Efficient IP-level network topology capture

Thomas Bourgeau, Timur Friedman

Efficient IP-level network topology capture

Thomas Bourgeau and Timur Friedman

LIP6-CNRS and LINCS laboratories, UPMC Sorbonne Universités

Abstract. Large-scale distributed traceroute-based measurement systems are used to obtain the topology of the internet at the IP-level and can be used to monitor and understand the behavior of the network. However, existing approaches to measuring the public IPv4 network space often require several days to obtain a full graph, which is too slow to capture much of the network's dynamics. This paper presents a new network topology capture algorithm, NTC, which aims to better capture network dynamics through accelerated probing, reducing the probing load while maintaining good coverage. There are two novel aspects to our approach: it focuses on obtaining the network graph rather than a full set of individual traces, and it uses past probing results in a new, adaptive, way to guide future probing. We study the performance of our algorithm on real traces and demonstrate outstanding improved performance compared to existing work.

1 Introduction

This paper proposes a fundamental improvement to distributed IP-level internet topology measurement systems. Such systems have focused on conducting full end-to-end route traces and as a result take a considerable time to obtain a graph of the network. Here, we propose an approach focused directly on obtaining the graph, which is, as a result, much faster (presuming the principal goal is to obtain this graph, rather than full routes). The main benefit of our approach is that we can get a much better view of the dynamics of the network at the IP-level by reducing the probing load generated and by accelerating the measurement accordingly. While the internet network topology will expand dramatically with the wide adoption of IPv6, it is crucial to design highly scalable and fast measurement systems as we sketch in our proposed algorithm. Furthermore, capturing fine-grained network topology dynamics at the IP-level would enhance monitoring solution with faster detection of topological changes [1] and would deepen our current knowledge of the evolving internet topology with new realistic models and generators [2].

These systems are characterized by a variety of parameters, such as the number of measurement sources, the number of destinations, and the time scale which constitute a full discovery round. Implicit in these parameters is a network discovery budget that represents the number of probes sent to discover the resulting graph.

Our challenge is to reduce the discovery budget per source, so as to enable more rapid probing rounds. In so doing, we face certain constraints. Lakhina et al. demonstrated how measuring from too few sources could introduce biases in the discovered graph [3]. Shavitt et al. have gone further to demonstrate how a broad distribution of sources and destinations yields good estimates of graph properties [4]. We cannot simply adopt an approach of capturing small and restricted network topology, as does RIPE's TTM system [5], if our aim is to obtain accurate graphs of the internet as a whole. On the other hand, as Bourgeau has described [6], large systems [7–9] risk missing important aspects of network dynamism as they require long probing rounds on the order of days.

Our approach consists in reducing the probing redundancy involved in discovering the same topological elements by exploiting previous measurement information. We also adapt our probing strategy to track dynamism events along the paths where topological changes have been diagnosed. Our approach differs from existing network topology discovery methods, in that we take advantage of previously detected topological features to guide the sampling of the network topology and we look at the impact of capturing network dynamism with partial traceroutes sampling.

This paper describes our network topology capture algorithm (NTC) and evaluates it based upon distributed traceroute measurements collected on PlanetLab [10]. Emulating NTC on this dataset, we find that NTC consumes just 6% of the probing budget of a classic system conducting end-to-end traceroutes. In so doing, it still covers 95% of the network topology. This outperforms the state of the art Doubletree approach, which (on another, similar, dataset), required 25% of a classic probing budget and discovers 93% of the network topology. This probing budget reduction translates directly into more efficient capture of network topology dynamism.

2 A Generic Distributed Tracing (GDT) framework

Distributed network tracing systems tend to be similar to each other. Each has a number of lightweight agents and a heavier weight central server. Probing is conducted in rounds, with each agent working from a fixed set of instructions for a round. Results from the agents are sent back from time to time to the central server. We formalize these notions into a Generic Distributed Tracing (GDT) framework. The framework leaves room for many different specific probing heuristics to be applied. The following section describes related work in the context of this framework, and the section after that describes our own NTC (network topology capture) heuristics.

The actors in the generic framework are a **server** and a set of **agents**. Tracing is conducted in a series of **rounds**, with three **phases** to each round: **dispatch**, in which instructions are sent from the server to the agents; **probing** by the agents; and **update**, in which the probing results are sent back to the server, which uses them to prepare the next round. Let us further detail each phase.

Dispatch phase: The information classically provided to agents in a distributed tracing system is simply a list of destinations that each one should probe, using full route traces. Once this information has been provided to the agents for a first round, it tends not to change much in subsequent rounds. However, the two changes that we introduce to tracing methodology – conducting partial route traces and using previously detected features to guide probing – require agents to receive fuller information and round-by-round updates. For the partial traces, the server must communicate not only the destination, but also the hop counts for a trace. And for the previously detected features, the server must inform an agent about what to expect to see in each partial trace.

We formalize a partial trace instruction as a **query**, in the sense that the agent will 'query' the network regarding the existence of a single edge of the network graph. We must clarify what we mean by an edge because of the well-known phenomenon of unresponsive interfaces, commonly called 'stars', that often appear in route traces, as well as the less frequently seen non-public or otherwise illegal IP addresses. For our purposes, an **edge** consists of two legal IP addresses: v_1, seen at a hop count h in a route trace from source s (the agent) to destination d; and v_2, seen at hop count $h + \ell$, where ℓ is a positive integer. If $\ell > 1$, this means that there are intermediate hops consisting of stars and/or illegal IP addresses, which we exclude from our graph of the network topology. In order to try to revisit the edge $e = (v_1, v_2)$, the query $q = (s, d, h, \ell)$ instructs agent s to probe towards d, starting at h and ending at $h + \ell$.

We formalize the notion that the agent is launching query q explicitly to visit edge e as an **expected view** $c = (q, e)$. By knowing the expected view, the agent can autonomously undertake additional probing if e should not be present. Not all probing can be based on prior experience, however. Typically, an agent's first probing round will consist of full route traces towards a set of destinations. There might be a reason to introduce full traces in other rounds as well, for instance to promote additional exploration. So the full instructions that a server provides to an agent consist in a set C of expected views complemented with a set D of destinations for full traces.

Probing phase: Agents carry out their instructions in the probing phase, recording the results to send back to the server. Agents might take autonomous action beyond their direct instructions, conducting more or less probing in response to what they, and possibly other agents, are seeing in the current round. A result might simply be that an expected edge has been seen. If it has not been seen, or if additional probing was conducted, then the trace information (destination, hop count, interface seen, for each hop) must be communicated. If probing is less than instructed, then a reason might be communicated.

Update phase: In the update phase, the server collects the results from each agent and updates its database of expected views. If history extends back only one round, all new information overwrites the old. A more sophisticated approach stores information from all rounds, allowing the next dispatch phase to be based on the fullest record possible.

3 Related work

This paper situates itself in the context of the small body of work on improving the efficiency of distributed route tracing systems. The distributed work builds on earlier work on the efficiency of single-agent systems. The essential distinguishing feature of the distributed problem is that the work can be divided among agents. (See Donnet et al. [11] for single-agent references.) There are two prior approaches to the distributed problem: Donnet et al.'s Doubletree [11] and Gonen and Shavitt's work [12].

Seen within the GDT framework described above, **Doubletree** innovates in the probing phase. It divides the destination set into as many subsets as there are agents, and it divides the probing phase of each round into that many sub-rounds. During each sub-round, an agent works on its own unique subset of the destination set. When it passes that subset on to the next agent for the next sub-round, it also passes along information about the IP addresses that it has seen when probing towards each destination in the subset. Those address-destination pairs form a tracing "stop set", allowing the next agent to avoid redundant probing. With each sub-round, each agent adds its own information to the stop set. The stop sets are not kept beyond the end of the probing phase, and each round begins anew.

Again, as seen within the GDT framework, **Gonen and Shavitt** have innovated in the dispatch phase. Their server designates destination sets for each agent that are subsets of the full destination set. Based upon knowledge of the route traces from a prior round, these instructions are aimed at reducing probing redundancy as much as possible while still maintaining 100% coverage in the current round.

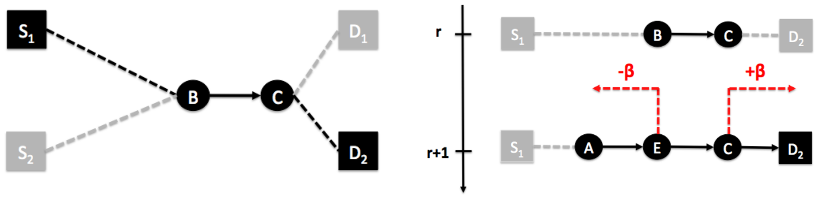
Both approaches function within the paradigm of the production route tracing systems, in which route traces are full end-to-end traces from each agent to every destination in a specified set. Doubletree allows partial traces to be conducted, but only on the condition that other information is available from which all full traces can be reconstituted (subject to some, hopefully small, error). Gonen and Shavitt dispense with the aim of being able to reconstitute traces from each agent to every destination in the set, focusing instead on obtaining the network graph topology that results from the complete set of traces. They allow a subset of the complete set of traces to be conducted.

Our NTC (Network Topology Capture) approach is the first to fully embrace the graph-based perspective. As with Gonen and Shavitt, we aim at obtaining the fullest possible graph, and are ready to dispense with some routing path knowledge in order to do so efficiently. We are also ready, however, to dispense with full route traces as the means to obtaining the graph, thereby opening up the possibilities for much greater efficiency.

Previous work has looked, as we also do, at the effect of more efficient distributed tracing on network graph coverage. However, ours' is the first work to look at the impact on the ability of such systems to effectively capture network topology dynamics.

4 Network Topology Capture (NTC) heuristics

Within the GDT framework described above, we employ two heuristics (see Fig. 1 and below) that, together, we call our Network Topology Capture (NTC) approach to distributed tracing.



(a) Redundancy aware probing (b) Dynamism aware probing

Fig. 1. Network Topology Capture (NTC) heuristics

Redundancy aware probing: We know from the Doubletree work [11] that a considerable amount of probing redundancy is due to a small proportion of discovered edges (80% of the probes sent discover just 10% of the edges in their case). Our redundancy aware probing heuristic looks at prior rounds' probing results and counts the number of different queries capable of seeing each edge. These include both multiple queries from a single agent to various destinations ("intra-monitor redundancy" in Doubletree terms) and queries from multiple agents (which goes beyond Doubletree's "inter-monitor redundancy" because there is no constraint that the traces must be towards the same destination). The heuristic intervenes at the dispatch phase by globally capping the number of queries per edge, across all agents, in a round at a value α. These expected views are chosen at random.

In Fig. 1(a), prior probing has show that four queries, two from S_1 and S_2 towards D_1 and D_2, yield the edge (B, C). With $\alpha = 1$, redundancy aware probing dispatches only a single expected view for (B, C), tracing from S_1 towards D_2 at the appropriate hop counts. In practice, because network dynamics might cause queries to fail, we might explicitly allow introduce edge redundancy by using an α value greater than 1.

Dynamism aware probing: When a query fails to yield the expected edge, this is a sign that routing has changed. An agent could content itself with reporting back just the interfaces that it has seen, but to do so would be to forgo the possibility of discovering more information surrounding the change. Our dynamism aware probing heuristic intervenes at the probing phase, in which the agent continues probing forwards and backwards from the expected view until it has discovered a number β of legitimate IP addresses in both directions (or until tracing terminates for the normal reasons of reaching source or destination or a maximum hop count).

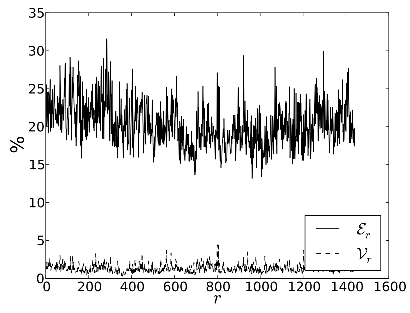
Fig. 1(b) shows that an expected view from round r of (B, C), when tracing from S_1 towards D_2, fails in round $r+1$, yielding (E, C) instead. Based on $\beta = 1$, agent S_1 continues probing backwards until it discovers one additional legitimate IP address, A. If also continues probing forwards, but just rediscovers D_2. Note that A has not been seen before, and we do not know what it connects to. The higher the value of β, the more chances we have to connect newly-found vertices and edges to the known topology.

5 Performance evaluation

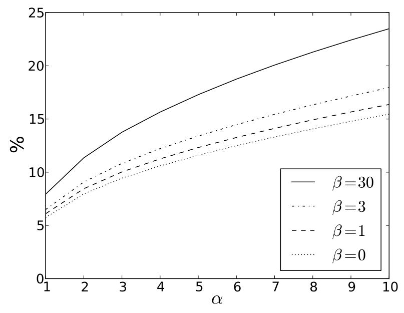
This section evaluates how well our NTC (network topology capture) heuristics do at covering the graph of the network in each probing round and how well they do at capturing the graph dynamics between probing rounds. There is a trade-off between the discovery budget, on the one hand, and the degrees of coverage and captured dynamics on the other. We explore this trade-off through the two tunable parameters that we have introduced: α, governing how many different ways we try to reprobe each edge, and β, governing how far we search for previously-seen IP addresses when we encounter unexpected IP addresses in our reprobing. Higher α and higher β both mean a greater discovery budget, and, as we see below, both bring gains of different sorts for coverage and capture. The maximum values ($\alpha = 10$ and $\beta = 30$) correspond to a probing budget of roughly 25% of a full trace probing budget, which is the budget reported for the state of the art Doubletree algorithm [11].

Our evaluation is based upon a real dataset that we have captured, with full traces from every source to every destination, on which we simulate how discovery would have proceeded if we had been conducting selected partial traces based on the NTC heuristics. Existing datasets [7–9] were not suitable to our purposes for a couple of reasons. First, their time granularity is coarser than we would wish for a study of network dynamics. An individual probing round taking on the order of days for Ark [7], DIMES [8] and one day for iPlane [9]. Second, we were concerned that traces that did not employ Paris Traceroute [13] would introduce false dynamics due to the interaction between per-flow load-balancing routers and the way in which classic Traceroute modifies the flow identifier for each probe packet that it sends. Not all distributed network probing systems have switched over to Paris Traceroute (Only Ark has deployed Paris Traceroute). We collected our measurements[1] over the course of two months, from 25 May to 25 July 2010, using the TDMI measurement infrastructure that is associated with the TopHat system [14]. We employed TDMI agents at over 230 PlanetLab nodes worldwide that we chose for their relative stability. Each agent performed one measurement round per hour, for a total of $R = 1480$ rounds. A round consisted of Paris Traceroutes towards 800 destinations, which are themselves PlanetLab

[1] Our dataset and algorithm description are available at `http://ntc.top-hat.info`.



(a) Portion of dynamism events observed



(b) Porportion of discovery budget used

Fig. 2. Network topology analysis and NTC discovery budget reduction

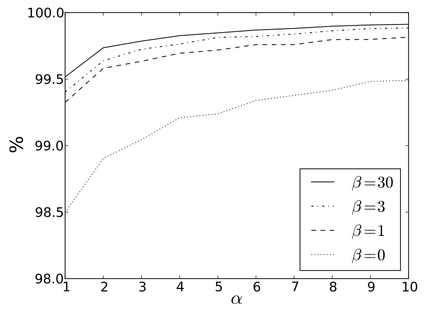
nodes. With Paris Traceroute, we traced a single path per source-destination pair, taking care to use the same flow identifier each time.

For each round r, we aggregate the discovered paths to build a directed graph $\mathcal{G}_r = (\mathcal{E}_r, \mathcal{V}_r)$ that we refer to as the **network topology**. Since there are typically unresponsive interfaces, or 'stars', in a route trace, and since non-public or otherwise illegal IP addresses can also appear, we define an edge $e \in \mathcal{E}_r$ to consist of two consecutive **legitimate interfaces** (public IP addresses), $v_1, v_2 \in \mathcal{V}_r$, separated by a number $\ell - 1$, possibly zero, of **unknown interfaces**: $e = (v_1, v_2, \ell)$. We term **network topology dynamism** to be the symmetric difference between two consecutive discovered graphs: $\mathcal{G}_r \mathbin{\Delta} \mathcal{G}_{r-1}$. The appearance or disappearance of a vertex or an edge between rounds is a **dynamism event**.

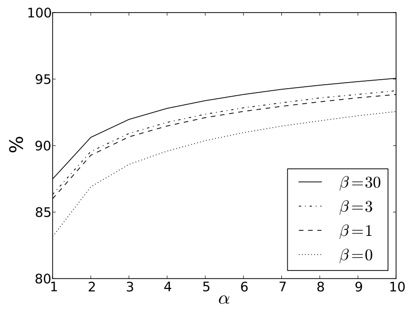
The graphs on average contained 13,950 vertices and 61,881 edges. Fig. 2(a) plots the rate of dynamism events per round. We see that **vertex dynamism**, $|\mathcal{V}_r \mathbin{\Delta} \mathcal{V}_{r-1}|/|\mathcal{V}_r|$, represents a small portion of a round 2% of all vertices, whereas **edge dynamism**, $|\mathcal{E}_r \mathbin{\Delta} \mathcal{E}_{r-1}|/|\mathcal{E}_r|$, represents on average 20% of the edges. We attribute the relatively high proportion of edge dynamism to the appearance and disappearance of unknown interfaces, a phenomenon already noted by Gunes and Sarac [15].

Discovery budget: The discovery budget is the number of probes that are sent per round. Fig. 2(b) shows the budget when using our NTC heuristics as a proportion of the budget consumed by conducting full traces, plotting the averages over all rounds. Depending upon the particular values of α and β that we choose, the budget is anywhere from 6% to 24% of the full trace budget. Since we still obtain excellent coverage (see below), this means that our NTC heuristics outperform the state of the art Doubletree, which in a similar scenario uses a probing budget of at best 25% of full trace budget [11].

Network topology coverage: The network topology coverage is the proportion of the graph that is discovered in a round, in comparison to the graph that is



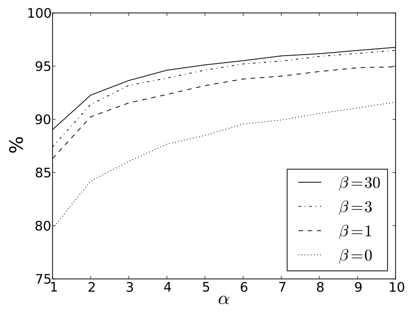
(a) Proportion of vertices covered



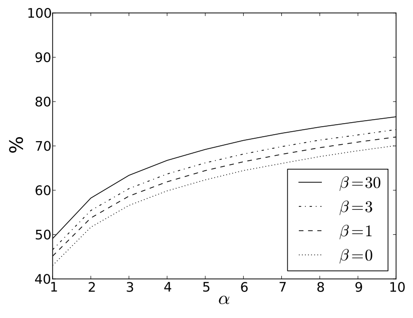
(b) Proportion of edges covered

Fig. 3. Proportion of vertices and edges covered when using NTC heuristics.

obtained from full traces. If $\mathcal{V}(\alpha, \beta)_r \subseteq \mathcal{V}_r$ is the set of vertices discovered using our NTC heuristics, with parameters α and β, in round r, the vertex coverage for that round is $|\mathcal{V}(\alpha, \beta)_r|/|\mathcal{V}_r|$. We plot the mean coverage over all rounds. Fig. 3(a) shows the vertex coverage and Fig. 3(b) shows the edge coverage, which is calculated similarly. We see that vertex coverage is between 98% and 99%, and that edge coverage varies between 82% and 95%, depending upon the parameter choices. For comparison, Doubletree, in similar circumstances, covers at most 93% of the edges that are seen in a full trace (and, as just noted, for a higher discovery budget).



(a) Proportion of vertex dynamics



(b) Proportion of edge dynamics

Fig. 4. Proportion of dynamic events captured when using NTC heuristics.

Dynamic event capture: As for budget and coverage metrics, we calculate dynamic event capture as a proportion, comparing the results when applying the

NTC heuristics to those of full traces. If d_r is the vertex dynamism, as defined above, for full traces, and $d(\alpha, \beta)_r$ is the vertex dynamism between the vertices $\mathcal{V}(\alpha, \beta)_r \subseteq \mathcal{V}_r$ found in round r, and the vertices $\mathcal{V}(\alpha, \beta)_{r-1} \subseteq \mathcal{V}_{r-1}$ found in round $r-1$, under NTC, then the vertex capture rate is $d(\alpha, \beta)_r/d_r$. Similarly for the edge capture rate.

Fig. 4(a) shows that the NTC heuristics capture over 80% of the vertex dynamics, and as much as 96% for the parameters that we studied. In Fig. 4(b), we see that the corresponding figures for edge dynamics are 44% and 75%. As we have already noted, we believe that a large part of edge dynamism results from changes in unknown interfaces, such as a 'star' appearing or disappearing in a route trace, and these dynamic events prove comparatively hard to capture.

6 Summary and future work

This paper has described a new network topology algorithm that demonstrates how large-scale measurement systems can be operated with a reduced discovery budget, so as to better capture network dynamics while at the same time maintaining broad coverage of the network topology. Compared to common measurement systems, such as our measurement dataset, our algorithm can save up to 94% of discovery budget while reaching 95% of edge coverage and capturing 75% of edge dynamics. To do so, our algorithm is based on a novel approach based on knowledge of previously learned graph elements, combined with adaptive probing in response to discovered dynamism. As our algorithm outperforms state of the art techniques, such as Doubletree, it opens the way to new large-scale and fast measurement systems deployment with the ability to efficiently capture network topology and its dynamics. Our future work will address an unknown interface filtering algorithm to enhance dynamism events capture and we will stress NTC to live deployment, scaling up to all IPv4 prefixes, in an attempt to provide to the research community the most precise network topology maps including exhaustive network topology dynamism information captured.

Acknowledgements

The research leading to these results has received funding from the EU FP7 OpenLab project (Grant No. 287581). The authors would like to thank Jordan Augé and Marc-Olivier Buob for there efforts and assistance in providing measurement data through the TopHat measurement system (`http://www.top-hat.info`).

References

1. E. Katz-Bassett, C. Scott, D. R. Choffnes, I. Cunha, V. Valancius, N. Feamster, H. V. Madhyastha, T. Anderson, A. Krishnamurthy, LIFEGUARD: practical repair of persistent route failures, in: Proc. ACM SIGCOMM, 2012.

2. B. Quoitin, V. Van den Schrieck, P. Francois, O. Bonaventure, IGen: Generation of router-level internet topologies through network design heuristics, in: Proc. ITC, 2009.
3. A. Lakhina, J. W. Byers, M. Crovella, P. Xie, Sampling biases in IP topology measurements, in: Proc. IEEE INFOCOM, 2003.
4. Y. Shavitt, U. Weinsberg, Quantifying the importance of vantage points distribution in internet topology measurements, in: Proc. IEEE INFOCOM, 2009.
5. M. Alves, L. Corsello, D. Karrenberg, C. Ogut, M. Santcroos, R. Sojka, H. Uijterwaal, R. Wilhelm, Providing active measurement as a regular service for ISP's, in: Proc. PAM, 2002.
6. T. Bourgeau, Monitoring network topology dynamism of large-scale traceroute-based measurements, in: Proc. CNSM, 2011.
7. K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, D. Krioukov, Internet mapping: from art to science, in: Proc. CATCH, 2009.
8. Y. Shavitt, E. Shir, DIMES: Let the internet measure itself, ACM SIGCOMM Computer Communication Review 35 (5) (2005) 71–74.
9. H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, A. Venkataramani, iPlane: An Information Plane for Distributed Services, in: Proc. Usenix OSDI, 2006.
10. B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, PlanetLab: an overlay testbed for broad-coverage services, ACM SIGCOMM Computer Communication Review 33 (3) (2003) 3–12.
11. B. Donnet, P. Raoult, T. Friedman, M. Crovella, Deployment of an algorithm for large-scale topology discovery, IEEE Journal on Selected Areas in Communications (JSAC) 24 (2006) 2210–2220.
12. M. Gonen, Y. Shavitt, A $\Theta(\log n)$-approximation for the set cover problem with set ownership, Information Processing Letters 109 (3) (2009) 183–186.
13. B. Augustin, T. Friedman, R. Teixeira, Measuring multipath routing in the internet, IEEE/ACM Transactions on Networking (TON) 19 (3) (2011) 830–840.
14. T. Bourgeau, J. Augé, T. Friedman, TopHat: supporting experiments through measurement infrastructure federation, in: Proc. TridentCom, 2010.
15. M. Gunes, K. Sarac, Analyzing router responsiveness to active measurement probes, in: Proc. PAM, 2009.