Practical Issues with Using Network Tomography for Fault Diagnosis

Yiyi Huang
Georgia Institute of Technology
yiyih@cc.gatech.edu

Nick Feamster
Georgia Institute of Technology
feamster@cc.gatech.edu

Renata Teixeira
CNRS and UPMC Paris Universitas (Paris 06)
renata.teixeira@lip6.fr

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.

ABSTRACT

This paper investigates the practical issues in applying network tomography to monitor failures. We outline an approach for selecting paths to monitor, detecting and confirming the existence of a failure, correlating multiple independent observations into a single failure event, and applying existing binary networking tomography algorithms to identify failures. We evaluate the ability of network tomography algorithms to correctly detect and identify failures in a controlled environment on the VINI testbed.

Categories and Subject Descriptors: C.2.3 [Network Operations]: Network monitoring C.2.3 [Network Operations]: Network management C.2.5 [Local and Wide-Area Network]: Internet

General Terms: Experimentation, Management, Measurement, Reliability

Keywords: Network tomography, fault detection

1. INTRODUCTION

Dynamic network conditions, such as traffic shifts, device failures, and router misconfigurations, can degrade the performance of end-to-end paths. To maintain high availability, operators must quickly detect, diagnose, and correct faults that cause these degradations. Network diagnosis involves both *detecting* that a network fault has occurred and *identifying* the location of the fault. Active monitoring services such as Keynote [11] and RIPE TTM [16] can detect some failures, but these systems do not identify the location of network faults, and they do not aim for fast detection. Due to the shortcomings of existing systems, operators have trouble detecting faults before their customers do. Even though network devices raise alarms and traps for certain types of failures, operators still typically resort to active measurement techniques to manually reason about the possible causes of performance problems (*e.g.*, running pings and traceroutes from various vantage points).

Various previous work has relied on "binary network tomography" [5, 6, 12] for determining the location of network faults. *Network tomography* correlates active measurements across multiple paths to determine the location of failures [4] by correlating probes from all monitors to all destinations at the time the failure was observed. Binary tomography algorithms assume that the network topology before the time of a failure is known and attributes the failure to the smallest set of links that explain all failed paths. Figure 1 presents an example scenario with two monitors, $m1$ and $m2$, and one destination, t. If both paths $p1$ and $p2$ fail, then the algorithm would infer that link $L2$ or $L3$ has failed; if only $p1$ fails, it would pinpoint the failure of $L1$ as the most likely cause. Unfortunately, most existing tomography studies make at least one assumption that is not valid in the general Internet setting, such as (1) the ability to send multicast probes [3], (2) control over all hosts that receive measurement probes [3, 5, 7, 12, 14], or (3) the lack of any network dynamics in the presence of failures [14].

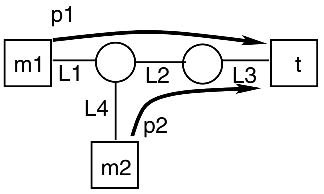

Figure 1: Topology with two probed paths $p1$ and $p2$.

This paper studies the effectiveness of network tomography on the VINI testbed. VINI is a controlled environment, where the assumptions above do not hold. We first devise a method for correlating active measurements into related observations for the same network failure event (Section 2). Using this method, we evaluate existing network tomography approaches by emulating the Abilene network topology on the VINI testbed and injecting controlled link failures into the testbed on each link in the topology (Section 3). We then study the extent to which an implementation of an existing network tomography algorithm, NetDiagnoser [5] correctly identifies the location of each failure (Section 4).

2. APPROACH

This section describes our approach for applying tomography algorithms on a live network. Our goal is to identify faults that occur within a *target network*. *Monitors* are hosts located inside or outside the target network that periodically send ping probes and traceroutes to *destinations*, which can be any host or router on the Internet. To cover as many interfaces in the target network as possible, the destinations should be widely distributed. We assume that a central *coordinator* collects measurements from all monitors as input to the network tomography algorithm.

We represent the target network as the set of interfaces within that network. Interfaces can be directly measured using traceroutes. This choice avoids unnecessary errors that may be introduced in alias resolution (*i.e.*, mapping interfaces to routers).

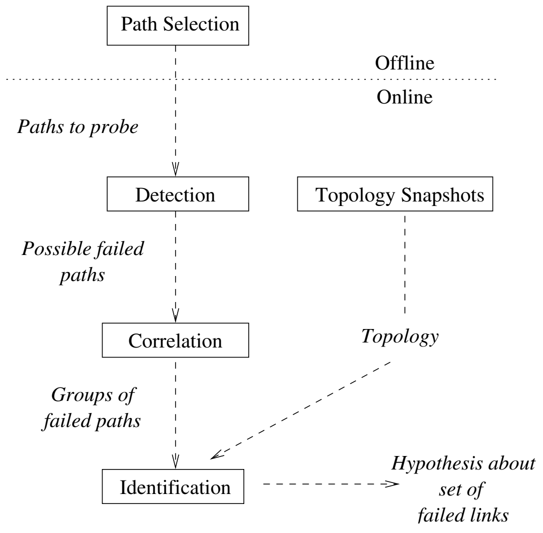

Figure 2: Applying network tomography in practice.

Figure 2 presents our approach for applying tomography on a live network, which comprises five operations:

Offline path selection. Based on a snapshot of the topology among all paths, the coordinator computes offline the set of paths that each monitor must probe to completely cover the target network for locating a failure. We use the path selection algorithm proposed by Nguyen and Thiran [13] using a combination of monitors and destinations for which some subpath crosses the target network.

Periodic topology snapshots. Monitors periodically collect traceroutes to all destinations and send path changes to a *coordinator* that maintains a "snapshot" of the network topology. This snapshot provides an input topology for the tomography algorithms that we apply in the identification phase described below. We select the last stable snapshot before the failure as input for the algorithm.

Detection and confirmation. To detect failures along paths, monitors periodically probe each destination in their set, determined from offline path selection. Each monitor issues both periodic traceroutes and active probes. When a monitor detects a lost probe along a path, it confirms the existence of a failure by sending three additional probes. After confirming a failure, it sends a report to the coordinator that some path (*i.e.*, collection of interfaces) has failed.

Correlation. Tomography takes as input a set of independently observed path failures that are presumed to be attributable to a single failure event. To produce this input, we must first *correlate* independent observations of failed paths from the monitors into collections of related failures.

We first group consecutive failure notifications along a single path. We then correlate independent path failures that occur close together in time. To do so, we sort failure reports based on the start time of each failure. Then, we traverse this list comparing every pair of consecutive paths to group multiple path failures into a "correlated event". If two paths p_1 and p_2 have failed at times t_1 and t_2, respectively we group them into the same event if $t_2 - t_1 \leq 10$ seconds. After the coordinator experiences a time interval longer than ten seconds with no new failure reports, it builds the reachability matrix using this set. In this paper, we perform this correlation offline; in practice, however, a central coordinator might be able to perform this correlation online.

Identification. When the monitors confirm the existence of a failure, the coordinator correlates failure notifications issued by all monitors and applies the NetDiagnoser binary network tomography algorithm [5] to identify the location of the failure. This process takes as input the topology before the failure and the set of failed paths and produces a set of possible locations of the network fault (*i.e.*, which links and nodes may have failed). In this paper, we perform identification offline.

3. EXPERIMENT SETUP

This section describes the experimental setup for evaluating the approach in Section 2 in a controlled environment on the VINI testbed [1]. VINI allows experimenters to perform realistic experiments and run real routing and network monitoring software in a distributed environment, while still retaining control over the time, duration, location, and nature of network failures. For our controlled experiments, we replicate the Abilene topology in the VINI testbed. We then send probes across the testbed that correspond to the same set of paths that the monitor selection algorithm selects for diagnosing failures on the actual Abilene testbed.

Topology. We configure VINI to emulate the Abilene network as closely as possible. Because Abilene points-of-presence (PoPs) no longer host PlanetLab nodes, we selected PlanetLab nodes that are both stable and geographically close the Abilene PoPs. We select nodes at universities connected to Abilene to ensure that our experiment only traverses Abilene and its access networks, *not* the commercial Internet. Each node in the VINI experiment runs the Quagga open-source software router [15] and forwards packets over tunnels that emulate the respective Abilene links. To ensure that paths through the emulated network closely mimic those in the actual Abilene topology, the VINI nodes run OSPF using the same OSPF adjacencies and link weights as the actual Abilene routers.

Paths to probe. To ensure that the controlled measurements on VINI reflect those that would be observed in an actual Internet scenario, we first collect path measurements from PlanetLab. We applied measurements from 242 PlanetLab nodes to 241 destinations yielding 58,322 paths across the real Abilene network. Then, we determine, for each path in the PlanetLab dataset, the Abilene routers where the path enters and exits the network (*i.e.*, the ingress and egress router). Then, we map each of these paths to the corresponding ingress and egress routers in Abilene, so that we can keep all measurements on VINI, where we have the ability to inject network faults. Ultimately, each router has a list of destinations that contains many duplicate entries for each egress, where each entry corresponds to some wide-area path that traverses that ingress-egress router pair.

We perform diagnosis for two different sets of paths: complete and reduced. *Complete* refers to running the detection phase using the complete set of paths from 242 PlanetLab node to 241 destinations (other PlanetLab nodes). *Reduced* refers to only probing the paths obtained from path selection. This set contains 15 paths. Reducing the number of paths that each monitor must probe also allows each monitor to probe each path more frequently; in other words, it reduces the *cycle time*. In the case of the complete measurements, the maximum cycle time is 3.83 seconds and the average is 3.45 seconds. In contrast, for the reduced set of paths, the maximum and average cycle times are 0.68 seconds and 0.45 seconds, respectively.

Failure emulation. We create link failures in VINI by installing a Click packet filter element in front of the interface of each end of the link for which we are creating the failure. This action causes all packets to be dropped for any path that traverses the failed link. In a real network, routers may reroute around some types of prolonged failures, but there are also many classes of failures for which re-routing will not occur (*e.g.*, configuration errors and other types of failures that do not result in the loss of "hello" packets but may still prevent packets from being forwarded). To determine how long these types of prolonged failures must last to allow the system to detect them, we slow Quagga's OSPF timers down so that routers do not re-route around long failures.

We fail each of the 13 links in Abilene covered by our measurements for durations of 5, 10, 30, 50, and 80 seconds. Running this experiment takes two hours and three minutes. We store and timestamp the results of all probes and the confirmation process at each monitor; all monitors are NTP-synchronized to within 10 milliseconds. At the end of each experiment, we copy these logs to a centralized server and run the identification phase offline.

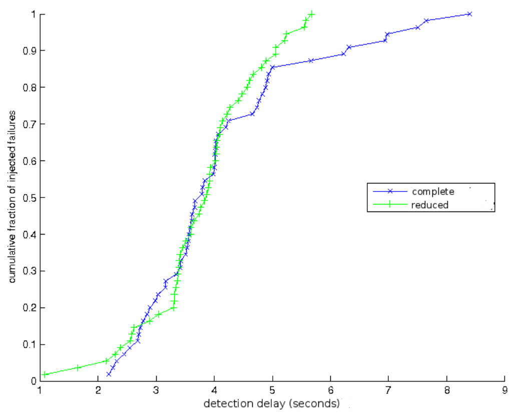

Figure 3: Distribution of detection time for link failures.

4. RESULTS

We now evaluate the approach that we proposed in Section 2. We first evaluate the ability of our approach to detect failures by injecting failures on the emulated Abilene topology that we run on the VINI testbed (Section 4.1). We then evaluate its ability to identify the location of these failures (Section 4.2). We evaluate the extent to which the approach can detect and identify failures both using the *complete* set of paths from monitors to destinations and the *reduced* set of paths from offline path selection.

4.1 Detection

We say that a failure is *detected* if the confirmation process reports it. The complete set covers all 13 failed links and detects almost all link failures we inject that last longer than 5 seconds. The reduced set of paths also detects all link failures, albeit with a considerably smaller number of paths than the complete set: 15 paths, as opposed to 58,322.

Figure 3 shows the distribution of detection delays for each failure type. The detection time corresponds to the time between the confirmation of the failure and the time we injected the failure. Most failures take between 3 and 5 seconds to detect. Given that the cycle time when probing with the complete set of paths is 3.45 seconds as opposed to just 0.45 seconds with the reduced set, it may seem counterintuitive that, for about 80% of all link failures, the detection delay when probing with the complete set of paths is about the same as that when we probe with the reduced set of paths. This similarity arises because many paths in the complete set may probe the same link or links in the Abilene topology within a single "cycle". Thus, even though it may take longer to probe the complete set of paths, any given link in the topology may be probed multiple times (and, hence, more frequently) in a single cycle.

4.2 Identification

In this section, we evaluate the extent to which the tomography algorithms can successfully identify the *location* of failures within the Abilene topology. We first evaluate the accuracy of the identification algorithms for various failure lengths, and for various failures in the topology. Then, we evaluate the time taken to run identification.

Accuracy. detection algorithms. We apply an existing binary network tomography algorithm, NetDiagnoser [5], to the set of probes collected from the set of monitored paths as determined from offline monitor selection, as described in Section 2. We determine whether this approach can successfully identify the location of the failures we inject into the emulated Abilene topology on the VINI testbed. Unfortunately, even in a controlled setting like VINI, emulated network links are in fact paths through the wide-area Internet, which may introduce transient faults that we did not inject.

	Complete	Reduced
lost probes	6,000	2,927
confirmed failures	1,953	1,855
correlated failures	69	58

Table 1: Number of lost probes vs. number of failures.

Table 1 shows the effectiveness of failure confirmation and correlation for reducing the number of alarms that do not correspond to failures. It presents, the total number of probes lost, the number of failures reported after confirmation, and the number of failures that exist after we correlate confirmed path failures into related failure events. Confirmation removes approximately more than two-thirds of spurious failures for the complete set of paths and about one-third of spurious failures for the reduced set of paths. The correlation mechanism then groups the confirmed failures from different monitors together.

The number of failures we detected after applying correlation in each case (69 for the complete set of paths and 58 for the reduced set of paths, some of which are false alarms) is much closer to the 65 failures that we actually injected (13 link failures for five different failure durations). Table 2 shows that probing with only the reduced set of paths results in considerably fewer false alarms. False alarms are faults that are reported but that we did not inject; given that the experiments run on an operational network, some of the false alarms we report might correspond to other failures that we did not inject, which might explain the higher number of false alarms for the longer failures. Probing using the reduced set of paths detects all injected failures that last ten seconds or longer, with no false alarms, except for the case of 80-second failures (which may be failures in the underlying infrastructure).

Table 2 shows that we cannot typically identify link failures that last for shorter than 5 seconds. If a failure is too short, reachability along respective paths may recover before the complete set of monitors can measure the failed link from a collection of paths. Thus, the failure must last long enough to allow for probing the failed interface, confirming the failure, and grouping the path with other paths that have

	Failures	Failure duration (seconds)				
		5	10	30	50	80
Complete	identified	4	12	13	13	13
	false alarms	0	0	3	1	10
Reduced	identified	3	13	13	13	13
	false alarms	0	0	0	0	3

Table 2: Identification of single link failures.

observed the same failure. Figure 3 shows that confirming a failure typically requires between 3 and 5 seconds. It takes another 5 seconds to correlate the the path failures into a single event, and correlation can sometimes take as long as 20 seconds even when using just the reduced set of paths.

Speed. We evaluate the identification delay of each link failure by determining the time the coordinator receives the first notification of a failed path related to that failure until it completes running the NetDiagnoser algorithm. On average, the time to identify a failure is around 20 seconds using the reduced set of paths and 30 seconds using the complete set of paths. This time includes three components: the time to confirm the failure (*detection delay*), the time to correlate measurements from multiple monitors (*correlation delay*), and the time to run the tomography algorithm. As shown in Figure 3, detection time typically ranges from 3 to 5 seconds. Correlating failures into distinct failure events (as described in Section 2) can take as long as ten seconds *after* the probes observe a complete set of related failures. Using only reduced set of paths reduces correlation time by as much as a factor of three. The average time for running NetDiagnoser with the complete set of paths is 17.69 seconds; using the reduced set of paths reduces this time to 10.2 seconds.

5. RELATED WORK

Network tomography. There has been a large body of work on statistical methods for fault identification [6, 8, 10, 12, 17] and on network tomography (Castro *et al.* [4] present a detailed survey on network tomography techniques).

We leverage previous work in "binary network tomography", which uses tomography to detect and locate failures. Duffield [6] first proposed binary tomography to detect the failed link at a tree topology with one monitor and multiple targets. More recently, NetDiagnoser [5] and Kompella *et al.* [12] extended binary tomography to a scenario with multiple monitors and destinations. These works evaluated their algorithms in simulation and with trace-based analysis. Some previous work has attempted to deploy and measure the performance of tomography techniques in operational networks [2, 7, 12, 14]. None of the experimental studies of tomography evaluate the extent to which practical challenges limit accuracy of fault detection.

Active monitoring and path selection. Previous work has proposed two approaches for reducing detection time: combining active and passive measurements [18] and reducing the set of monitored paths [13] (the approach in this paper).

Hubble [9] combines monitoring of BGP feeds and probes to a selected set of prefixes to monitor the wide-area Internet for reachability faliures. Hubble can detect reachability problems within about 15 minutes. Given that we focus on a single ISP, we use different techniques to select a small set of paths at a faster rate, which allows our approach to detect shorter reachability problems. Additionally, Hubble's inferences have not been verified against any ground truth, whereas we evaluate the limits of our techniques in a controlled environment.

6. DISCUSSION & RESEARCH AGENDA

Unfortunately, a gap remains between research in network tomography and practical systems for scalable network monitoring. Our evaluation highlights tradeoffs and challenges for making tomography practical, which we discuss below.

Fast detection vs. scalability: Quickly detecting failures using active monitoring requires frequently probing paths. Conventional network tomography algorithms assume the ability to probe all paths frequently, but in real networks, the set of paths may be large, which introduces a tradeoff between measurement overhead and detection time. Our evaluation shows that reducing the total number of paths probed can help reduce failure detection time (recall from Figure 3 that the reduced set of paths results in shorter detection times for about 20% of paths) while still maintaining accurate identification (similar detection rates with a much lower false positive rate than with the complete set of paths). In our future work, we are investigating whether other correlation algorithms could result in faster overall detection time.

Lack of a consistent view: Most conventional network tomography algorithms assume that if a particular link fails, then all paths that traverse that link will observe the failure. In practice, it is difficult to ensure that probes from different monitors will traverse the same failed link at similar (let alone synchronized) points in time. Our results in Section 4 show that it is more difficult to accurately identify shorter failures because not all monitors observe the failure; similarly, longer failures can result in false alarms if other factors (*e.g.*, unrelated failures, re-routing) create an inconsistent view of the topology when a link fails. In a real network, transient packet loss may introduce false alarms or result in inaccurate localization. Network dynamics may also introduce false alarms because routing protocols may re-route around a failure, changing the underlying network topology (as well as the fate of probe packets). In fact, these factors may make it difficult to ever obtain a consistent view of the network topology during a failure scenario; in our ongoing work, we are examining these effects in more detail and determining whether a version of network tomography that operates on data streams, rather than snapshots, might further reduce detection time.

Acknowledgments

We thank Christophe Diot for his guidance throughout this project. This work was supported by NSF grants CNS-0721581 and NSF CAREER Award 0643974, by the European Community's Seventh Framework Programme (FP7/2007-2013) no. 223850, and by Thomson. Part of this work was performed while Yiyi Huang was a summer intern at Thomson.

REFERENCES

[1] A. Bavier, N. Feamster, M. Huang, J. Rexford, and L. Peterson. In VINI Veritas: Realistic and Controlled Network Experimentation. In *Proc. ACM SIGCOMM*, 2006.

[2] R. Caceres, N. Duffield, S. Moon, and D. Towsley. Inference of Internal Loss Rates in the MBone. In *Proc. IEEE Global Internet*, 1999.

[3] R. Caceres, N. G. Duffield, J. Horowitz, D. F. Towsley, and T. Bu. Multicast-Based Inference of Network-Internal Characteristics: Accuracy of Packet Loss Estimation. In *Proc. IEEE INFOCOM*, 1999.

[4] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499–517, 2004.

[5] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot. NetDiagnoser:Troubleshooting network unreachabilities using end-to-end probes and routing data. In *Proc. CoNEXT*, 2007.

[6] N. Duffield. Network tomography of binary network performance characteristics. *IEEE Trans. Information Theory*, 52, 2006.

[7] N. G. Duffield, F. L. Presti, V. Paxson, and D. F. Towsley. Inferring Link Loss Using Striped Unicast Probes. In *Proc. IEEE INFOCOM*, 2001.

[8] S. Kandula, D. Katabi, and J.-P. Vasseur. Shrink: A Tool for Failure Diagnosis in IP Networks. In *ACM SIGCOMM Workshop on mining network data (MineNet)*, August 2005.

[9] E. Katz-Bassett, H. V. Madhyastha, J. P. John, , A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying Black Holes in the Internet with Hubble. In *Proc. USENIX Symposium on Networked Systems Design and Implementation*, 2008.

[10] I. Katzela and M. Schwartz. Schemes for Fault Identification in Communication Networks. *IEEE/ACM Trans. Networking*, 3(6), 1995.

[11] Keynote Systems – The mobile and Internet performance authority. `http://www.keynote.com/`.

[12] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Detection and Localization of Network Blackholes. In *Proc. IEEE INFOCOM*, May 2007.

[13] H. Nguyen and P. Thiran. Active measurement for multiple link failures diagnosis in ip networks. In *Proc. of Passive and Active Measurement Workshop*, 2004.

[14] H. Nguyen and P. Thiran. Network Loss Inference with Second Order Statistics of End-to-End Flows. In *Proc. Internet Measurement Conference*, 2007.

[15] Quagga Routing Suite. `http://www.quagga.net/`.

[16] RIPE. Test Traffic Measurements Service. `http://www.ripe.net/ttm/`.

[17] M. Steinder and A. S. Sethi. Probabilistic Fault Localization in Communication Systems Using Belief Networks. *IEEE/ACM Trans. Networking*, 12(5), 2004.

[18] M. Zhang, C. Zhang, V. Pai, L.Peterson, and R. Wang. PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-Area Services. In *Proc. USENIX OSDI*, Dec 2004.