from each /24 subnet range that might not correspond to a live system. However, path traces not reaching their final destinations contribute little useful information and considerably slow down the probing process.

Moreover, the ratio of unknown nodes has significantly increased for Ark data (9% in 2007 while 22% in 2013). It fluctuates but is relatively high in iPlane data (∼67%). CAIDA reports that skitter had several updates to destination IP address lists and had a major change in its topology collection system in mid 2004 where it utilized dynamic destination lists with increased probing frequency at its monitors. So, for skitter datasets, there are different trends. The average ratio of unknown nodes is ∼38% before 2005, while it is ∼81% after 2005. These high ratios point to the prevalence of unresponsive router resolution to obtain realistic sample topologies.

Another observation from the table is that the ratio of temporarily unresponsive nodes is not negligible (∼13% in Ark, while ∼31% in iPlane). Yet, they were ignored by all of the previous unresponsive router resolution approaches.

Next, we are interested in the length distribution of path segments formed by consecutive '*'s in path traces. We call a path segment in the form of a $(IP_1, *_1, *_2, \ldots, *_l, IP_2)$ a **∗-subpath** of length l and denote it with $u_{(IP_1,IP_2,l,A)}$ where A is the set of nodes between IP_1 and IP_2, i.e., $\{*_1, *_2 \ldots, *_l\}$. We are interested in the frequency distribution of *-subpaths with respect to their length l. Although a *-subpath of length one may have different interpretations about the cause of router unresponsiveness, occurrence of longer *-subpaths may be an indication of ISP policy of preventing active probing in its network. Moreover, resolution of unresponsive routers in such cases is much harder. It is easier to correctly resolve an unresponsive router when all of its neighbors are responsive. However, when there are neighboring unresponsive routers, deciding which router an unknown node belongs to is challenging (as explained in Section 5.1) and is where most of the resolution algorithms, including GBI, introduce false positives.

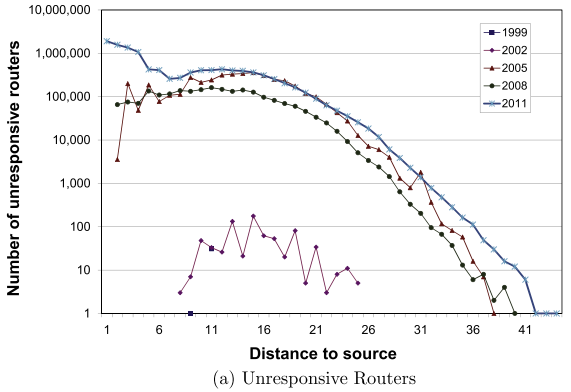
Table 2 represents the *-subpath distribution for the same CAIDA and iPlane datasets. In the table, we present

Table 2
*-Subpath characteristics.

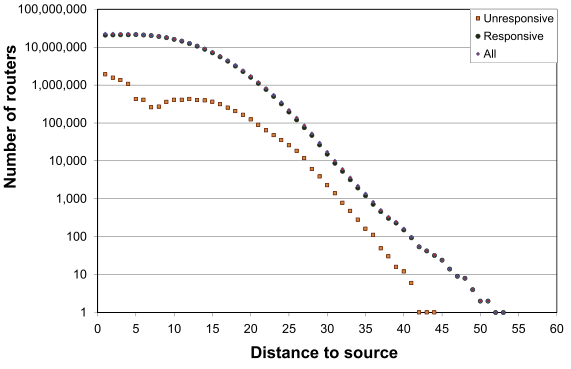
		Unique		*-Subpath length					
Dataset	Year	*-Subpaths	Same AS (%)	1	2	3	4	5	>5
Skitter (CAIDA)	2005	225,456	12.6	151,133	63,662	6360	4301	–	–
	2006	207,067	11.6	137,829	59,171	5828	4239	–	–
	2007	305,331	14.4	212,263	73,263	14,019	5779	7	–
	2008	231,633	14.0	148,182	63,944	13,733	5772	2	–
Ark (CAIDA)	2009	124,700	28.7	71,120	30,071	14,191	9318	–	–
	2010	159,151	26.5	89,196	34,650	24,547	10,758	–	–
	2011	178,095	25.2	98,085	42,725	25,166	12,119	–	–
	2012	203,221	25.3	120,757	44,817	24,462	13,185	–	–
	2013	235,467	25.1	142,942	50,707	26,551	15,267	–	–
iPlane	2006	472,656	25.6	330,672	100,198	40,536	825	269	155
	2007	731,077	25.9	510,186	215,924	2941	1327	441	242
	2008	931,880	23.3	530,698	216,452	110,300	71,097	1313	1809
	2009	840,132	23.1	500,488	180,283	92,988	64,837	781	728
	2010	844,712	22.7	629,468	207,725	3651	1781	1163	833
	2011	851,404	23.0	622,900	218,128	4942	2579	1312	1502
	2012	850,629	23.5	625,264	216,871	3893	2742	1271	588
	2013	850,629	23.5	625,264	216,871	3893	2742	1271	588

the number of unique *-subpaths in the original data set. For uniqueness, we represent a *-subpath of length l as a triplet (IP_1, l, IP_2) and avoid counting the duplicate triplets of this form. The data sets prior to 2005 for CAIDA have only length $l = 1$ *-subpaths and are not included in the table. Starting 2005, we observe *-subpaths of longer lengths with the majority of *-subpaths being of length 1 or 2. We partly attribute the behavior of routers to changes in the data collection process such as the increased probing rate. Longer *-subpaths might also be due to growth in networks where more hops of an unresponsive AS are traversed or due to increased use of MPLS tunnels. According to the results for the iPlane data sets, $l = 1$ *-subpaths are almost doubled in the last five years. We observe that both the overall number of unresponsive routers and the longer length *-subpaths have increased.

We also look into the percentage of *-subpaths that appear within a single AS in the *Same AS* column. For a given *-subpath, say (IP_1, l, IP_2), we look at the relation between IP_1 and IP_2 and map each such IP address to corresponding AS numbers with AS lookup tool of CYMRU [60]. If the IP addresses share the same AS number, then these IP addresses belong to the same domain and therefore the unresponsive nodes in between most likely belong to the same domain. Given that most *-subpaths are of length 1 or 2 and the probability of two IP addresses being in different ASes is about 75% (according to Table 2), we think that the majority of *-subpaths originated from routers at domain boundaries or exchange points between neighboring ASes.



(a) Unresponsive Routers



(b) Routers for 2011 CAIDA data set

Fig. 3. Historic distance distribution.

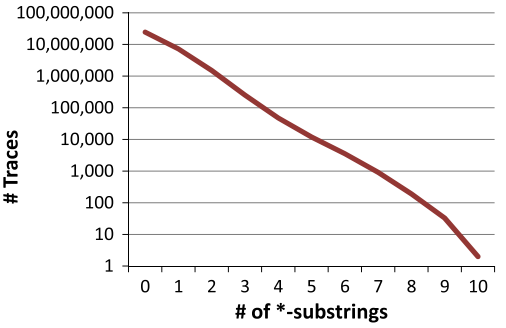


Fig. 4. Distribution of traces with multiple *-subpaths.

We are also interested in the position of unresponsive nodes within path traces and counted the number of unknown nodes at each hop distance from the vantage point. Fig. 3a presents the distance distribution of unknown nodes for five different CAIDA data sets. According to the figure, early data sets (i.e., before 2005) contain small number of unknown nodes that were mostly distributed 10–20 hops away from the source. On the other hand, recent data sets include much more unknown nodes, the majority of which appear 3–25 hops away from the source. The figure also shows a high number of unknown nodes at a distance of 2 from the source for the 2011 data set. A close examination of the corresponding data set shows that this is due to the existence of a permanently unresponsive router at a 2 hops distance to one of the vantage points.

Fig. 3b presents the distance distribution for known, unknown, and all of the nodes for the 2011 CAIDA data set. Since the number of unknown nodes is relatively smaller than the number of known nodes when accounting for each occurrence separately, the distance distribution of all the nodes is quite similar to distance distribution of known nodes. Moreover, unlike the distribution of known nodes, there is a decrease in the number of unknown nodes between distance 1–9. Starting from hop distance 10, they start to exhibit a similar trend.

Fig. 4 presents the distribution of *-subpaths. As seen in the figure, there are considerable number of traces with multiple *-subpaths, which can be up to 10 such regions.

3.3. Load balancing practices

A major issue to keep in mind during topology collection is the effect of load balancing practice of ISPs. Certain traffic engineering practices for load balancing may cause traceroute to return IP addresses that do not correspond to a real end-to-end path in the network [19]. This happens when a router forwards consecutive traceroute probes on different paths toward the destination, a common phenomenon in the Internet [61]. For instance, consider the Internet2 backbone in Fig. 5 where there are two paths between **y** and **z** (underlying links are marked with black lines). When router **S** forwards packets through both paths towards destination **z**, then path traces will

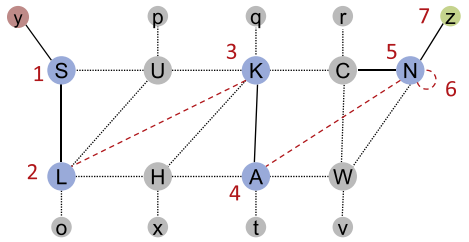


Fig. 5. Effect of load balancing (#s indicate observed hop distance).

include non-existent links (marked with dashed red lines) as we would observe a path of **S–L–K–A–N–N–z** from **s**.

Routers allow configuration of several parameters to determine load balancing including destination IP, source IP, protocol and port number [62,63]. Additionally, routers can be set to randomly send packets as flow preserving parameters consume storage and computation resources. Augustin et al. classify load balancers as *per destination*, i.e., only destination IP affects load balancing decision, *per flow*, i.e., any combination of the parameters are utilized for load balancing decision, and *per packet*, i.e., packets are randomly distributed [19].

In general, topology collection systems have developed several strategies to overcome load balancing effect in the collected topologies. For example, traditional traceroute tools would find accurate links with per destination load balancer but not with per flow and per packet load balancers. *Paris traceroute* fixes flow identifiers so that per flow load balancing routers will choose the same next hop for probe packets toward the same destination [19]. However, *Paris traceroute* can only detect the existence of per packet load balancers on a path and would fail to accurately identify actual links on the balanced paths. Similarly, *sidecar* can detect changes in traversed paths by enabling record route option of probe packets [64]. However, *sidecar* does not provide a method to force routers to use the same path in forwarding probe packets so that accurate path traces could be collected. Cheleby [41] utilizes the *Paris traceroute* to collect path traces and identify per packet load balancers.

3.4. Summary

In summary, in this section, we presented a measurement study to understand the responsiveness of routers to active probing. In our historical analysis, we observed that in general responsiveness reduced during the last decade. We also observed that network operators are increasingly rate limiting ICMP responses. Another observation from our study is that the destination reachability considerably reduced over the time indicating that systems (i.e., routers and end systems) are increasingly unwilling to respond to direct probes.

Based on the findings in the measurement study, in our unresponsive router resolution approach, we address the lack of (i) *temporary unresponsive router resolution* as the ratio of temporarily unresponsive routers have been increasing and (ii) **-subpath resolution* as the length of *-subpaths and the number of longer ones have been increasing. Neither of these cases are properly addressed by the previous resolution studies while both trends are observed to be increasing in recent years.

4. Graph Based Induction for unresponsive router resolution

4.1. Preliminaries

In this section, we present a formal definition of unresponsive router resolution problem. First, we clarify a set of terms/conditions that will help us define the problem. The notations introduced in this section will also be used in the development of the algorithms in the remainder of this section.

Definition (*Router-Level Graph:*). Let $G = (V, E)$ be a router level network graph where V represents the set of vertices (i.e., routers and end-hosts) and E represents the set of edges (i.e., communication links) connecting the vertices in V. Each vertex $v \in V$ has one or more interfaces i_e^v and each interface i_e has at least one $i_e.address$ that corresponds to a globally unique IP address.

For the ease of presentation, we use i_e to represent both an interface and its corresponding identifier, i.e., its IP address.

Definition (*Trace:*). A trace (a.k.a path trace), $trace(v_i, v_j)$, is a path in the graph where the trace visits a vertex, v_k, starting from v_i all the way to v_j and returns a list of *nodes* representing the interfaces (one for each vertex) as its output, i.e., $trace(v_i, v_j) = (i_x^{v_i}, \ldots, i_e^{v_p}, i_f^{v_r}, \ldots, i_y^{v_j})$ where each $v_k \in V$.

If a router $v_p \in trace(v_i, v_j)$ is an unresponsive router, the *trace* output will have a *unknown node*, $*_e^{v_p}$, instead of a *known node*, $i_e^{v_p}$.

Definition (*Alias Set:*). An alias set is a set of known nodes (e.g., $i_e^{v_p}$) and unknown nodes (e.g., $*_f^{v_p}$) that are observed due to a single router v_p.

The alias set of *known nodes* is handled by IP alias resolution process [14,15,65,66,21].

Definition (*Subpath:*). A subpath $u_{(a,b,l,A)}$ is a continuous segment of a trace $(a, v_1, v_2, \ldots, v_l, b)$ where a and b are known nodes, l is the length of subpath between a and b excluding a and b, and A is the set of these nodes where each v_i is a known node (e.g., i_e) or an unknown node (e.g., $*_f$).

Definition (**-Subpath:*). A *-subpath $u^*_{(a,b,l,A)}$ is a subpath $u_{(a,b,l,A)}$ where all the nodes in A are unresponsive. Given a path trace $trace(v_i, v_j) = (i^{v_i}, \ldots, i_a, *_1, *_2, \ldots, *_l, i_b, \ldots, i^{v_j})$, a *-subpath is $u^*_{(i_a,i_b,l,A)} = (i_a, *_1, *_2, \ldots, *_l, i_b)$.

Note that a given trace may have zero or more *-subpaths (as shown in Fig. 4 for a sample data set).

Definition (*l-neighbor:*). Two known nodes are l-neighbor if there are l unknown nodes in between. For example, a and b are l-neighbor of one another, if there is a path $(a, *_1, *_2, \ldots, *_l, b)$ from a to b.

Definition (*Trace Preservation Condition:*). Trace preservation condition serves as an accuracy condition during topology construction. In the context of unresponsive router resolution, it states that if $(*_e, *_f) \in trace(v_i, v_j)$, then $*_e$ and $*_f$ cannot be in the same alias set, i.e., they cannot belong to the same router.

Definition (*Mergeable:*). A set of unknown nodes A are mergeable, i.e., *mergeable*(A), if they (or any other node in their alias sets) do not appear in the same path trace.

Definition (*Conflict set:*). Conflict set is a set of traces that a node appeared in. In order to ensure the trace preservation condition, a node should not be merged with another node if they appear in the same trace. This structure helps us eliminate the need to query all path traces to check if a set of nodes are mergeable during alias and unresponsive router resolution. If the intersection of conflict sets of two nodes is non-empty, these two nodes appear in the same trace(s), i.e., they are not mergeable.

4.2. Methodology

In this section, we present a *Graph Based Induction* (GBI) technique to resolve unresponsive routers that introduce a large number of artificial nodes in traceroute-based topology maps [18]. Given a set of path traces $\bigcup\{trace(v_i, v_j)\}$, and IP address alias sets of *known* nodes from a network graph $G = (V, E)$, unresponsive router resolution problem is to build a graph $\overline{G} = (\overline{V}, \overline{E})$ such that.

- $\overline{V} = \bigcup\{V_{(v_i,v_j)}\}$ and $\overline{E} = \bigcup\{E_{(v_i,v_j)}\}$.
- If $*_e^{v_p} \in trace(v_k, v_l)$ and $*_f^{v_p} \in trace(v_m, v_n)$, then there should be only one unknown node $*^{v_p} \in \overline{V}$ corresponding to *permanently unresponsive* router $v_p \in V$.
- If $*_e^{v_p} \in trace(v_k, v_l)$ and $i_f^{v_p} \in trace(v_m, v_n)$, then there should be only one node $i^{v_p} \in \overline{V}$ corresponding to the *temporarily unresponsive* router $v_p \in V$.
- $\forall *_e^{v_p}$ and $\forall i_e^{v_p}$ of a node $v_p \in \overline{V}$ and a node $v_r \in \overline{V}$, there should at most be one $e_{(v_p,v_r)} \in \overline{E}$.

Note that, we assume that the IP alias resolution process is performed prior to or along with unresponsive router resolution and the following condition is satisfied:

- If $i_e^{v_r} \in trace(v_k, v_l)$ and $i_f^{v_r} \in trace(v_m, v_n)$, then there should be one and only one $v_r \in \overline{V}$.

Multiple network topologies with unresponsive routers can result in the same observed topology. Without knowledge of the underlying topology, it is not possible to prove which one is the correct one. Yao et al. analyzes this issue and proposes to use the minimal topology under trace and distance preservation conditions as the underlying topology for an observed network topology [36]. As there can be multiple topologies that lead to the same observation, researchers in general assume the minimal topology that meets certain criteria to be the correct one. Different from [36], we indicate that distance condition is not necessarily correct as observed paths are known to be non-shortest paths.

Moreover, as shown in [49], minimal topology might be far from the underlying topology. Hence, we focus on the minimality of substructures under the *trace preservation* condition.

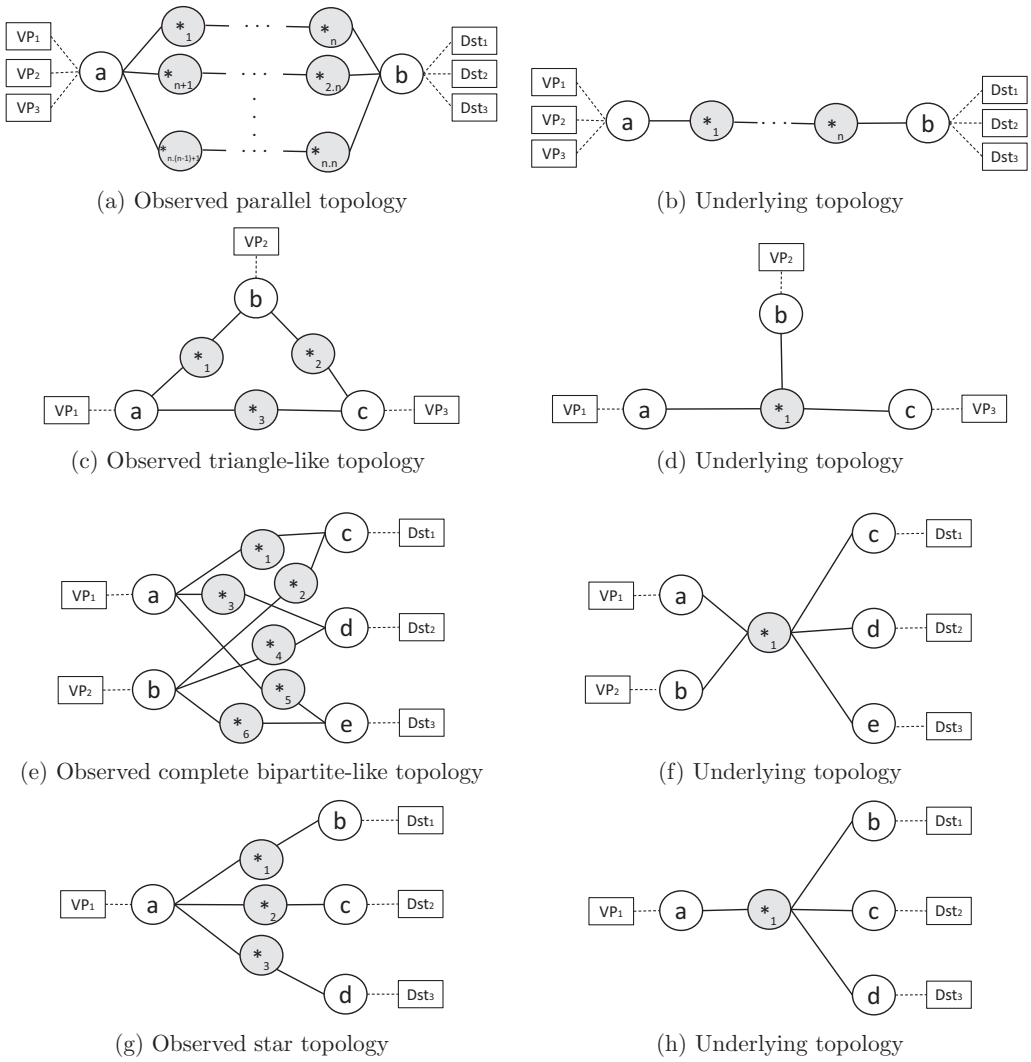
Trace preservation condition requires that there should not be any routing loops in the collected path traces. Path traces with loops may exist due to routing instabilities (for instance, %3.45 of path traces in [41] had routing loops), and they should be filtered prior to the unresponsive router resolution. Moreover, Paris traceroute [19] improves trace accuracy by trying to send probe packets over the same path.

In developing our approach, we analyzed different scenarios involving an unresponsive router and observed how different path traces would lead to artificial nodes. In particular, we analyzed traces from one source to one destination, one source to multiple destinations, multiple sources to multiple destinations, and between all sources. Then we formulated a number of graph structures that are observed in sample topologies. These structures are shown in Fig. 6a, c, e, and g and the corresponding connectivity relations in the underlying actual network are shown in Fig. 6b, d, f, and h respectively. We also investigated raw data and observed evidence to each of these cases.

When there are m traces through a sequence of n unresponsive routers as in Fig. 6b, we observe m parallel unknown sequences as in Fig. 6a. Similarly, when there are traces between all neighbors of an unresponsive router as in Fig. 6d, we observe a clique structure as in Fig. 6c after parallel sequences are resolved. Fig. 6e appears when there are traces from one set, i.e., {a, b}, to the other set, i.e., {c, d, e}, of bipartite graph in Fig. 6f. Finally, Fig. 6g will appear if there are traces from a source node to multiple destinations in Fig. 6h.

In order to address the unknown nodes, we replace the subgraphs Fig. 6a, c, e, and g with the subgraphs in Fig. 6b, d, f, and h within collected topology datasets. Note that networks in Fig. 6b, d, f, and h are the minimum possible underlying networks for the sampled networks shown in Fig. 6a, c, e, and g, respectively.

We modify our *Structural Graph Indexing* technique [33] where we search for structures similar to the identified ones in traceroute-collected topology data and transform unknown nodes in them into their corresponding routers with *Graph Based Induction* approach. In this process, we first start with a router-level topology graph $G = (V, E)$ that is constructed from a set of traceroute-collected path traces. We find all *-subpaths $u^*_{(a,b,l,A)}$ in this graph, and then apply *Structural Graph Indexing* to efficiently perform queries and *Graph Based Induction* to resolve unresponsive routers based on the structures shown in Fig. 6.



(a) Observed parallel topology (b) Underlying topology

(c) Observed triangle-like topology (d) Underlying topology

(e) Observed complete bipartite-like topology (f) Underlying topology

(g) Observed star topology (h) Underlying topology

Fig. 6. Structures (genuine and observed).

In graph indexing, we index predefined structures in the network so that subsequent queries regarding the network graph are more efficient. In our case, we index star, complete bipartite, and triangle structures in a given graph. An important difference of our approach from the previous indexing studies is that we do not limit the size of candidate subgraphs. We try to index maximal graphs that match the structure formulation. For instance, a maximal complete-bipartite is a complete-bipartite that cannot be extended by adding one more vertex from the graph. In order to reduce computational complexities, we index the structures within the original graph in a consecutive manner. That is, we first identify star structures, then complete-bipartite, and finally triangle structures from the preceding ones as detailed below.

4.3. Structural Graph Indexing

In this section, we present each of the identified structures, their underlying topology, and the Structural Graph Indexing of these structures for Graph Based Induction.

4.3.1. Parallel *-subpath structures

First common pattern that we observe in traceroute-based topology maps is the occurrences of same length *-subpaths with the same known nodes at the ends of each *-subpath. While collecting path traces from a vantage point, an unresponsive router may appear as '*' in multiple path traces resulting in multiple parallel *-subpaths between the same known nodes. As an example in Fig. 1a, traceroute queries from A_{vp} to W_{vp} return path traces including *-subpaths such as $(A, *_1, W_{vp})$. Similarly, traceroutes from W_{vp} to A_{vp} result in additional *-subpaths such as $(W_{vp}, *_2, A)$. When path traces between all vantage points are collected, we observe 10 parallel *-subpaths between A and W_{vp} in the resulting topology map as shown in Fig. 1b. Note that in this example *-subpaths include only one unresponsive router. A similar pattern can be observed for *-subpaths of longer lengths as in Fig. 6a.

Resolution of unresponsive routers in this type of structures requires detection of similar *-subpaths (i.e., same

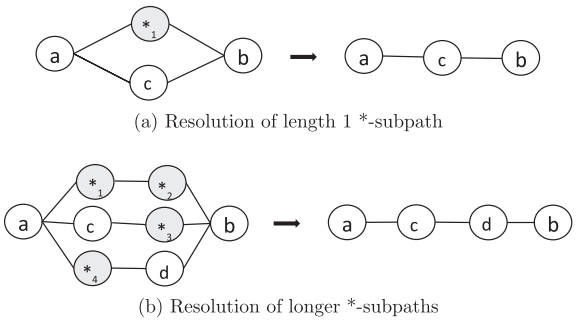
length *-subpaths with the same known nodes at their end points). As described in Section 3.3, certain load balancing practices might cause inaccurate and incomplete path traces in the collected topologies. Hence, we first prune raw path traces. The pruning breaks path traces with a loop (e.g., IP_A, IP_B, IP_C, IP_D, IP_E, IP_C, IP_F, IP_G) into three pieces based on the repeated IP address (i.e., IP_C) and utilize the first part (i.e., IP_A, IP_B, IP_C) and the last part (i.e., IP_C, IP_F, IP_G) of the trace in the remainder of processing. In data sets collected with Cheleby [41], 772 K of 22.4 M path traces (i.e., 3.45%) contain routing loops among which 143 K has multiple loops. Moreover, we observed border firewalls that filter ICMP packets from/to a network domain and occasionally respond with their IP address. However, the hop distance of these IP addresses are not consistent. Hence, we filter any IP address that appears at the end of a trace after three unresponsive nodes. After this parsing/filtering, the algorithm Algorithm 1 in Fig. 7 provides a graph search module for parallel *-subpaths. In the algorithm, we extract all *-subpaths $u^*_{(a,b,l,\{*_1,*_2,\ldots,*_l\})}$ from the path traces and identify the same length *-subpaths with the same known end nodes (i.e., a and b) to merge unknown nodes with each other. The algorithm also builds the initial data structures that we will utilize in the graph indexing to resolve other structures. Note that we construct the conflict sets at this step.

While reading the traces, each $u^*_{(a,b,l,N)}$ is stored based on the known end nodes a and b in a hash table. Subsequently read $u^*_{(c,d,k,P)}$'s are then compared to the ones with the same hash value in the data structure. This results in a complexity of $O(|U|.log(|U|))$ where $|U|$ is the number of *-subpaths.

Another related pattern is caused by routers that apply ICMP rate limiting or that stay unresponsive when congested, i.e., temporary unresponsiveness. Such a router may appear as a known node in some path traces and may appear as a '*' in others. For instance, in Fig. 8a, an ICMP rate limiting router c may cause occurrences of related subpaths in the form of $(\ldots, a, c, b, \ldots)$ and $(\ldots, a, *_1, b, \ldots)$ in different traces, and we resolve $*_1$ to c. However, in some cases, there might be a subpath $(\ldots, a, d, b, \ldots)$ as well. In such cases, we resolve $*_1$ to either c or d only if just one of them is marked as temporarily unresponsive. Additionally, as in Fig. 8b, ICMP rate limiting routers c and d may cause occurrences of related subpaths in the form of $(\ldots, a, *_1, *_2, b, \ldots)$, $(\ldots, a, c, *_3, b, \ldots)$ and $(\ldots, a, *_4, d, b, \ldots)$ in different traceroute outputs. In this case, we resolve $*_1$ and $*_4$ to c; and $*_2$ and $*_3$ to d.

Let $G = (V, E)$; $V \leftarrow \emptyset$; $E \leftarrow \emptyset$; $U \leftarrow \emptyset$; $maxLength \leftarrow 0$;
for (each $trace$ in $\bigcup trace(v_i, v_j)$)
  $V \leftarrow V \cup \{a, b\}$; $E \leftarrow E \cup \{e_{(a,b)}\}$ $\forall u_{(a,b,0,\emptyset)} \in trace$
  for (each $u^*_{(a,b,l,\{*_1,*_2,\ldots,*_l\})} \in trace$)
    if $(\neg\exists\ u^*_{(a,b,l,\_)} \in U)$
      $V \leftarrow V \cup \{a, *_1, *_2, \ldots, *_l, b\}$
      $U \leftarrow U \cup u^*_{(a,b,l,\{*_1,*_2,\ldots,*_l\})}$
      $E \leftarrow E \cup \{e_{(a,*_1)}, e_{(*_l,b)}\} \cup \{e_{(*_i,*_{i+1})} \forall i, 1 \le i < l\}$
      if $maxLength < l$
        $maxLength \leftarrow l$

Fig. 7. Algorithm 1: Finding Parallel *-Subpaths.



(a) Resolution of length 1 *-subpath

(b) Resolution of longer *-subpaths

Fig. 8. Resolution of temporarily unresponsive routers.

The algorithm Algorithm 2 in Fig. 9 resolves the temporarily unresponsive routers. In the algorithm, for each *-subpath $u^*_{(a,b,l,\{*_1,*_2,\ldots,*_l\})}$, we look for the same length paths between the same end nodes a and b. Finding paths for given end nodes in a naive manner might result in a high time complexity. While building the initial graph in algorithm Algorithm 1 in Fig. 7, we also index the neighbors of each node in our node structure.

In order to find the paths between given end nodes, we find $\lceil l/2 \rceil$-hop neighbor set of the left end node and $\lfloor (l+1)/2 \rfloor$-hop neighbor set of the right end node. If the intersection set of these two sets is not an empty set, we identify the path between the end nodes. Moreover, if there are multiple paths, we can process the unknown nodes with the highest ranked path, which can be determined based on how many times it appears, and repeat this with other paths based on their rank order as long as there are remaining unknown nodes. Note that, as earlier decisions affect the later ones, an error would be propagated. Hence, ranking of nodes based on the number of appearance in traces allows us to process more frequently observed nodes.

The algorithm takes $O(|U| \cdot (a_{nd})^{l/2})$ time where $|U|$ is the number of *-subpaths and a_{nd} is the average node degree of all nodes. Note that we limit the length of *-subpaths to be processed in this step by 5 as longer *-subpaths increase the time complexity and the number of such subpaths are relatively small according to data sets in Table 2. Thus, worst case complexity becomes $O(|U| \cdot (a_{nd})^3)$.

4.3.2. Star structures

After building *-subpath database from path traces, we build an index of the star structures (e.g., Fig. 6g) that will be utilized in resolving unresponsive routers as in Fig. 6h and finding bipartite and triangle substructures. The star structure typically appears in path traces collected from a single vantage point toward a number of destinations or from multiple vantage points toward the same destination. The observed topology looks like the one presented in Fig. 6g. We identify this type of structures by clustering unresponsive neighbors (e.g., $*_v$) of nodes (e.g., a, the head node of *-subpaths in Fig. 6g).

We index maximal star structures for each node $v_i \in V$ using Algorithm 3 in Fig. 10. We define a node b as an

INPUT: $G = (V, E)$ and U from Alg.1
  for (each $u^*_{(a,b,l,\{*_1,*_2,\ldots,*_l\})} \in U$) where $l <= 5$
    if ($\exists\ u_{(a,b,l,\{v_1,v_2,\ldots,v_l\})}$))
      for (each $u_{(a,b,l,\{v_1,v_2,\ldots,v_l\})} \in$ rankByFreq($\bigcup\ u_{(a,b,l,\{v_1,v_2,\ldots,v_l\})} \in U$))
        if (mergeable($*_1 \leftarrow v_1$) & ... & mergeable($*_l \leftarrow v_l$))
          set $*_1 \leftarrow v_1$, $*_2 \leftarrow v_2$, ..., $*_l \leftarrow v_l$ in G
          $U \leftarrow U - u^*_{(a,b,l,A)}$

Fig. 9. Algorithm 2: Resolving Temporarily Unresponsive Routers.

INPUT: $G = (V, E)$ and U from Alg.2;
$maxLength$ from Alg.1; $S \leftarrow \emptyset$
for (each *node* $v \in V$)
  for (i=1:maxLength)
    $S \leftarrow S \cup s_{(v,i,\emptyset,\emptyset)}$
for (each $u^*_{(a,b,l,A)} \in U$)
  $s_{(a,l,N,M)} \leftarrow s_{(a,l,(N\cup\{b\}),(M\cup A))}$
  $s_{(b,l,N,M)} \leftarrow s_{(b,l,(N\cup\{a\}),(M\cup A))}$
for (each $s_{(v,l,N,M)} \in S$)
  if $|N| < 2$
    $S \leftarrow S - s_{(v,l,N,M)}$

Fig. 10. Algorithm 3: Star Structure Indexing.

l-neighbor of a, if there is a trace with l unknown nodes in between them. For example, a and b are l-neighbor of one another, if there is a path trace $(a, *_1, *_2, \ldots, *_l, b)$ with exactly l unresponsive routers from a to b. Star structures within a graph $G = (V, E)$ are represented as $s_{(v_i,l,N_i,M_i)}$ where v_i is the pivot node, l is the number of unknown nodes between the v_i and each of its l-neighbors, N_i is the set of all l-neighbors of v_i, and M_i is the set of all unknown nodes in this star structure. The algorithm builds on the *-subpaths database U and produces star structure database S.

The algorithm first builds a star structure $s_{(v_i,0,\emptyset,\emptyset)}$ for each known node $v_i \in V$. Then, for each *-subpath $u^*_{(a,b,l,A)}$, it appends node b to the neighbor set of the star structure of node a with length l, and vice versa. It also appends the set A to the M set of both nodes' star structures. After processing all *-subpaths, the algorithm removes star structures $s_{(v,l,N,M)}$ that have less than two neighbors, i.e. $|N| < 2$.

The overall run time complexity of the algorithm is $O(|V| + |U|)$.

4.3.3. Complete bipartite structures

After building Star structure database S, we index complete bipartite structures K. A complete bipartite subgraph (e.g, a 2x3 complete bipartite in Fig. 6e) is formed among the known neighbors of an unresponsive router (i.e, $*_1$ for Fig. 6f). This type of structure occurs when path traces are between two sets of the unresponsive nodes' neighbors (i.e., $\{a, b\}$ and $\{c, d, e\}$ for Fig. 6e). In this case, the topology

INPUT: $G = (V, E)$ from Alg.2; S from Alg.3
$K \leftarrow \emptyset$
for (each $s_{(a,l,N,M)} \in S$)
  $L_{can} \leftarrow \emptyset$
  for (each $b_i \in N$)
    $L_{can} \leftarrow L_{can} \cup N^*$ where ($\exists\ s_{(b_i,l,N^*,M^*)} \in S$)
  $L_{can} \leftarrow L_{can} - \{a\}$
  $R_{can} \leftarrow N$
  for (each $v_i \in L_{can}$)
    $R_{new} \leftarrow R_{can} \cap N^+_i$ where ($\exists\ s_{(v_i,l,N^+,M^+)} \in S$)
    if ($|R_{new}| \geq 2$)
      $L_{new} \leftarrow \{a\} \cup \{v_i\}$
      for (each $v_j \in L_{can}$)
        if ($R_{new} \subset N^{\#}_j$) where ($\exists\ s_{(v_j,l,N^{\#},M^{\#})} \in S$)
          $L_{new} \leftarrow L_{new} \cup \{v_j\}$
          $M_{new} \leftarrow M_{new} \cup M^{\#}$
      $K \leftarrow K \cup k_{(L_{new},R_{new},l,M_{new})}$

Fig. 11. Algorithm 4: Complete Bipartite Structure Indexing.

database includes *-subpaths $(n_i, *_v, n_j)$ among all known neighbors of $*_v$. In general, this structure frequently occurs in paths collected using (k, m)-traces (i.e., tracing from a relatively small number of k vantage points to a larger number of m destinations) that is very common in topology mapping studies.

We index all complete bipartite structures using Algorithm 4 in Fig. 11. Note that, finding a complete bipartite subgraph $K_{m,n}$ with the maximal number of edges $m \cdot n$ is an NP-complete problem [67]. Our search is a special form where there are pre-indexed nodes that we pivot around in the search process. Note that as Internet graph has low degree, our approach can perform in a reasonable amount of time as shown in Section 5.2. We represent a complete bipartite graph as $k_{(V_1,V_2,l,M)}$ where V_1 and V_2 are two disjoint sets of neighboring nodes, l is the number of unknown nodes between the nodes v_i and v_j for any two nodes $v_i \in V_1$ and $v_j \in V_2$, and M_i is the set of all unknown nodes in this complete bipartite structure. The algorithm builds on the star database S from Algorithm 3 in Fig. 10.

In the algorithm, for each star structure $s_{(a,l,N,M)}$, we identify the maximal complete bipartite involving the node a. For this purpose, we first identify two candidate sets of nodes which will constitute the left and right hand side of the bipartite structure involving the node a. R_{can} set

represents the candidates for the right side of the complete bipartite and is also the neighbor set N of this star structure. L_{can} set indicates candidates for the left side of the complete bipartite. This set consists of all l-neighbors of each node in N.

We first find a $k_{(V_1^*,V_2,N^*,M^*)}$ where $|V_1^*| = 2$ and then grow it to $k_{(V_1,V_2,N,M)}$ where $|V_1| \geqslant 2$. Finding $k_{(V_1^*,V_2,N^*,M^*)}$, we iterate over each candidate node in the L_{can} as a pivot node and determine its neighbor intersection with that of the node a. If the intersection set is larger than two, these nodes belong to the right hand side. After determining the nodes in the R_{can} set, we grow the left hand side (i.e., L_{can}) and hence the $k_{(V_1^*,V_2,N^*,M^*)}$ structure by finding all nodes that has the right hand side nodes (i.e., R_{new}) as a neighbor.

Overall, finding complete bipartite graphs takes $O(|S| \cdot a_N^2)$ where $|S|$ is the number of star structures in the graph, and a_N is the average neighbor set size of all stars.

4.3.4. Triangle structure

Finally, we build the indexes of the triangle structures. A triangle is formed between an unresponsive router $*_v$ and its known neighbors $\{n_1, n_2, n_3\}$ when we consider the known neighbors of $*_v$. This type of structure occurs when path traces exists between all three known neighbors of an unknown node, i.e., the topology database includes all *-subpaths $(n_i, *_v, n_j)$ where $i,\ j \in [1,3],\ i \neq j$. Fig. 6b presents an example of this case where the data set includes *-subpaths among all known neighbors of the unknown node as shown in Fig. 6a. Note that larger cliques may appear in collected path traces however in our earlier study we observed their occurrence frequency to be very low [18]. As search for larger cliques is more costly and they had very small occurrence rate in our previous study, in this study, we limit our search for cliques to 3-cliques, i.e., triangles.

We index all triangles in the graph using Algorithm 5 in Fig. 12. The algorithm iterates over the star structures reported by Algorithm 4 in Fig. 10. Since we are interested in triangles, we consider only the star structures having $l = 1$. In the algorithm, for each star structure $s_{(a,1,N,M)}$, and $s_{(N_i,1,N^*,M^*)}$ where $N_i \in N$, we obtain the intersection set I of the N and N^* sets. For each $I_i \in I,\ t_{(\{a,N_i,I_i\},1,M)}$ constitutes a triangle where $a,\ N_i$, and I_i are the three known nodes of the triangle, and M is the set of all unknown nodes in this triangle structure.

INPUT: $G = (V, E)$ from Alg.2; S from Alg.3
$T \leftarrow \emptyset$; $I \leftarrow \emptyset$
for (each $s_{(a,1,N,M)} \in S$)
  for (each $N_i \in N$) where $N_i > a$
    if ($\exists\ s_{(N_i,1,P,M^i)}$)
      $I \leftarrow N \cap P$
      for (each $I_i \in I$) where $I_i > N_i$
        $T \leftarrow T \cup t_{(\{a,N_i,I_i\},1,\{A^1,A^2,A^3\})}$
        where $u^*_{(a,N_i,1,A^1)}$, $u^*_{(a,I_i,1,A^2)}$, $u^*_{(N_i,I_i,1,A^3)}$

Fig. 12. Algorithm 5: Triangle Structure Indexing.

Indexing all triangles takes $O(|S| \cdot a_N \cdot log(a_N))$ where $|S|$ is the number of star structures, and a_N is the average neighbor set size of all stars.

4.4. Unresponsive router resolution

After building graph indexes for star, complete bipartite and triangle structures, we resolve the corresponding unresponsive routers. During the resolution process, we first handle the triangle structures, then complete bipartite structures, and finally the star structures as there is a higher possibility of conflicts within star or complete bipartite structures. That is, observing a triangle structure versus a star structure provides more evidence for a single unresponsive router.

4.4.1. Triangle resolution

We resolve the triangle structures in the graph using the Algorithm 6 in Fig. 13. If trace preservation condition is satisfied, we combine unresponsive nodes A of a triangle structure $t_{(a,b,c,l,A)}$ into a single unresponsive router in the constructed map. Moreover, before the resolution, we sort triangles based on the total number of neighbors and process triangles from the one with the smallest number of neighbors to the largest. This ordering leaves cliquish structures that are due to multiple neighboring unresponsive routers to be processed later. In those scenarios, (as observed in our preliminary experiments) we might have multiple triangles some of which conflict the trace preservation condition.

Resolving all triangles takes $O(|T| \cdot log(a_C)$ where $|T|$ is the number of triangle structures and a_C is the average conflict set size. As there are three nodes in a triangle that need to be compared their mergeability can be determined using a sorted conflict list for each node.

4.4.2. Complete bipartite resolution

Next, we resolve the complete bipartite structures in the graph using the Algorithm 7 in Fig. 14. We combine unresponsive nodes A of a complete bipartite structure $k_{(V_1,V_2,l,A)}$ into a single or a chain of nodes in the constructed map under the trace preservation condition. If the unresponsive nodes in A violate the trace preservation rule, we find the maximal mergeable subset A^* of these unresponsive routers. A set of unknown nodes A are mergeable, i.e., *mergeable*(A), if they (or any other node in their alias sets) do not appear in the same path trace. If there is any, we combine unresponsive nodes in this maximal mergeable subset. Similar to triangle structures, we first sort complete bipartite structures based on their sizes,

INPUT: $G = (V, E)$ from Alg.2; T from Alg.5
sort (T)
for (each $t_{(\{a,b,c\},l,A)} \in T$)
  if (***mergeable***(A))
    merge$(*_e \in A)$

Fig. 13. Algorithm 6: Resolving Triangle Substructures.

INPUT: $G = (V, E)$ from Alg.6; K from Alg.4
sort (K)
for (each $k_{(ls,rs,l,A)} \in K$)
  if (***mergeable***(A))
    merge$(*_e \in A)$
  else
    $A^* \leftarrow$ ***findMergeable***(A)
    merge$(*_e \in A^*)$

Fig. 14. Algorithm 7: Resolving Complete Bipartite Substructures.

i.e. $|V_1| * |V_2|$. We then start processing from the smallest complete bipartite structure to the largest as they have smaller probability of having conflicts.

Resolving all complete bipartite structures takes $O(|K| \cdot a_K \cdot log(a_C))$ where $|K|$ is the number of complete bipartite structures, a_K is the average size of the complete bipartites, and a_C is the average conflict set size. As mergeability search begins with higher ranked nodes, which can be determined based on the number of occurrance in traces, we can find mergeable subsets. That is, as we are searching for the mergeabilty we can leave unmergeable nodes out of the set.

4.4.3. Star resolution

Finally, we resolve the star structures in the graph using the Algorithm 8 in Fig. 15. We combine all unknown nodes A of a star structure $s(a, l, N, A)$, i.e. all unresponsive neighbors of a node a, into a single node in the topology map under the trace preservation condition. If the unresponsive nodes in A do not satisfy the trace preservation condition, we find the maximal mergeable subset A^* of these unresponsive routers. If there is any, we combine the unresponsive nodes in this maximal mergeable subset. If there are multiple of them, we follow the same strategy that we used in Algorithm 2 in Fig. 15. We process the unknown nodes with the highest ranked path, which can be determined based on how many times it appears, and repeat this with other paths based on their rank order as long as there are remaining unknown nodes. Note that, as earlier decisions effect the later ones, an error would be propagated. Hence, ranking of nodes based on the number of appearance in traces allows us to process more frequently observed nodes. In the process, we first sort star structures

INPUT: $G = (V, E)$ from Alg.7; S from Alg.3
sort (S)
for (each $s_{(r,l,A)} \in S$)
  if (***mergeable***(A))
    merge$(*_e \in A)$
  else
    $A^* \leftarrow$ ***findMergeable***(A)
    merge$(*_e \in A^*)$

Fig. 15. Algorithm 8: Resolving Star Substructures.

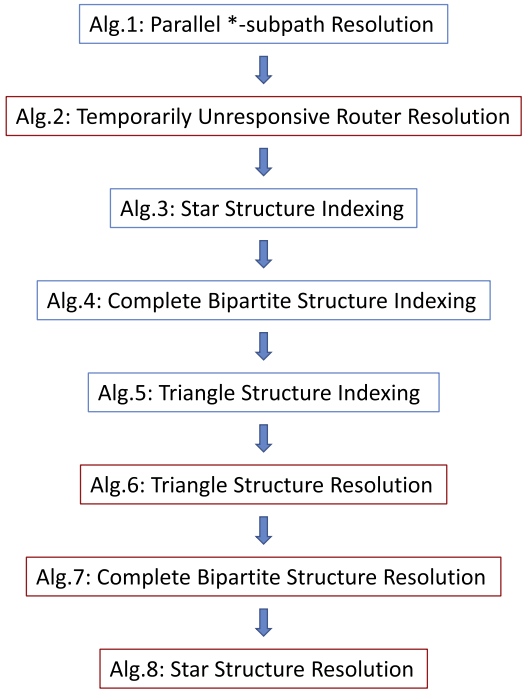


Fig. 16. Overall resolution process.

based on the total number of unresponsive routers they have. We then start processing from the star structure with the smallest number of unresponsive routers to the largest. This way, non-conflicting sets of unresponsive nodes will be processed before the ones that cannot be merged into a single node.

Resolving all star structures takes $O(|S| \cdot a_S \cdot log(a_C))$ where $|S|$ is the number of complete bipartite structures, a_S is the average size of the star structures, and a_C is the average conflict set size. Similar to complete bipartites, we can find mergeable subsets as we are searching for the mergeabilty.

4.5. Overall

Fig. 16 presents the overall flow chart of the unresponsive router resolution.

5. Evaluations

In this section, we use simulations and sampled topology data to evaluate accuracy and performance of *Graph Based Induction* (GBI) approach to resolve unresponsive routers.

5.1. Simulation-based evaluations

In our simulations, we use both synthetic and sampled topologies to compare the accuracy of our approach with that of *Initial Pruning* (IP) [34] and *Neighbor Matching* (NM) [37]. IP is a commonly used technique and corresponds to the Algorithm 1 in GBI. NM is similar to the star

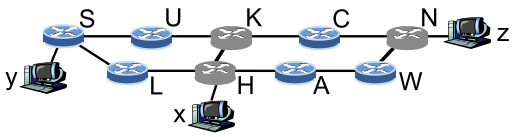
resolution step in GBI. Note that we did not compare GBI with graph minimization [36], dimensionality reduction [37] and spectral embedding [38] approaches due to their high complexities (see Section 5.2 for details).

For our comparisons, we use two network topologies:

- *Sample AMP network* obtained from AMP measurement infrastructure [68]. This topology consists of path traces among 130 vantage points and includes 2376 routers and 3770 links after unresponsive router resolution using GBI and extensive IP alias resolution [14,69].
- *Synthetic Transit-Stub (T-S) network* generated by GT-ITM topology generator [70]. This network consists of 50,000 nodes and 138,500 links.

We utilize the above topologies as the underlying actual networks. We randomly select a number of nodes in the networks as unresponsive (between 2% and 14%) and collect a number of (k, m) path traces to reflect traceroute-collected path traces. We use (10, 500) and (10, 1000) path traces from the sampled AMP network and (10, 1000), (10, 2000), and (10, 3000) path traces from the synthetic TS network. Note that in this study, we only consider *permanently unresponsive* routers as IP and NM approaches are completely ineffective for *temporarily unresponsive* routers. As indicated in Related Work Section, previous approaches focus on permanently unresponsive routers and ignore temporarily unresponsive ones.

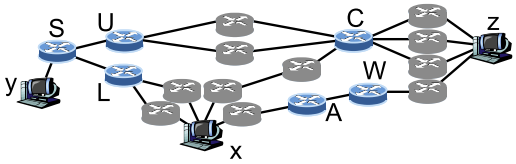
For the simulations, we randomly select a number of nodes in the actual networks as unresponsive, i.e., we choose the unresponsive routers using independent Bernoulli trials. Let's assume that we have the sampled topology in Fig. 17a where routers H, K, and N are unresponsive with two of them being neighbors. If we collect



(a) Sampled Network

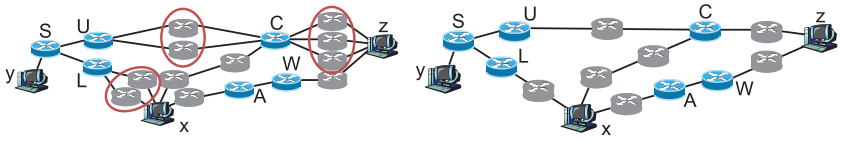
- x - ✱ - L - S - y
- x - ✱ - A - W - ✱ - z
- y - S - L - ✱ - x
- y - S - U - ✱ - C - ✱ - z
- z- ✱ - C - ✱ - ✱ - x
- z - ✱ - C - ✱ - U - S - y

(b) Collected Traces

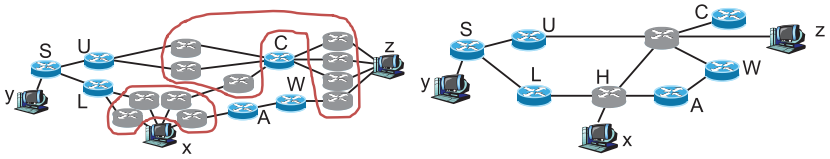


(c) Resulting Topology (traces between all vantage points)

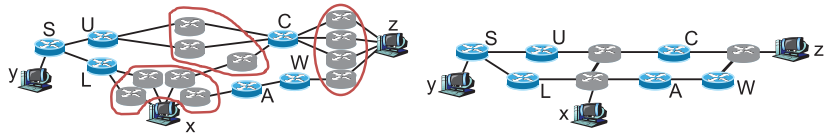
Fig. 17. Sample network for *-substrings.



(a) Initial Pruning (IP)



(b) Neighbor Matching (NM)



(c) Graph Based Induction (GBI)

Fig. 18. Resolution of analyzed approaches.

traces between vantage points, we will obtain the traces as in Fig. 17b, and the resulting observed topology will be as in Fig. 17c. Fig. 18 illustrates how IP, NM, and GBI approaches resolves the unresponsive routers in this topology.

In practice, there are four cases while merging unknown nodes into their corresponding unresponsive routers:

- *Perfect merge* is when all unknown nodes of an unresponsive router are correctly grouped into an alias set without false positives or false negatives.
- *Over merge* is when all unknown nodes of an unresponsive router are correctly grouped into an alias set but the set incorrectly includes additional unknown nodes. Hence, there are false positives.
- *Under merge* is when all unknown nodes corresponding to the unresponsive router are not grouped into the same alias set. Hence, there are false negatives.
- *Mixed merge* is when some (i.e., not all) unknown nodes of an unresponsive router are grouped into an alias set with other nodes from other unresponsive routers. In this case, we have both false positives and false negatives.

In general, merge operation can add *false positives* or lack of identifying merges can leave *false negatives*. As seen in the results presented below, IP has few false positives whereas NM has few false negatives. Our approach balances between both to obtain optimum results. Overall, Algorithm 1 and Algorithm 2 can be considered aggressive in their merge operations whereas others can be considered more cautious.

The following metrics are used for our evaluations. *Edit distance* and *unresponsive router ratio* assess the accuracy while *topology size* and *clustering coefficient* provide insight into topological characteristic of obtained networks.

Edit distance is the number of primitive operations required to transform the sampled graph to the actual graph. The primitive operations we define are *node split* and *node merge*. During *node split*, a node v is split into two nodes and a new vertex v' is added to the set of vertices. This operation fixes false positives. During *node merge*, two separate nodes v and v' are merged into a single node v and v' is removed from the set of vertices. This operation fixes false negatives. Note that this metric is similar to the widely used graph distance which accounts for edges as well. Edit distance, however, fits better in our context as it explicitly counts both false positives and false negatives.

We analyze edit distance of all samples and observe that on average edit distance of NM and IP are 39% and 379% higher than that of GBI, respectively. Fig. 19 shows the edit distance for (10, 500) AMP and (10, 2000) T-S samples. In some cases, NM algorithm does not improve over the Initial Pruning noticeably. As an example, when 14% of the nodes in the (10, 2000) T-S sample are unresponsive, edit distance reduces from 1800 with IP to 1784 with NM. Overall, in all samples, GBI has the smallest edit distance, i.e., GBI has least number of errors. Finally, Table 3 presents the average edit distances for all samples with Initial graph with no resolution. According to the results, GBI-based topologies have smallest number of errors in all cases.

Unresponsive router ratio is the ratio of the number of unresponsive routers in the induced topology to the ones in the actual topology. It helps observe the inflation caused by the unresponsive routers and assesses the effectiveness of the resolution. An unresponsive router ratio of one does not necessarily mean that unresponsive routers are correctly resolved without considering the edit distance.

We analyze unresponsive router ratio to assess the effectiveness of each approach. On average, unresponsive router ratios are 53.7, 3.72, 1.75, and 1.56 for Initial network, IP, NM, and GBI approaches, respectively. Fig. 20

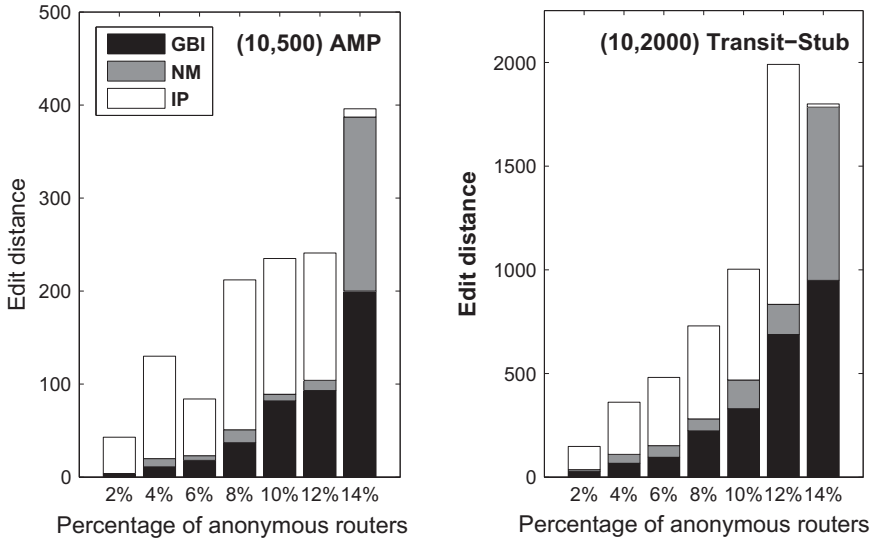


Fig. 19. Edit distances for two samples.

Table 3
Average edit distances.

Unresponsive (%)	2%	4%	6%	8%	10%	12%	14%
Initial	3798	4576	8093	10,519	11,045	16,383	19,079
IP	135	229	501	666	718	967	1,252
NM	32	58	189	272	319	502	1,190
GBI	23	37	146	215	274	430	633

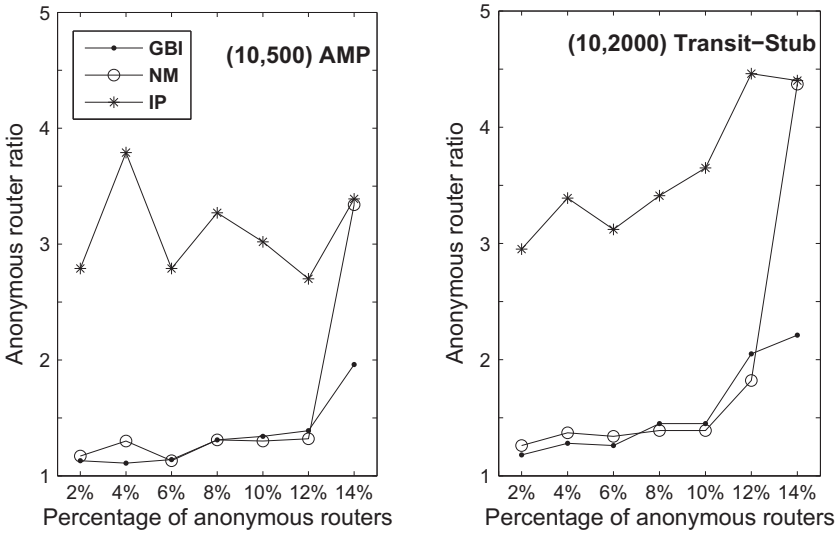


Fig. 20. Unresponsive router ratio for two samples.

Table 4
Average unresponsive router ratios.

Unresponsive (%)	2%	4%	6%	8%	10%	12%	14%
Initial	59.8	48.6	57.4	52.0	45.9	53.9	58.2
IP	3.4	3.3	3.9	3.9	3.9	3.7	4.2
NM	1.3	1.3	1.4	1.5	1.4	1.6	3.7
GBI	1.2	1.2	1.5	1.6	1.6	1.7	2.1

Table 5
Changes in graph characteristics.

	Initial (%)	IP (%)	NM (%)	GBI (%)
Number of nodes	+262	+21	+5	+4
Number of edges	+593	+83	+55	+49
Avg. node degree	+91	+51	+48	+43
Clustering coefficient	−98	−40	−19	−17
Assortativity	+117	+34	+13	+23

shows the unresponsive router ratio for (10, 500) AMP and (10, 2000) T-S samples. Similarly, Table 4 presents the average unresponsive router ratios for all samples. In general, GBI-based topologies have smallest ratios while, in some cases, NM results in smaller unresponsive router ratios. However, close examination of the results indicates that NM has higher edit distance in each of these cases as compared to GBI. The achieved reduction in graph size is basically due to higher number of false positives.

Topology size, in terms of the number of nodes and links, reveals the basic information about a network. Due to unresponsive routers, a traceroute-based topology map may include artificial nodes and artificial links. The first two rows in Table 5 compare approaches based on the average percentage increase in the number of nodes and links in the resulting topologies. On average, GBI based topologies have the smallest number of artificial nodes and artificial links.

Clustering coefficient is the ratio of the number of triangles to the number of triplets in a graph. This metric characterizes the connectivity density of a given graph. This metric is a useful metric to compare different resolution techniques as the resolution process modifies a given graph and changes its connectivity density. The fourth row in Table 5 compares the approaches based on the percentage difference of clustering coefficients as compared to that of the original graph. Most of the initial topologies have approximately zero clustering coefficient. On average, the clustering coefficient of the GBI based graph is closest to that of the original graph suggesting that the resulting graph resembles the original graph the most in terms of its connectivity density.

Assortativity coefficient measures the tendency of a network to connect nodes of the same or different degrees [71]. Positive values indicate assortativity, i.e., most of the links are between similar degree nodes and negative

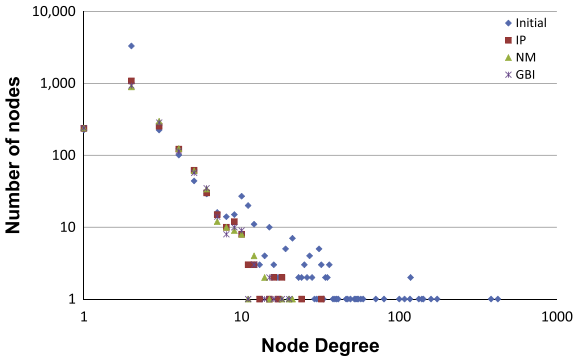


Fig. 21. Degree distribution.

values indicate disassortativity, i.e., most of the links are between dissimilar degree nodes. A value of 0 implies non-assortativity.

In addition to the above metrics, we studied **node degree distribution**. Fig. 21 presents the degree distribution of the resulting topologies. As shown in the figure, initial topology is considerably different from any of the resolved graphs.

Impact of Unresponsive Regions: In order to assess the effect of unresponsive regions as reported in Table 2, we generated synthetic topologies where 14% of routers were unresponsive while they followed the *-subpath distribution of 2013 Ark dataset. Below are the steps to generate the synthetic topologies for this experiment:

1. Compute number of unresponsive routers that we assume to exist in the topology. (i.e., $N * 0.14$).
2. Compute number of unresponsive chains that we assume to exist in the topology. (In Ark 2013 data, 37% of the *s are in length 1 *-subpaths, 26% of the *s are in length 2 *-subpaths, 21% of the *s are in length 3 *-subpaths, 16% of the *s are in length 4 *-subpaths. So, we need to place $N * 0.14 * 0.37 / 1$ length 1 *-subpaths, $N * 0.14 * 0.26 / 2$ length 2 *-subpaths, $N * 0.14 * 0.21 / 3$ length 3 *-subpaths, $N * 0.14 * 0.16 / 4$ length 4 *-subpaths, respectively)
3. Next, we place these unresponsive chains to the topology by randomly picking a responsive node (i.e., any node which has not been marked as unresponsive earlier) from the topology each time, and insert the chain by making this node as the starting point of the chain if the following conditions are satisfied:
   (a) Any neighbor of this chain should not also have been marked as unresponsive earlier (as this might cause longer chains).
   (b) According to the size (lets assume l) of the chain, if l-successive nodes of that node have not been marked as unresponsive.
4. We repeat step 3 until all chains are injected.[1]

[1] Note that, we implemented this approach just to generate our experimental synthetic topology. However, this approach might not always be able to inject all chains.

Table 6
14% unresponsive router regions for (10, 2000) T–S sample.

	Avg. unres. router ratio (%)	Avg. edit distance
Initial	60.1	23,122
IP	6.2	1726
NM	3.8	1403
GBI	1.8	718

Table 6 presents the results for (10, 2000) T-S sample which shows that GBI performs much better than the corresponding simulations in Tables 3 and 4. This indicates that GBI is much better in data sets that contains unresponsive regions as observed in real traces.

Impact of Alias IP Resolution: After collecting accurate path traces two of the crucial processes in the topology construction is the IP alias resolution and unresponsive router resolution [72,23].[2] Each of these tasks might have inaccuracies, i.e. false positives and/or false negatives. In order to analyze the impact of IP alias resolution on unresponsive router resolution, we measure edit distance values with and without IP alias resolution. As an example on the (10, 2000) T-S sample, we used both an ideal IP alias resolution where all IP aliases are resolved and partial resolution with APARv2 [14,41] that provided 65–70% resolution on average. We also varied the unresponsive router ratio in the topology. Table 7 presents the edit distance values for initial and final topologies. Applying IP alias resolution before unresponsive router resolution improves the edit distance values for all approaches. Even though the improvement in edit distance is higher for IP and NM approaches than GBI approach, GBI produces the best edit distance results for all cases.

In **summary**, in this section, we have compared GBI with practical approaches IP and NM, and have shown that GBI performs better in terms of edit distance or unresponsive router ratio. We also have shown that GBI based graphs are closer to the original graphs in terms of topological characteristics. In addition, GBI handles all types of unresponsiveness whereas IP and NM are completely ineffective for *temporary unresponsiveness* (which are not considered in the above simulations) because they only consider permanent unresponsiveness.

5.2. Experimental results on sampled topologies

In this section, we use sampled topology data to analyze the feasibility of GBI. We use three data sets collected in February, 2011:

(i) iPlane: from 234 vantage points to ~144 K destinations [32];
(ii) Ark: from 51 vantage points to ~9.5 M destinations [31]; and
(iii) Cheleby: from 400 vantage points to ~3.5 M destinations [41].

[2] Note that load balancing and tunneling are common practices in the Internet. However, topology collection study should focus on obtaining accurate path traces and paths with inaccuracies should be corrected/filtered prior to the topology construction.

Table 7
Impact of IP alias resolution.

	No. resolution		Partial resolution		Perfect resolution	
	2%	14%	2%	14%	2%	14%
Initial	4212	17,726	3712	15,127	3426	13,419
IP	158	1597	146	1346	140	1293
NM	68	1372	33	1213	30	1096
GBI	30	732	24	658	21	645

Table 8
Graph Based Induction technique on real data sets.

	iPlane	Ark	Cheleby
#Traces	32 M	22.7 M	15 M
#IPs	300 K	1.2 M	1.2 M
Initial #*s	15,182,604	7,862,649	7,207,885
#*s resolved by Algorithm 1	14,320,510	7,213,793	6,137,750
#*s resolved by Algorithm 2	212,460	122,820	251,279
#*s resolved by Algorithm 6	3212	2688	2858
#*s resolved by Algorithm 7	158,862	115,194	143,880
#*s resolved by Algorithm 8	406,101	244,076	419,204
Total resolved	15,101,944	7,698,571	6,954,971
Final #*s	80,660	164,078	252,914

After filtering inaccurate and incomplete path traces, we resolve IP aliases using APARv2 [14,41]. Table 8 presents the results for each step of GBI to resolve unresponsive routers in the data set. Algorithm 1 applies the Initial Pruning to considerably reduce the number of unknown nodes. Algorithm 2 then identifies unknown nodes due to temporarily unresponsive routers (i.e., the ones due to ICMP rate limiting or due to congestion at the router). This step resolves over 27% of the existing unknown nodes. Note that none of the previous approaches handle this type of unresponsiveness.

We then index star, complete bipartite and triangle structures using Algorithms 3, 4, and 5, respectively. Next, Algorithm 6 addresses the unknown nodes in the triangle structures. Note that due to the (k,m) nature of the traces that span from a few sources to a large number of destinations, the number of observed clique structures and the corresponding resolutions is small [18]. In the following step, Algorithm 7 resolves the unknown nodes in the complete bipartite structures. Finally, Algorithm 8 processes the unknown nodes in star structures.

Overall, GBI reduces the number of unknown nodes by 99%, 98% and 97% for iPlane, Ark and Cheleby data sets, respectively (and by 91%, 75% and 76% if we consider the topology after Initial Pruning as the starting point). Note that no resolution process will reduce the number of unknown nodes by 100% when there are permanently unresponsive routers. Moreover, we observe that resolution is better with denser graphs such as iPlane that focuses traces on a region.

Additionally, in order to assess the impact of unresponsive routers on the large scale topological characteristics of the resulting graphs, we analyze the node degree distributions of the known nodes with and without unresponsive router resolution. Fig. 22 presents the node degrees of

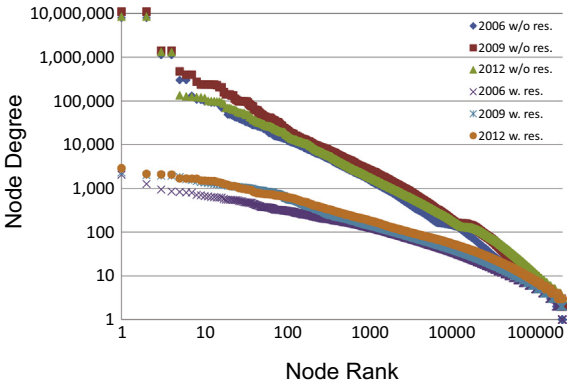


Fig. 22. Effect on degree distribution.

Table 9
Complexity and operational overhead of GBI.

	Time complexity	Number of operations		
		iPlane	Ark	Cheleby
Algorithm 1	$O(\|U\| \cdot log(\|U\|))$	$5.0 * 10^6$	$2.2 * 10^6$	$6.7 * 10^6$
Algorithm 2	$O(\|U\| \cdot (a_{nd})^3)$	$\mathbf{4.0} * 10^8$	$1.8 * 10^7$	$\mathbf{3.3} * 10^7$
Algorithm 3	$O(\|V\| + \|U\|)$	$0.9 * 10^6$	$1.2 * 10^6$	$1.2 * 10^6$
Algorithm 4	$O(\|S\| \cdot (a_N)^2)$	$11 * 10^6$	$\mathbf{4.1} * 10^6$	$5.6 * 10^6$
Algorithm 5	$O(\|S\| \cdot a_N \cdot log(a_N))$	$1.3 * 10^6$	$0.5 * 10^6$	$0.7 * 10^6$

Table 10
Size of the data structures.

	iPlane	Ark	Cheleby
$\|U\|$	0.9 M	0.4 M	1.1 M
$\|V\|$	0.3 M	1.2 M	1.2 M
$\|S\|$	219 K	93 K	148 K
$\|K\|$	858 K	752 K	793 K
$\|T\|$	99 K	4.7 K	5.3 K
a_{nd}	7.67	3.55	3.11
a_N	7.07	6.62	6.11

the known nodes for 2006, 2009, and 2012 iPlane datasets that we utilized in Section 3.2 (other ones had similar trends as well). As shown in the figure, unresponsive routers significantly inflate the node degrees.

Finally, we examine the operational overhead of GBI in Table 9. We estimated the number of required operations by using the data structure sizes presented in Table 10. We use these values to compare the run time overhead of GBI with earlier approaches. Based on the run time of the GBI algorithms, the highest time complexity is due to the Algorithm 2, i.e., approximately $4.0 * 10^8$ operations for the iPlane data. Note that, even though the complexity seems to be $O(n \cdot m^3)$, the value of m is much smaller than n. Moreover, Table 10 presents the space requirements of each index structure that is used in the resolution process.

The NM approach has a time complexity of $O(n^2)$ where n is the total number of nodes in the data set after the Initial Pruning. For iPlane data, n is approximately 1.15 M and hence NM would take 10^{12} steps. Similarly, the

dimensionality reduction approach of [37] would take 10^{18} operations while the graph minimization approach of [36] would take 10^{30} operations. Since Almog et al. [38] have not provided the time complexity of their approach, we are not able to compare the computational overhead of their approach. However, they utilize a distance matrix to resolve the unknown nodes and the straightforward resolution of a large scale graph is not practical with this approach. Authors partition the map into subgraphs and handle each separately.

6. Conclusion

In this paper, in order to assess the extent of unresponsive routers in Internet topology mapping studies, we first analyze the nature of unresponsive routers and identify different types of unresponsiveness. To this end, we present an experimental study on the responsiveness of routers to the active probing. In our historical analysis, we observe that in general responsiveness reduced during the last decade and regions of unresponsive routers expanded. We also observe that network operators are increasingly using rate limiting of active probes.

In the second part of the paper, we develop a *Graph Based Induction (GBI)* approach to resolve unresponsive routers. In this approach, based on our novel *Structural Graph Indexing*, we index observed subgraphs that contain unknown nodes. Then, we determine the corresponding minimal underlying structure that satisfies the trace accuracy condition. Our work improves the state of the art in unresponsive router resolution in terms of both accuracy and efficiency. Regarding accuracy, GBI addresses all cases of unresponsive routers whereas the previous approaches ignore temporary unresponsiveness. Regarding efficiency, the run time complexity of our algorithm is significantly less than that of existing algorithms. Our experiments on three different large scale data sets have shown a significant reduction in the practical run time overhead of our approach (approximately, $4.0 * 10^8$ operations) as compared to the previous approaches (approximately, 10^{12} to 10^{30} operations in the worst case). Overall, GBI can be utilized for Internet topology mapping studies aimed to obtain sample network graphs. Most of such measurement studies ignore unresponsive routers due to the high complexity in processing unknown nodes. GBI presents an efficient solution that makes a balance between false positives and false negatives while resolving unresponsive routers.

Acknowledgment

This work was supported in part by National Science Foundation Award #CNS-1321164.

References

[1] M. Newman, Networks: An Introduction, Oxford University Press, Inc., New York, NY, USA, 2010.
[2] M.B. Akgun, M.H. Gunes, Link-level network topology generation, in: Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops, ICDCSW '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 140–145.
[3] P. Gill, M. Arlitt, Z. Li, A. Mahanti, The flattening Internet topology: natural evolution, unsightly barnacles or contrived collapse?, in: PAM'08: Proceedings of the 9th International Conference on Passive and Active Network Measurement, Springer-Verlag, Berlin, Heidelberg, 2008, pp 1–10.
[4] H. Karaoglu, M. Yuksel, M. Gunes, On the scalability of path exploration using opportunistic path-vector routing, in: 2011 IEEE International Conference on Communications (ICC), 2011, pp. 1 –5.
[5] M. Gunes, S. Bilir, K. Sarac, T. Korkmaz, A measurement study on overhead distribution of value-added Internet services, Comput. Networks 51 (14) (2007) 4153–4173.
[6] H. Haddadi, M. Rio, G. Iannaccone, A. Moore, R. Mortier, Network topologies: inference, modeling, and generation, Commun. Surv. Tutorials, IEEE 10 (2) (2008) 48–69.
[7] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, A. Joglekar, An Integrated Experimental Environment for Distributed Systems and Networks, Boston, MA, 2002.
[8] PlanetLab Project. <http://www.planet-lab.org>.
[9] L. Cheng, N. Hutchinson, M. Ito, Realnet: a topology generator based on real Internet topology, in: Advanced Information Networking and Applications (AINAW), 2008, pp. 526–532.
[10] E. Arslan, M. Yuksel, M.H. Gunes, Network management game, in: 2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN), 2011, pp. 1–6.
[11] V. Jacobson, Traceroute, Lawrence Berkeley Laboratory (LBL). <ftp://ee.lbl.gov/traceroute.tar.gz> (February 1989).
[12] F. Viger, B. Augustin, X. Cuvellier, C. Magnien, M. Latapy, T. Friedman, R. Teixeira, Detection, understanding, and prevention of traceroute measurement artifacts, Comput. Netw. 52 (5) (2008) 998–1018.
[13] B. Donnet, M. Luckie, P. Mérindol, J.-J. Pansiot, Revealing MPLS tunnels obscured from traceroute, SIGCOMM Comput. Commun. Rev. 42 (2) (2012) 87–93.
[14] M. Gunes, K. Sarac, Resolving IP aliases in building traceroute-based Internet maps, IEEE/ACM Trans. Network. 17 (6) (2009) 1738–1751.
[15] M.E. Tozal, K. Sarac, Palmtree: an IP alias resolution algorithm with linear probing complexity, Comput. Commun. 34 (5) (2011) 658–669 (special Issue: Complex Networks).
[16] M. Gunes, K. Sarac, Inferring subnets in router-level topology collection studies, in: ACM SIGCOMM Internet Measurement Conference (IMC), San Diego, CA, 2007.
[17] M.E. Tozal, K. Sarac, Tracenet: an internet topology data collector, in: Proceedings of the 10th Annual Conference on Internet Measurement, IMC '10, ACM, New York, NY, USA, 2010, pp. 356–368.
[18] M. Gunes, K. Sarac, Resolving anonymous routers in Internet topology measurement studies, in: Proceedings of IEEE INFOCOM, Phoenix, AZ, USA, 2008.
[19] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, R. Teixeira, Avoiding traceroute anomalies with Paris traceroute, in: Proceedings of IMC, Rio de Janeiro, Brazil, 2006.
[20] R. Sherwood, A. Bender, N. Spring, DisCarte: A disjunctive Internet cartographer, in: Proceedings of the ACM SIGCOMM, Seattle, WA, USA, 2008.
[21] N. Spring, R. Mahajan, D. Wetherall, T. Anderson, Measuring ISP topologies using rocketfuel, IEEE/ACM Trans. Network. 12 (1) (2004) 2–16.
[22] M.B. Akgun, M.H. Gunes, Bipartite internet topology at the subnet-level, in: IEEE International Workshop on Network Science (NSW 2013), IEEE, West Point, NT, 2013 (IEEE).
[23] M. Gunes, K. Sarac, Importance of IP alias resolution in sampling Internet topologies, in: IEEE Global Internet (GI), 2007.
[24] R. Teixeira, K. Marzullo, S. Savage, G. Voelker, In search of path diversity in ISP networks, in: Proceedings of the USENIX/ACM Internet Measurement Conference, Miami, FL, USA, 2003.
[25] M.B. Akgun, M.H. Gunes, Impact of multi-access links on the internet topology modeling, in: IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2013), IEEE, San Francisco, CA, 2013 (IEEE).
[26] H.B. Acharya, M.G. Gouda, A theory of network tracing, in: SSS '09: Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 62–74.
[27] G. Antichi, A. Di Pietro, D. Ficara, S. Giordano, G. Procissi, F. Vitucci, Network topology discovery through self-constrained decisions, in: Global Telecommunications Conference, 2009, GLOBECOM 2009, IEEE, 2009, pp. 1–6.

[28] J. Ni, H. Xie, S. Tatikonda, Y. Yang, Efficient and dynamic routing topology inference from end-to-end measurements, IEEE/ACM Trans. Network. 18 (1) (2010) 123–135.
[29] M.H. Gunes, K. Sarac, Analyzing router responsiveness to active measurement probes, in: Proceedings of the 10th International Conference on Passive and Active Network Measurement, PAM '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 23–32.
[30] D. McRobb, K. Claffy, T. Monk, Skitter: CAIDA's Macroscopic Internet Topology Discovery and Tracking Tool, 1999. <http://www.caida.org/tools/skitter/>.
[31] Archipelago Measurement Infrastructure (Ark). <http://www.caida.org/projects/ark>.
[32] H.V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, A. Venkataramani, iPlane: an information plane for distributed services, in: OSDI, 2006.
[33] H. Kardes, M.H. Gunes, Structural graph indexing for mining complex networks, in: Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems Workshops, ICDCSW '10, Washington, DC, USA, 2010, pp. 99–104.
[34] S. Bilir, K. Sarac, T. Korkmaz, Intersection characteristics of end-to-end Internet paths and trees, in: IEEE International Conference on Network Protocols (ICNP), Boston, MA, USA, 2005.
[35] D. Feldman, Y. Shavitt, An optimal median calculation algorithm for estimating Internet link delays from active measurements, in: IEEE E2EMON, Munich, Germany, 2007.
[36] B. Yao, R. Viswanathan, F. Chang, D. Waddington, Topology inference in the presence of anonymous routers, in: IEEE INFOCOM, San Francisco, CA, USA, 2003.
[37] X. Jin, W.-P.K. Yiu, S.-H.G. Chan, Y. Wang, Network topology inference based on end-to-end measurements, IEEE J. Sel. Areas Commun. 24 (12) (2006) 2182–2195 (Special Issue on Sampling the Internet).
[38] A. Almog, J. Goldberger, Y. Shavitt, Unifying unknown nodes in the Internet graph using semisupervised spectral clustering, in: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops, IEEE Computer Society, Washington, DC, USA, 2008, pp. 174–183.
[39] Cheleby: An Internet Topology Mapping System. <http://cheleby.cse.unr.edu/>.
[40] Y. Shavitt, E. Shir, DIMES: let the Internet measure itself, ACM SIGCOMM CCR 35 (5) (2005) 71–74.
[41] H. Kardes, M.H. Gunes, T. Oz, Cheleby: a subnet-level internet topology mapping system, in: COMSNETS, 2012, pp. 1–10.
[42] S. Triukose, Z. Wen, A. Derewecki, M. Rabinovich, Dipzoom: An open ecosystem for network measurements, in: Proceedings of IEEE INFOCOM, Anchorage, AK, USA, 2007.
[43] E. Blanton, M.E. Tozal, K. Saraç, S. Fahmy, Location matters: eliciting responses to direct probes, in: IEEE 32nd International Performance Computing and Communications Conference, IPCCC 2013, San Diego, CA, USA, December 6–8, 2013, 2013, pp. 1–10.
[44] N. Spring, D. Wetherall, T. Anderson, Scriptroute: A public internet measurement facility, in: Proceedings of USENIX Symposium on Internet Technologies and Systems, 2003.
[45] M. Luckie, Y. Hyun, B. Huffaker, Traceroute Probe Method and Forward IP Path Inference, in: Internet Measurement Conference (IMC), Vouliagmeni, Greece, 2008, pp. 311–324.
[46] A. Broido, K. Claffy, Internet topology: connectivity of IP graphs, in: Proceedings of SPIE ITCom Conference, Denver, CO, USA, 2001.
[47] B. Cheswick, H. Burch, S. Branigan, Mapping and visualizing the Internet, in: ACM USENIX, San Diego, CA, USA, 2000.
[48] J. Du, Y. Li, A solution for anonymous routers discovery based on source-routing traceroute, in: Proceedings of 2013 International Conference on Advances in Materials Science and Manufacturing Technology, Trans Tech Publications Ltd., Stafa-Zuerich, Switzerland, 2013, pp. 1050–1054.
[49] Y.-A. Pignolet, S. Schmid, G. Tredan, Misleading stars: what cannot be measured in the internet?, Distrib Comput. 26 (4) (2013) 209–222.
[50] H. Kardes, D. Konidena, S. Agrawal, M. Huff, A. Sun, Graph-based approaches for organization entity resolution in mapreduce, in: Proceedings of TextGraphs-8 Graph-Based Methods for Natural Language Processing, in Conjunction with EMNLP 2013, 2013, pp. 70.
[51] N. McNeill, H. Kardes, A. Borthwick, Dynamic record blocking: efficient linking of massive databases in mapreduce, in: Proceedings of the 10th International Workshop on Quality in Databases (QDB), in Conjunction with VLDB 2012, 2012.
[52] H. Kardes, S. Agrawal, X. Wang, A. Sun, Ccf: fast and scalable connected component computation in mapreduce, in: Proceedings of the International Conference on Computing, Networking and Communications (ICNC), 2014, IEEE, 2014, pp. 994–998.
[53] H. Kardes, A. Sevincer, M.H. Gunes, M. Yuksel, Six degrees of separation among us researchers, in: Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), IEEE Computer Society, 2012, pp. 654–659.
[54] D. Cook, L. Holder, Mining Graph Data, John Wiley & Sons, 2006.
[55] T. Matsuda, H. Motoda, T. Washio, Graph-based induction and its applications, Adv. Eng. Inform. 16 (2) (2002) 135–1434.
[56] D. Shasha, J.T.L. Wang, R. Giugno, Algorithmics and applications of tree and graph searching, in: Symposium on Principles of Database Systems, 2002, pp. 39–52.
[57] C.A. James, D. Weininger, J. Delany, Daylight Theory Manual – Daylight 4.91, April 2005.
[58] X. Yan, P.S. Yu, J. Han, Graph indexing: a frequent structure-based approach, in: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD '04, ACM, New York, NY, USA, 2004, pp. 335–346.
[59] iPlane. <http://iplane.cs.washington.edu/>.
[60] CYMRU. <http://www.team-cymru.org/services/ip-to-asn.html>.
[61] M. Antic, A. Smiljanic, Routing with load balancing: increasing the guaranteed node traffics, IEEE Commun. Lett. 13 (6) (2009) 450–452, http://dx.doi.org/10.1109/LCOMM.2009.081874.
[62] Route Selection in Cisco Routers, Tech. Rep. Document ID: 8651, Cisco. <http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094823.shtml> (January 2008).
[63] Configuring a Load-Balancing Scheme, Tech. rep., Cisco. <http://www.cisco.com/en/US/docs/ios-xml/ios/ipswitch_cef/configuration/15-0m/isw-cef-load-balancing.pdf> (October 2011).
[64] R. Sherwood, N. Spring, Touring the Internet in a TCP sidecar, in: Proceedings of the ACM/SIGCOMM on Internet Measurement Conference, ACM Press, New York, NY, USA, 2006, pp. 339–344.
[65] R. Govindan, H. Tangmunarunkit, Heuristics for Internet map discovery, in: IEEE INFOCOM, Tel Aviv, ISRAEL, 2000.
[66] K. Keys, Y. Hyun, M. Luckie, K. Claffy, Internet-Scale IPv4 Alias Resolution with Midar: System Architecture, Tech. rep., Cooperative Association for Internet Data Analysis – CAIDA, May 2011.
[67] R. Peeters, The maximum edge biclique problem is NP-complete, Discr. Appl. Math. 131 (3) (2003) 651–654.
[68] NLANR, Multicast Beacon, National Laboratory for Applied Network Research. <http://dast.nlanr.net/Projects/Beacon/> (June 2000).
[69] M. Gunes, K. Sarac, Analytical IP alias resolution, in: IEEE International Conference on Communications (ICC), Istanbul, Turkey, 2006.
[70] E.W. Zegura, K.L. Calvert, M.J. Donahoo, A quantitative comparison of graph-based models for Internet topology, IEEE/ACM Trans. Network. 5 (6) (1997) 770–783.
[71] M.E.J. Newman, Assortative mixing in networks, Phys. Rev. Lett. 89 (20) (2002) 208701.
[72] M. Gunes, K. Sarac, Impact of alias resolution on traceroute-based sample network topologies, in: Passive and Active Measurement Conference (PAM), Louvain-la-neuve, Belgium, 2007.

Hakan Kardes is a researcher at Inome Inc. He received his B.S degree in Computer Engineering from Bogazici University, TURKEY in 2009. He got his M.S and PhD degrees in Computer Science from University of Nevada, Reno in 2010 and 20012, respectively. His main research interests are complex networks, graph data mining and Internet measurements.

Mehmet Hadi Gunes is an Associate Professor at University of Nevada, Reno. He received B.S. degrees in Computer Science & Engineering and Electronics Engineering from Isik University of Turkey in 2002; M.S. in Computer Science & Engineering from Southern Methodist University in 2004; and Ph.D. in Computer Science from University of Texas at Dallas in 2008. His research interests include Communications: protocols, health systems, smart grid communications; Complex networks: biological networks, social networks, graph data mining, network visualization; Internet measurements: Internet topology, Internet modeling; Network security: anonymizers, private communication, secure cloud. His research is funded by National Institute of Justice and National Science Foundation.

Kamil Sarac received the M.S. and Ph.D. degrees in computer science from the University of California at Santa Barbara, in 1997 and 2002, respectively. He is currently an Associate Professor in the Department of Computer Science at the University of Texas, Dallas. His research interests include Computer networks; network and service monitoring and Internet measurements; overlay networks; network security and denial-of-service defense; broadcast in ad hoc networks; group communication and IP multicast.