NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data

Amogh Dhamdhere, Renata Teixeira, Constantine Dovrolis, Christophe Diot

NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data

Amogh Dhamdhere[†], Renata Teixeira[§], Constantine Dovrolis [†], Christophe Diot[‡]
[†] Georgia Tech [§] CNRS and Université Pierre et Marie Curie, Lip6 [‡] Thomson
{amogh,dovrolis}@cc.gatech.edu, renata.teixeira@lip6.fr, christophe.diot@thomson.net

ABSTRACT

The distributed nature of the Internet makes it difficult for a single service provider to troubleshoot the disruptions experienced by its customers. We propose NetDiagnoser, a troubleshooting algorithm to identify the location of failures in an internetwork environment. First, we adapt the well-known Boolean tomography technique to work in this environment. Then, we significantly extend this technique to improve the diagnosis accuracy in the presence of multiple link failures, logical failures (for instance, misconfigurations of route export filters), and incomplete topology inference. In particular, NetDiagnoser takes advantage of rerouted paths, routing messages collected at one provider's network and Looking Glass servers. We evaluate each feature of NetDiagnoser separately using C-BGP simulations on realistic topologies. Our results show that NetDiagnoser can successfully identify a small set of links, which almost always includes the actually failed/misconfigured links.

1. INTRODUCTION

We define *network unreachabilities* as network connectivity disruptions experienced by end users due to failures on the IP forwarding path. The failure of an IP path is an important source of connectivity disruptions (A study of the Sprint network shows that it is the most important source of disruptions for VOIP traffic [2]). Such unreachability problems are frustrating for end users and hard to detect for Internet Service Providers (ISPs). This is indicated by the large number of messages related to unreachability problems on mailing lists such as NANOG. A customer has no means to identify the cause of unreachabilities on her own. An ISP can detect problems in its network, but does not know how these problems impact its customers. It is beneficial to both parties to locate the cause of these unreachabilities. Customers expect providers to match their service-level agreements or otherwise to obtain financial compensation. ISPs would benefit from locating a problem on their network before their customers complain. ISPs would also benefit from locating remote problems outside their network, so that they know where to complain or how to avoid the problematic network.

In this work, we focus on network unreachabilities that are *non-transient*, i.e., they cannot be recovered by routing protocols. Several events can can lead to such non-transient failures. Failures of physical links and routers affect all IP paths that depend on them. Router misconfigurations (such as incorrectly set BGP policies or packet filters) can cause a link to fail "partially", meaning that the link works for a subset of paths but not for others. Unfortunately, path redundancy (due to multihoming or load balancing) does not always protect against multiple correlated link or router failures, and router misconfigurations that could prevent a backup link from coming up. Such failures are non-transient in nature because they can only be resolved by the intervention of a network operator. The diverse nature of failures makes the troubleshooting problem even harder.

To address these issues, we propose a troubleshooting algorithm for ISPs to identify the location of a network failure *by combining information from end hosts with control plane information from their own network.* The algorithm relies on an overlay of *troubleshooting sensors* located at end hosts in multiple ASes. These sensors perform traceroute-like measurements in a full mesh, the results of which are used by a troubleshooter located at the ISP's Network Operation Center (NOC). The placement of the sensors is fixed (e.g., sensors can be co-located with end hosts and at popular service locations such as web-server farms, data centers, etc.), and the algorithm aims to diagnose only those failures that lead to unreachability among some sensors. This troubleshooting system corresponds to at least two realistic scenarios. A DSL provider could deploy troubleshooting sensors in gateways located at customer premises, and these sensors would help troubleshoot problems experienced by customers. A third party (see www.nettest.com, for example) could also provide a trou-

*CoNEXT'*07, December 10-13, 2007, New York, NY, U.S.A.

bleshooting service to end users that installs the troubleshooting software on their own machines.

Our starting point is a "Boolean tomography" approach [7, 6], which assumes either a "good" or "bad" state for links, and tries to identify the smallest set of links that explains the end-to-end measurements. Then, we design *NetDiagnoser* to handle several practical issues that arise in a multi-AS environment. NetDiagnoser introduces a number of important features. First, it uses information from rerouted paths, improving the accuracy in diagnosing multiple link failures. Second, it uses the concept of "logical links" to diagnose reachability failures due to router misconfigurations. Third, it introduces mechanisms to combine routing messages collected at an ISP with end-to-end probing data to improve the diagnosis accuracy. Finally, NetDiagnoser uses a heuristic to identify the AS(es) responsible for a failure, in case some ASes block traceroute-like probes.

We evaluate the performance NetDiagnoser under various failure scenarios using simulations with realistic multi-AS topologies. We find that a basic "Boolean tomography" algorithm has several limitations, and performs poorly in diagnosing multiple link failures and router misconfigurations. NetDiagnoser, on the other hand, is almost always able to identify the failed link(s), with a small number of false positives. This is true for various failure scenarios, such as multiple link failures, router failures and router misconfigurations. In situations where ASes block traceroutes, we find that the use of information from Looking Glass servers enables NetDiagnoser to identify the AS responsible for the failure.

The rest of this paper is organized as follows. Section 2 extends the classical Boolean tomography solution to the case of multiple sources and destinations in a multi-AS environment. Section 3 presents each feature of the NetDiagnoser algorithm. Sections 4 and 5 describe our evaluation methodology and the evaluation results, respectively. Section 6 discusses some practical issues with deploying NetDiagnoser. Section 7 presents a review of related work and we conclude in Section 8.

2. THE TOMOGRAPHY APPROACH

The classical approach to infer network-internal characteristics from end-to-end measurements is to use network tomography [7, 4, 3, 8]. In particular, "Boolean tomography" [7, 23] represents the state of the art in detecting the most likely set of link failures on a known topology. This section first presents the Boolean tomography problem as formulated by Duffield [7]. Then, it adapts this approach to the multi-AS environment to diagnose problems affecting an overlay of sensors. We call this the "multi-AS tomography algorithm", **Tomo**.

2.1 Boolean tomography

Boolean tomography is a class of network tomography in which links can have one of two states: "good" or "bad". If all links on a path are good, then the path itself is good, whereas if at least one link on a path is bad, then the path is bad. Duffield's formulation [7] considers a topology consisting of paths from a single source to multiple destinations, and assumes that this topology is accurately known. Failures of link(s) from this topology cause some of those paths to go to the "bad" state. Duffield [7] presents the "Smallest Common Failure Set" (SCFS) algorithm, which designates as bad only those links nearest the source that are consistent with the observed set of bad paths.

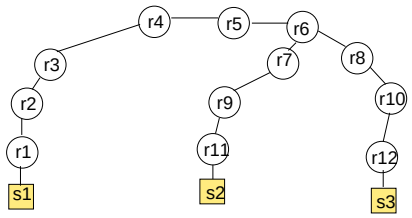

Figure 1: Boolean tomography problem

We use an example to demonstrate the Boolean tomography approach. Figure 1 shows a network topology, where squares denote sensors and circles denote routers. Let the single-source tree consist of the paths from s_1 to s_2 and s_3. If link $r9$-$r11$ fails, path $s1$-$s2$ will be broken, while the path $s1$-$s3$ will be working. To minimize false positives, the Boolean tomography algorithm marks the the link $r6$-$r7$ as failed, as it is the link closest to the source that can explain the path failures. In reality, however, any of the following links would cause the observed path failures: $r6$-$r7$, $r7$-$r9$, $r9$-$r11$, $r11$-$s2$. It is not possible to narrow down the set of potential failed links without more information.

2.2 The multi-AS case

We describe a multi-AS deployment scenario in which a provider has access to an overlay of troubleshooting sensors located at end hosts in multiple ASes. The location of these sensors is fixed, and the goal is to troubleshoot only those failures that cause some pairs of sensors to become unreachable. We call the provider that performs the troubleshooting process AS-X.[1] The goal of AS-X is to find the smallest set of links, which, if failed, would explain the failed paths. In case it cannot find the exact set of failed links, the goal is to find at least the AS(es) containing the failed link(s).

Figure 2 shows an internetwork topology that we use to illustrate this multi-AS deployment. Say that, at a certain time, the link $b1$-$b2$ fails, causing some pairs of sensors to become unreachable. The goal of AS-X is to determine that the link $b1$-$b2$ failed (or that the failed link lies in AS-B). It is important that the set of potential failed links returned by the algorithm should

[1]To simplify notation, "AS-X" will refer to the "troubleshooter entity in AS-X" in the rest of this paper.

include the actual failed links (no false negatives), even if it also includes some working links (false positives). In other words, *false positives are preferred to false negatives.* This narrows down the set of possibly failed links that the operator must check to determine the failure.

There are two reasons why the Boolean tomography approach [7] cannot be directly applied in this scenario: It requires knowledge of the complete topology, which is not possible in a multi-AS scenario, where each AS only has knowledge of its own topology. Futher, it works on a single source tree, while the paths between the sensors are from multiple sources to multiple destinations.

It is relatively simple to obtain the topology among the sensors in a multi-AS scenario by using active measurement tools like traceroute. The paths between the sensors are obtained from traceroutes, and the topology graph $\mathcal{G}$ is inferred from the union of these traceroute paths. Note that AS-X does not face issues such as router aliasing, since it is does not need to determine which interfaces belong to the same router. The set of links (pairs of interfaces) inferred from traceroutes is not the complete topology of each AS, but it does contain each link on the paths between sensors. This information is sufficient for a Boolean tomography approach. Every sensor uses traceroute to examine the reachability from itself to every other sensor, and sends the results to AS-X. AS-X then constructs the reachability matrix $\mathcal{R}$ from these measurements. In practice, load balancing could lead to multiple paths between a pair of sensors. In this case, a tool such as Paris traceroute [1] can discover all paths between a pair of sensors.

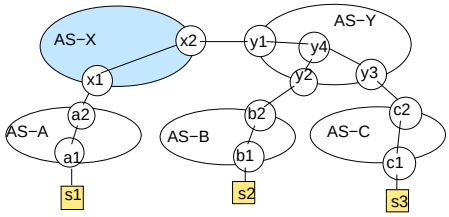

Figure 2: Multi-AS example

Given the inferred topology graph $\mathcal{G}$ and the reachability matrix $\mathcal{R}$, we are now faced with a multiple-source, multiple-destination version of the Boolean tomography problem stated in the previous section. Unfortunately, the SCFS algorithm [7] is not applicable in this case, as it is designed for a tree topology. Therefore, we introduce in the next section a multiple-source, multiple-destination Boolean tomography algorithm.

2.3 The multiple source, multiple destination Boolean tomography problem

Given a set of N sensors, $\mathcal{S} = \{s_1 \dots s_N\}$, we denote as p_{ij} the path from sensor s_i to sensor s_j, where $i, j \in 1 \dots N$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the directed graph constructed from the union of all traceroute paths.sensors.

At a certain time, some links from $\mathcal{E}$ fail, which causes some paths to fail. We associate with each link $\ell \in \mathcal{E}$ and each path p_{ij} a status of 0 (down) or 1 (up). Let $\mathcal{R}$ be the reachability matrix that reflects the status of each path between the sensors. $\mathcal{R}_{ij}$=1, if the path from sensor s_i to s_j is up, and $\mathcal{R}_{ij}$=0 otherwise. Similarly, let $f(\ell)$ denote the status of link ℓ. If $\mathcal{R}_{ij}$=1, then $\forall \ell \in p_{ij}, f(\ell)$=1. Let $\mathcal{W}$ be the set of constraints obtained from working paths. If $\mathcal{R}_{ij}$=0, then $\exists \ell \in p_{ij}, f(\ell)$=0. For each failed path p_{ij}, we define the failure set, $L_{ij} = \cup_{\ell \in p_{ij}} \ell$. Let $\mathcal{L}$ be the set of all failure sets.

The multiple-source, multiple-destination Boolean tomography problem is defined as follows: *Find the smallest set of links, H, which is consistent with the observations in $\mathcal{R}$.* In other words, H is the smallest set of links that "intersects" with each failure set, i.e., $H \cap L_{ij} \neq \emptyset$, $\forall L_{ij} \in \mathcal{L}$. Further, H must not violate any of the constraints in $\mathcal{W}$, i.e., no link from H can appear on any working path.

The problem as stated here is exactly an instance of the *Minimum Hitting Set* problem. The optimization version of the Minimum Hitting Set problem is NP-hard, as it is the dual of the NP-hard *Min Set Cover* problem [12]. It has been shown that a greedy heuristic approximates the solution to Min Set Cover within an approximation ratio of $log|\mathcal{U}|$, where $\mathcal{U}$ is the set of elements from which the hypothesis set can be chosen [13]. Due to the duality of the problems, a similar greedy heuristic for the Minimum Hitting Set problem has the same approximation ratio $log|\mathcal{U}|$. We present our version of the greedy heuristic in the next section.

2.4 Greedy heuristic

We now describe a greedy heuristic, called **Tomo**, to solve the Minimum Hitting Set problem on the inferred graph $\mathcal{G}$ and reachability matrix $\mathcal{R}$. Let H be our hypothesis set and n_f be the number of failed paths, so that we have n_f failure sets in $\mathcal{L}$. Let $\mathcal{U}$ be the union of these failure sets, $\mathcal{U}$=$\cup L_{ij}$ $\forall L_{ij} \in \mathcal{L}$. Note that $\mathcal{U}$ is a set containing the links from each failure set, while $\mathcal{L}$ is the set of failure sets. The actual set of failed links must come from the set $\mathcal{U}$, called the *candidate set.*

Algorithm 1 gives the details of Tomo. The greedy heuristic proceeds iteratively as follows. We first remove from the candidate set $\mathcal{U}$ each link that appears on a working path, as these links cannot be down. We maintain a set of unexplained failure sets, which are failure sets that the current hypothesis set does not intersect with. In each iteration, we compute, for each link $\ell \in \mathcal{U}$, the number of unexplained failure sets that ℓ intersects with (called the "score" of link ℓ). We then add the link (or set of links) with the highest score to the hypothesis set. This heuristic runs in polynomial time with the number of links in the candidate set.

Tomo constitutes our extension of the Boolean tomography approach [7] to the multi-AS scenario with multiple probing sources and destinations.

Algorithm 1 Tomo

Require: Set of all links $\mathcal{E}$
Require: Reachability matrix $\mathcal{R}$
Require: Set of equations from working paths $\mathcal{W}$
Initialize $\mathcal{L} = \emptyset$ {The set of failure sets}
For each $\mathcal{R}_{ij} = 0$, $\mathcal{L} = \mathcal{L} + L_{ij}$ {Add the failure set due to each broken path}
Initialize $H = \emptyset$ {Hypothesis set of failed links}
Initialize $\mathcal{L}_u = \mathcal{L}$ {Set of unexplained failure sets}
$\mathcal{U} = \cup L_{ij}\ \forall L_{ij} \in \mathcal{L}$ {The candidate set.}
Remove from $\mathcal{U}$ every link ℓ on a working path
while $\mathcal{L}_u \neq \emptyset$ AND $\mathcal{U} \neq \emptyset$ **do**
    for each link $\ell \in \mathcal{U}$ **do**
        $C(\ell)$=set of failure sets in $\mathcal{L}_u$ containing ℓ
        score(ℓ) = $|C(\ell)|$ {The number of unexplained failure sets that ℓ intersects with.}
    end for
    $\mathcal{L}_m = \{\ell_m | \ell_m = argmax_{\ell \in \mathcal{L}}$ score$(\ell)\}$ {The set of links with the maximum score.}
    for each link $\ell_m \in \mathcal{L}_m$ **do**
        $H = H \cup \{\ell_m\}$ {Add ℓ_m to hypothesis set}
        $\mathcal{L}_u = \mathcal{L}_u - C(\ell_m)$ {All failure sets in $C(\ell_m)$ are now explained. Remove from $\mathcal{L}_u$.}
        $\mathcal{U} = \mathcal{U} - \{\ell_m\}$
    end for
end while
return H

2.5 Limitations

Several practical issues limit the effectiveness of Tomo in diagnosing unreachability failures in the Internet:

1. Links can fail "partially". Router misconfigurations (such as incorrectly set BGP policies or packet filters) may cause a link to fail only for a subset of the paths using that link. Tomo cannot detect such failures because it assumes that if a link is up, then *each* path using that link is up.

2. There is life after link failures. Routing protocols (either IGP or BGP) try to reroute around failed links. Tomo misses information by not considering the paths obtained after rerouting.

3. Inference using only edge data can be inaccurate. Tomo uses only end-to-end probing data from sensors. Control-plane information from AS-X could be used to improve the diagnosis.

4. Some ASes block traceroute. If traceroutes are incomplete, then Tomo does not have access to the complete graph $\mathcal{G}$.

3. NETDIAGNOSER

This section presents the NetDiagnoser algorithm. This algorithm includes four features designed to overcome the aforementioned limitations of the Boolean tomography approach. Sections 3.1 and 3.2 comprise the version of NetDiagnoser that uses only end-to-end probing data (**ND-edge**) and can be used, for example, by a third-party that provides a troubleshooting service to end-users without the cooperation of ISPs. Section 3.3 presents the version of NetDiagnoser that uses routing data (**ND-bgpigp**), and hence is appropriate for use by ISPs. Section 3.4 shows how NetDiagnoser deals with incomplete topology information due to blocked traceroutes. Here, we present the basic ideas behind these features. For more details, we refer the reader to [5].

3.1 Locating router misconfiguration failures

BGP speaking routers in the Internet typically use complex policies to control which routes they exchange with their neighbors. Unfortunately, the complexity of these policies leads to frequent misconfiguration [10]. To use again the example of Figure 2, a misconfiguration could cause AS-Y to stop announcing to AS-X the route towards AS-B, while it announces the route towards AS-C. This results in the situation where link x_2-y_1 works for path s_1-s_2, while it does not work for path s_1-s_3.

To handle failures due to such misconfigurations, we describe a procedure that AS-X uses to extend $\mathcal{G}$ with *logical links*. For each link in the IP graph, AS-X determines the ASes corresponding to the endpoints of the link (using a well-known IP-to-AS mapping technique [19]). If the link is intradomain, then AS-X simply includes it in the graph as is. If the link is interdomain, then it is replaced by a set of *logical links*. To capture router configurations and policies at the finest granularity, we should ideally have logical links on a per-prefix basis. However, this could result in a very large graph (some tier-1 ISPs have more than 200,000 prefixes in their routing tables). Further, BGP policies are usually set *on a per-neighbor basis*, rather than on a per-prefix basis [20], which means that logical links on a per-neighbor basis should be sufficient. Even though some core ASes in the Internet could have more than 2000 neighbors, logical links are added only for the *neighbors seen in the traceroutes*. As long as sensors are not deployed in each AS in the Internet, the graph is tractable. In [5], we discuss scalability issues in more detail.

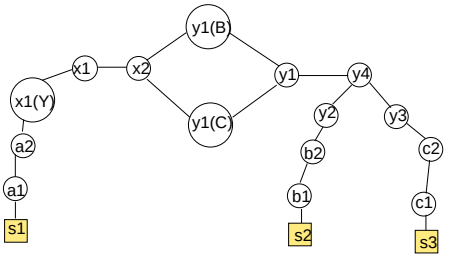

Figure 3: Example of logical links

Figure 3 shows an example of how the IP-level graph of Figure 2 can be converted into a graph with logical

links. Suppose the traceroutes are from s_1 to s_2 and s_3. For this set of paths, AS-X sees one out-neighbor AS-Y for AS-X, and two out-neighbors AS-B and AS-C for AS-Y. For path p_{12}, the inter-AS link a_2-x_1 is replaced by the links a_2-$x_1(Y)$ and $x_1(Y)$-x_1, while x_2-y_1 is replaced by the links x_2-$y_1(B)$ and $y_1(B)$-y_1. Logical links are formed similarly for path p_{13}.

We use the same example to show how logical links can identify router misconfigurations. Under normal circumstances, y_1 announces to x_2 the routes it learns from ASes B and C (routes towards sensors s_2 and s_3). Now, a misconfiguration at the outbound route filter of y_1 causes it to announce to x_2 *only the route towards B, while it does not announce the route towards C*. As a result, the path s_1-s_2 works, while s_1-s_3 fails. The Tomo algorithm applied on the original IP graph would mark the link x_2-y_1 as non-failed, as it carries a working path. However, when Tomo is used on the graph of Figure 3, the logical links x_2-$y_1(C)$ and $y_1(C)$-y_1 are added to the hypothesis set. The algorithm can now identify the link $x_2 - y_1$ as the misconfigured link.

3.2 Using information from rerouted paths

The Tomo algorithm uses the traceroutes *before* the failure to construct $\mathcal{G}$, but not the traceroutes after the failure. Therefore, Tomo is not able to use information from paths that were rerouted, and work after the failure event. If a path is rerouted[2] but still works after the failure, then every link on the new path must be working. This path can be added to the set of constraints obtained from working paths ($\mathcal{W}$) described earlier. This information helps to reduce the size of the hypothesis set, and also to detect multiple link failures. Consider a situation where multiple links fail simultaneously. If all the link failures can be recovered by rerouting, then all end-to-end paths work after the failure, and the troubleshooting algorithm is not invoked. On the other hand, if some failures cannot be rerouted, then some paths fail while others are rerouted. Since Tomo does not use the paths after rerouting, it is not able to identify the link failures that caused path rerouting.

Let $T-$, T and $T+$ be three time instants such that $T- < T < T+$ and T be the time at which the failure event occurs. Let p_{ij} be a path that is working both before and after the failure event. At time $T-$, p_{ij} consists of the set of links p_{ij}^{T-}=$\{\ell_1,\ell_2,\ell_3,\ell_4\}$, and at time $T+$, p_{ij}^{T+}=$\{\ell_1,\ell_2,\ell_5,\ell_6\}$. Comparing p_{ij}^{T-} and p_{ij}^{T+}, we find that the links ℓ_3, ℓ_4 are in the old path, but not in the new path. This means that the failure of link ℓ_3 or ℓ_4 would explain the rerouting of path p_{ij}. We call $\{\ell_3, \ell_4\}$ a *reroute set*, and obtain one reroute set R_{ij} for each rerouted path.

$$R_{ij} = \{\ell \quad | \quad \ell \in p_{ij}^{T-} \quad AND \quad \ell \notin p_{ij}^{T+}\}$$

We use rerouted paths in Tomo while calculating the "score" of a link. Let $C(\ell)$ be the set of unexplained failure sets that contain ℓ, and $R(\ell)$ the set of unexplained reroute sets that contain ℓ. The score of ℓ is $a|C(\ell)| + b|R(\ell)|$, where a and b are weights that reflect the relative importance of failures and reroutes. In this work, we assume a,b=1. The rest of the algorithm is the same as Tomo. We call the version of NetDiagnoser that uses logical links and reroute information **ND-edge**.

3.3 Using control-plane information

Control plane information in the form of BGP and IGP messages from AS-X can be useful in detecting the cause of unreachability problems. For instance, AS-X can directly detect a link failure from IGP "link down" messages. AS-X can infer that a sensor is unreachable if it receives a BGP withdrawal for the destination prefix that contains that sensor. However, it is not easy to determine the impact of a routing event on end-to-end paths. Further, network operators only have access to routing messages from their own network. Hence, we propose a mechanism that combines *control-plane information from AS-X with the end-to-end probing data* to obtain better troubleshooting performance.

This algorithm is called **ND-bgpigp**. Using IGP messages from AS-X is straightforward, as these messages directly indicate the status of IGP links. When AS-X receives a "link down" message for ℓ, it adds ℓ to the hypothesis set H. We illustrate with an example how AS-X uses BGP withdrawals. Consider again the example in Figure 2 and the case where the paths s_2-s_1 and s_3-s_1 have failed, while all other paths are working. The Tomo algorithm returns a hypothesis set containing links y_4-y_1, y_1-x_2, x_2-x_1, x_1-a_2, a_2-a_1 and a_1-s_1. Now suppose that after the failure, x_1 receives a route withdrawal from a_2 for a prefix A, corresponding to sensor s_1.[3] This indicates that the failed link must be on the portion of the path between AS-X and s_1. AS-X can remove links y_4-y_1, y_1-x_2, x_2-x_1, and x_1-a_2 from H, thus reducing the size of the hypothesis set.

3.4 Dealing with blocked traceroutes

NetDiagnoser so far assumes that traceroute measurements are complete, i.e., every hop on all paths between sensors responds with a valid IP address. However, certain ASes block traceroutes for privacy reasons, and almost all routers rate-limit ICMP responses. Additionally, routers sometimes send ICMP responses from a different interface than the one receiving the incoming packet, and this interface could have a private IP address. In such cases, traceroutes will either contain "stars" for non-responding hops, or have hops with

[2]Rerouted paths can be distinguished from path changes due to load balancing by using a tool such as Paris traceroute to determine all possible paths. For simplicity, the remainder of the analysis and evaluation does not consider load balancing.

[3]AS-X should use a withdrawal message only if it is for the most specific prefix known for a destination.

private IP addresses. We call all such hops *unidentified hops* or UHs. UHs represent a major challenge for NetDiagnoser, because it relies on traceroute to construct $\mathcal{G}$. If the failed link falls in an AS that blocks traceroute, then it is impossible to exactly determine that link. We make the assumption that if an AS blocks traceroutes, then no router in that AS will respond, and if an AS allows traceroutes, *each* router in that AS will respond with a valid IP address. We disregard the case where only a few routers in an AS do not respond due to ICMP rate limiting. This problem can be solved by repeating the traceroute for the source-destination pair.

We introduce a feature in NetDiagnoser that can be used to identify the *AS(es)* with failed links, when the traceroute graph contains UHs. We call this algorithm **ND-LG**, because it uses information from Looking Glass servers [21]. Looking Glass servers located in an AS allow queries for IP addresses or prefixes, and return the AS path as seen by that AS to the queried address or prefix. The ND-LG algorithm proceeds in two steps: First, we map each UH to an AS. Next, we cluster links with UHs that could actually be the same link.

AS-X maps UHs to ASes using Looking Glass servers. For example, consider the topology shown in Figure 4 and suppose that p_{ij} is the path that contains UHs. Let the IP path be s_i-x-u_1-u_2-u_3-y-s_j, where u_1, u_2, u_3 are UHs. Let s_i, x map to AS A and y, s_j map to AS C. The goal is to map the UHs u_1, u_2, and u_3. To do this, AS-X needs to obtain the AS path from the source to the destination. AS-X queries the Looking Glass of AS-A to obtain the AS path to destination s_j. If the Looking Glass of the source AS is not available, then AS-X queries the first available Looking Glass on the path. Suppose the Looking Glass returns the AS path *A-B-C* from s_i to s_j. The UHs u_1, u_2 and u_3 can clearly be marked as belonging to AS-B. However, AS-X may not always be able to map UHs to ASes unambiguously. In the same example, suppose the AS path returned is *A-B-D-C*. In this case, we cannot say which of the UHs belong to AS-B and which belong to AS-D. Hence, we assign these UHs the combined tag $\{B, D\}$. This means that these UHs could belong either to AS-B or AS-D. For mapping downstream UHs, AS-X can use its own BGP information to determine the AS path to the destination. Looking Glasses are useful in mapping UHs that are upstream of AS-X.

After mapping each UH to a particular AS, AS-X still needs to infer whether two UHs are actually the same hop. Some links may have a UH as one (or both) endpoints. Let us call such links *unidentified links*. An unidentified link can appear in only one path, and hence, appears in at most one failure set. Consider the case of two unidentified links ℓ_1=u_1u_2 and ℓ_2=u_3u_4. AS-X uses the following rules to determine if ℓ_1 and ℓ_2 are actually the same link: (i) u_1 must have the same AS tag as u_3, and u_2 must have the same AS tag as u_4; (ii) ℓ_1 and ℓ_2 must not occur on the same path; (iii)

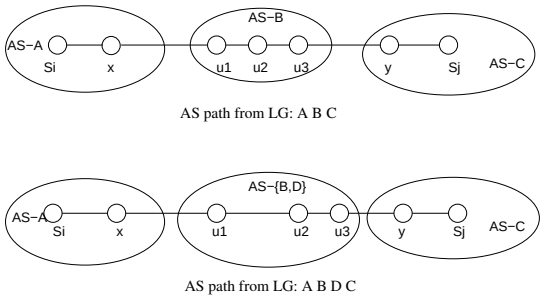

Figure 4: Mapping UHs to ASes in ND-LG.

ℓ_1 and ℓ_2 must appear in the same number of failure sets (either zero or one). Using these three rules, AS-X constructs a set of links for each unidentified link ℓ, called linkCluster(ℓ). If link $\ell' \in$ linkCluster(ℓ), then it is the same link as ℓ. Now, to determine the score of ℓ in Tomo, we must add the number of path failures explained by ℓ', $\forall \ell' \in$ linkCluster(ℓ).

4. EVALUATION METHODOLOGY

We evaluate each feature of NetDiagnoser via simulation. There are two compelling reasons in favor of a simulation-based evaluation: It is the only reliable way to know the ground truth; and we can evaluate a vast parameter space to understand the effects of our assumptions and practical constraints on troubleshooting performance. Though evaluation on a real network is important, we believe that evaluation via extensive simulation is a necessary first step. We use the open source BGP simulator C-BGP [25], which allows us to simulate internal and multi-AS topologies, IS-IS and BGP message exchange, and traceroute.

Network topology.

We use the topology of the "research part" of the Internet, constructing the multi-AS topology as follows. We use Abilene, GEANT, and WIDE[4] as the core ASes which are connected in full mesh. Using publicly available BGP traces from Abilene, GEANT and Routeviews [26], we obtain the AS path from these core networks towards the stubs, which gives us a tree rooted at each of the core networks. To allow the simulations to run in a reasonable amount of time, we scale down this topology, by performing a breadth-first search on the graph starting from the core networks, and select the first 165 ASes. This gives us a topology with three core ASes, 22 tier-2 ASes (of which 50% are multihomed), and 140 stub ASes (of which 25% are multihomed). The interconnection points for Abilene, GEANT and WIDE are known exactly. For interconnecting other ASes, we randomly choose routers from these ASes as the con-

[4]Their topologies and peering connectivity are available at: abilene.internet2.edu, www.geant.net, and www.wide.ad.jp.

nection points, and reproduce the inter-AS connectivity (including multihoming) found in the measurements[5].

For the intradomain topologies, we use the accurate router-level topology of Abilene, GEANT, and WIDE using IS-IS traces and topology maps found on their web pages. Since it is extremely difficult to get the accurate topology for other networks, we use a 12-node hub-and-spoke topology for the tier-2 ASes, which is similar to some intradomain topologies we have observed. We model stubs as single-router ASes. It is possible that this topology contains less path diversity than what is found in the Internet. However, path diversity only determines the number of failure instances that lead to unreachabilities. It does not influence the performance of our algorithms, since they are invoked only for the failures that lead to unreachabilities.

Sensor placement and diagnosability.

Even though the network topology is fixed, varying the sensor placement can result in significantly different graphs. Note that the graph $\mathcal{G}$ used by our algorithm is not the complete topology, but only the part of the topology inferred by the traceroutes. As a result, the traceroute graph (and not the network topology) is the main parameter that determines how hard or easy it is to identify the failed link(s). We do not specifically study sensor placement in this work. However, it is still useful to define a metric that quantifies "diagnosabiity" i.e. the difficulty of diagnosing failures on a graph.

Intuitively, it is easier for Tomo to diagnose failures in a graph, $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$, when each link $\ell \in \mathcal{E}$ is traversed by a unique set of paths. In this case, each link failure produces a different reachability matrix $\mathcal{R}$. When there is a set of links L such that the same paths cross each link in L, the failure of any link $\ell \in L$ produces the same reachability matrix $\mathcal{R}$, and it is hard to identify the exact link that failed.

Based on this intuition, we define a metric for diagnosability. For each $\ell \in \mathcal{E}$, let the hitting set of ℓ, $h(\ell)$, be the set of paths that traverses ℓ, and $\mathcal{HS}(\mathcal{G}) = \{h(\ell), \ell \in \mathcal{E}\}$ be the set of all distinct hitting sets. We define the *diagnosability* of $\mathcal{G}$ as:

$$\mathcal{D}(\mathcal{G}) = \frac{\text{number distinct hitting sets}}{\text{number probed links}} = \frac{|\mathcal{HS}(\mathcal{G})|}{|\mathcal{E}|}.$$

$\mathcal{D}(\mathcal{G})$ takes values between 0 and 1. $\mathcal{D}(\mathcal{G})$=1 means that we can precisely identify any single link failure in $\mathcal{G}$, whereas $\mathcal{D}(\mathcal{G})$=0 only when the number of paths is 0, in which case diagnosability is obviously 0.

We illustrate the relation between sensor placement and diagnosability with a simple case study. In the first placement ("same AS"), N sensors are placed in the same AS. In this case, there is more diversity in how the paths traverse the set of links in that AS. Consequently, we expect diagnosability to be high for this placement. In the second placement ("distant AS"), $N/2$ sensors are placed in one AS, and the remaining $N/2$ are placed in another AS. In this scenario, all paths from sensors in one AS to the other cross the same sequence of links. The failure of any link in this sequence produces the same reachability matrix, leading to low diagnosability. In a third placement, most sensors are placed as in the "distant AS" placement, but some sensors are placed at intermediate nodes between the networks ("distant AS, split path"). The goal is to have a more diverse coverage of the links between the two ASes, which should improve diagnosability. In a fourth placement, sensors are placed at randomly chosen edge networks ("random"). Figure 5 shows the diagnosability for these placements as we vary the number of sensors. We see that the "same AS" placement shows much better diagnosability than the "distant AS" placement. The placement of sensors on the sequence of links between the ASes improves the diagnosability of the "distant AS" placement. The random placement shows the worst diagnosability.

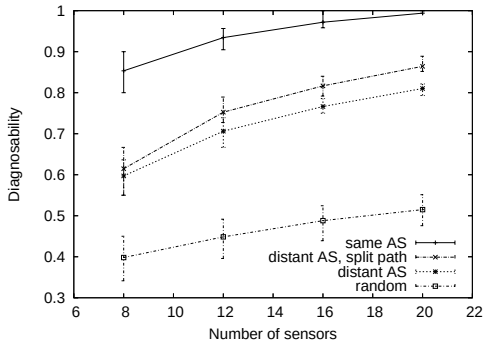

Figure 5: Sensor placement and diagnosability

In the simulation results of the following section, the sensors are all placed at randomly chosen stub ASes, meaning that *we evaluate the performance of NetDiagnoser under the worst-case scenario.* The number of sensors is fixed at N=10 (experiments with N ranging from 5 to 100 show similar trends). We compute the diagnosability for all simulations with 10 sensors and find that the diagnosability ranges from 0.25 to 0.6. To relate these values of diagnosability to a real-world setting, we perform the following measurement. We randomly select 10 nodes (from different ASes) from the PlanetLab infrastructure as our sensors. We use traceroute to obtain the paths between every pair of sensors, and compute the diagnosability of this set of paths. The value we obtain is 0.41, which means that we investigate a range of diagnosability which is similar to that seen in real topologies with the same number of sensors.

Failure scenarios.

We evaluate our algorithms under various failure sce-

[5]The topology and simulation scripts are available at `www.cc.gatech.edu/~amogh/NetDiagnoser.html`

narios consisting of link failures, router failures and router misconfigurations. We simulate link failures by randomly breaking x links in $\mathcal{E}$, where $x = \{1, 2, 3\}$. In the case of multiple simultaneous failures, the failed links could be in different ASes. There are two classes of link failures. If the link failure can be recovered by routing protocols, then we call such a failure *reroutable*. A reroutable link failure leads to rerouted paths. If a link failure cannot be recovered by the routing protocols, then we call such a failure *non-recoverable*. A non-recoverable failure leads to failed paths. We simulate router failures by breaking all links attached to a router. We simulate router misconfigurations as follows. We choose an interdomain link at random from among the probed paths, and choose the router at one end of that link as the target router to be misconfigured. We then choose some route(s) from the routing table of the target router and apply an export-filter such that the selected routes are not advertised to the peer (only the peer at the other end of the misconfigured link), thus simulating a BGP policy misconfiguration. After introducing the misconfiguration, we let C-BGP converge to a stable network state, and then perform a new set of traceroutes to obtain the paths after the failure.

Metrics.

Let F be the set of failed links and H the hypothesis set defined in Section 2. We use the metrics sensitivity and specificity to evaluate the "goodness" of the hypothesis set produced by our algorithm. Sensitivity and specificity are well-known evaulation metrics used in medical diagnosis. Sensitivity is defined as:

$$\text{sensitivity} = \frac{\text{number true positives}}{\text{total number failed}} = \frac{|F \cap H|}{|F|}$$

Sensitivity measures how well the algorithm is able to detect the actual failed links. If the number of false negatives is high, then sensitivity will be low. Specificity is defined as:

$$\text{specificity} = \frac{\text{number true negatives}}{\text{number non-failed}} = \frac{|(\mathcal{E} \setminus F) \cap (\mathcal{E} \setminus H)|}{|\mathcal{E} \setminus F|}$$

Specificity measures the number of false positives produced by the algorithm. If the hypothesis set contains many false positives, specificity will be low. Both sensitivity and specificity vary from zero to one, and high values are desirable for both metrics. Sensitivity and specificity can also be defined at the granularity of ASes with failed links, rather than actual links.

Given that the number of links in the graph ($|\mathcal{E}|$) is orders of magnitude higher than the number of links we expect to fail at the same time ($|F|$), it is expected that specificity will always be close to one for our algorithms. For instance, say that $|\mathcal{E}|$=150 (in the simulation results that we report, the number of probed links is around this number) and consider single link failures (i.e., $|F|$=1). If $|H|$=10, the specificity would be $140/149 = 0.939$. Consequently, we are very interested in specificity increases, even if they are quite small.

5. EVALUATION RESULTS

This section evaluates the NetDiagnoser algorithm. Unless otherwise stated, we report the results of 1000 simulation runs for each scenario, with 10 random sensor placements and 100 failures per placement.

5.1 Evaluation of Tomo

Tomo represents our starting point for troubleshooting in a multi-AS scenario. First, we evaluate Tomo under link failures and router misconfigurations.

The top graph in Figure 6 shows the cumulative distribution of the sensitivity of Tomo for one, two, and three link failures across 1000 simulation runs. Not surprisingly, Tomo is able to find the failed link when there is only a single link failure (sensitivity is one for almost all simulation instances). This is because for single link failures, Tomo is invoked only if the failure is non-recoverable. In this case, there are only failed paths, and no rerouted paths. As a result, Tomo does not miss any information from rerouted paths, and performs quite well. However, Tomo cannot identify the failed links under two and three link failures, showing a much lower sensitivity. This is because with multiple link failures, there may be some failures that are reroutable, and some that are non-recoverable, leading to both failed and rerouted paths. Since Tomo does not use reroute information, it cannot detect the link failures that lead to rerouted paths.

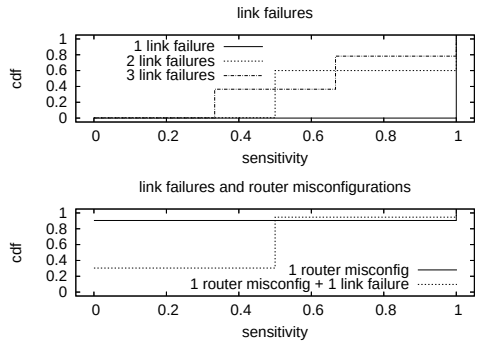

Figure 6: Tomo under different failure scenarios.

Tomo also shows poor performance under router misconfigurations. The bottom graph in Figure 6 presents the CDF of the sensitivity for one router misconfiguration and a combination of one router misconfiguration and one link failure. The main observation is that the sensitivity is zero in almost 90% of instances. Tomo cannot identify router misconfuration failures, because it assumes that any link carrying a working path must be working. This condition is clearly violated in the case of router misconfigurations.

These results confirm the limitations of Tomo in diagnosing failures more complex than single link failures. Next, we evaluate each feature of NetDiagnoser.

5.2 Comparing Tomo with ND-edge

In this section, we evaluate the version of NetDiagnoser that uses logical links and information from rerouted paths. Figure 7 shows that ND-edge achieves a signficant improvement in sensitivity as compared to Tomo. The top graph compares the sensitivity of Tomo and ND-edge for the case of three link failures. We find that ND-edge almost always has a sensitivity of one, whereas Tomo has low sensitivity. The results for one and two link failures are similar. The lower graph in Figure 7 compares the sensitivity of Tomo and ND-edge under a combination of router misconfiguration and link failures. We find that the sensitivity of ND-edge is almost always one. Again, Tomo could not identify the actual failed links in 90% of the instances.

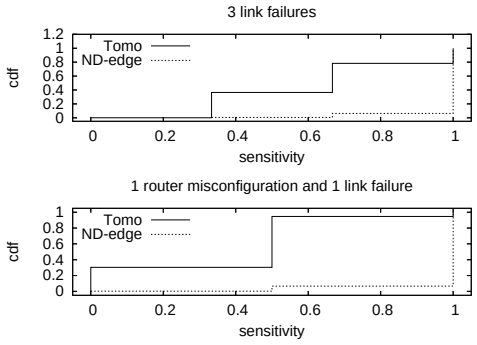

Figure 7: Sensitivity of Tomo and ND-edge.

This comparison shows that adding rerouting information and logical links to Tomo guarantees that in almost all cases, we achieve perfect sensitivity, i.e., $F \subset H$. To evaluate the size of the hypothesis set returned by ND-edge, we present the specificity of ND-edge in Figure 8, for the case of a single link failure and a single router misconfiguration. We find that ND-edge reports a high value (larger than 0.9) of specificity in all cases of single link failures, and close to or greater than 0.9 for two and three simultaneous link failures (not shown). This is true both for link failures and router misconfigurations. In fact, we observe from Figure 8 that ND-edge shows a *much better* specificity in detecting router misconfigurations. This is due to the fact that a router misconfiguration appears as a failed logical link in our graph, and the working paths allow the algorithm to eliminate several physical links from the hypothesis set.

To get a more intuitive feel for the specificity of ND-edge, we measure the size of the hypothesis set from each simulation run.[6] We find that for single link failures, ND-edge sometimes reports up to 12 links in the hypothesis set. While this number is small compared to the number of probed links (which is greater than 200 in our simulations), there are still some false positives.

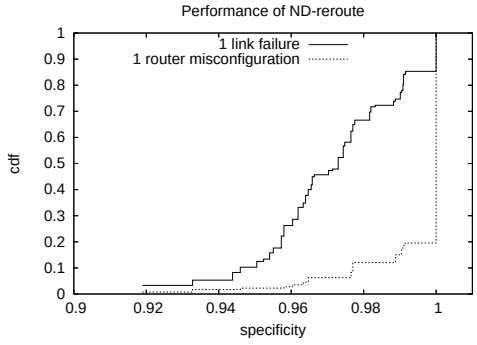

Figure 8: Specificity of ND-edge

Intuitively, the specificity (and consequently the size of the hypothesis set) of ND-edge depends on the diagnosability of the inferred graph. We vary the number of probing sources from 5 to 90, and calculate the diagnosability of the resulting set of paths, which spans the range from 0.1 to 0.9. The specificity varies with the diagnosability as shown in Figure 9, where each point corresponds to one (sensor placement, failure) pair. We find that the diagnosability directly affects the specificity of the algorithm. Although the specificity seems high (it is always greater than 0.75), the size of the hypothesis set is greater than 1 (indicating false positives) in more than 50% of the instances, particularly when the diagnosability is low. This indicates that using end-to-end probing data alone is not sufficient to obtain perfect specificity, especially when the diagnosability of the underlying graph is low. In the next section, we demonstrate how the addition of routing data improves specificity of the diagnosis.

In some cases, it may be more important for AS-X to identify the AS(es) in which the failed links lie, rather than the actual links. We evaluate the AS-sensitivity and AS-specificity of ND-edge. We find that in more than half of the cases, ND-edge can find the exact AS(es) containing the failure, and for over 90% of the instances there is at most one AS-false positive. The number of AS-false negatives is low, and in more than 90% of the cases, there are no AS-false negatives. Thus, ND-edge accurately detects the AS responsible for the failure.

Finally, we investigate the performance of ND-edge when dealing with router failures. This situation is similar to the failure of a Shared Risk Link Group (SRLG). In the case of an SRLG, the failure of a physical component results in the correlated failure of several IP links that depend on that component. We say that ND-edge detects a router failure if the hypothesis set contains at

[6]The size of the hypothesis set and the specificity are equivalent; given the number of links in the topology and the hypothesis set, it is easy to compute the specificity.

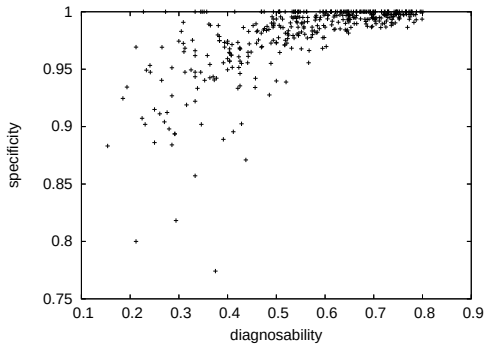

Figure 9: Diagnosability vs. specificity

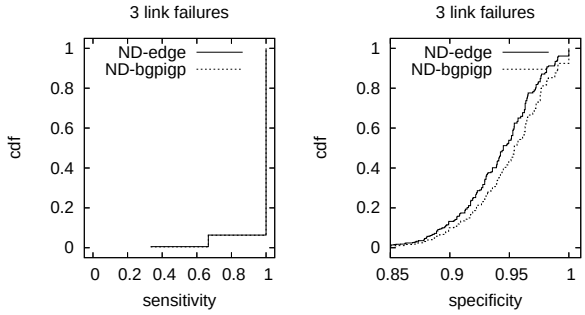

Figure 10: ND-edge vs. ND-bgpigp

least one link from the router that was failed. We find that in *each simulation run*, ND-edge is able to identify the router that failed. We further evaluate the sensitivity and specificity calculated using the set of links attached to the failed router. The results are similar to those for three link failures reported earlier.

To summarize, we find that ND-edge is quite effective at removing the limitations of Tomo in diagnosing complex failure scenarios such as multiple link failures, router misconfigurations and router failures.

5.3 Performance of ND-bgpigp

Here, we evaluate the improvement achieved by combining traceroute data with IGP and BGP messages from AS-X. Figure 10 compares the sensitivity and specificity of ND-edge and ND-bgpigp for three link failures (results for one and two failures are similar). The main observation is that the use of control plane information improves the specificity of the algorithm. This is because, as described in Section 3, the use of BGP information helps ND-bgpigp to eliminate some non-failed links from its hypothesis set. Though the improvement does not appear to be large, these results are from simulations on a relatively small topology. If these simulations were at the scale of the real Internet, the benefit of using BGP and IGP information would be greater.

The performance improvement of ND-bgpigp over ND-edge depends on the location of the failures with respect to AS-X. If all failed links lie inside AS-X, then the use of IGP information ensures that ND-bgpigp can *always* find the exact set of failed links This is a significant performance improvement over ND-edge, specially because it is more important to exactly identify the failed links if they are inside AS-X. If the failed links lie outside AS-X, then ND-bgpigp can only use BGP withdrawals received at AS-X to reduce the size of the hypothesis set. This leads to an improvement in the specificity, as shown in Figure 10. The improvement, however, is less than when the link failure is within AS-X.

We also study whether the performance of ND-bgpigp depends on the position of AS-X in the Internet topology (i.e., whether AS-X is a core or stub AS). Due to space constraints we summarize our main findings without presenting quantitative results. We find that the position of AS-X makes no difference to the sensitivity of ND-bgpigp. However, the specificity is either the same or higher when AS-X is at the core, compared to when it is a stub. This is explained as follows. On receiving a BGP withdrawal for a prefix corresponding to destination s_d, AS-X removes from the hypothesis set any upstream links on the path to s_d, improving the specificity of the hypothesis. When AS-X is a core AS, it is more likely to be in the middle of several paths, and the likelihood of being able to remove upstream links is higher. As a result, the use of BGP information helps a core AS more often than an edge AS.

5.4 Blocked traceroutes

We conclude our evaluation with a study of the impact of ASes that block traceroutes. We compare ND-LG with ND-bgpigp, which simply ignores any unidentified link in traceroute paths. When the failed links are in an AS that blocks traceroutes, it is impossible for ND-bgpigp to identify the exact links that failed. Thus, we focus on identifying the AS(es) with failed links. Let f_b be the fraction of ASes that block traceroutes.

We initially assume that each AS provides access to its Looking Glass server, and compare the performance of the algorithms for values of f_b from 0 to 0.8. For each value of f_b, we compute the average AS-sensitivity and AS-specificity across 1000 simulation runs. The AS-sensitivity and AS-specificity are calculated using the ASes covered by the probes, which consists of 15 ASes on average. We present the results in Figure 11. The average AS-sensitivity and AS-specificity of ND-LG are both around 0.8, even when 80% of the ASes on the paths block traceroutes. This means that for single link failures, the algorithm returns the failed AS 80% of the times, with an average of two false positive ASes. ND-bgpigp has a much lower sensitivity under the same settings, and the value is close to 1-f_b. Intuitively, this is because ND-bgpigp cannot diagnose a failure in an AS that blocks traceroutes, which happens

with a probability around f_b.

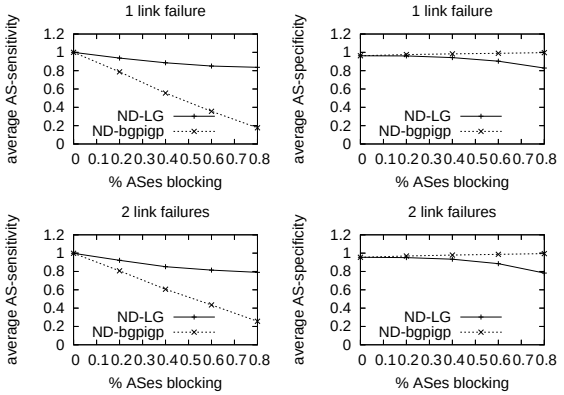

Figure 11: The effect of blocked traceroutes

We also evaluate the performance of ND-LG when some ASes do not have Looking Glass servers. Here, we use f_b=0.25, 0.5 and 0.75, and vary the fraction of ASes that allow the use of Looking Glass servers from 5% to 100%. We calculate the average AS-sensitivity and AS-specificity across 1000 simulation runs. Figure 12 shows the AS-sensitivity as a function of the fraction of ASes providing Looking Glass servers. We also show the sensitivity of ND-bgpigp, which does not depend on the fraction of ASes that allow Looking Glass servers, resulting in the horizontal lines. There is a significant gain over ND-bgpigp, even when a small fraction of ASes allow the use of their LGs. This benefit increases quickly as more ASes allow LGs, and after about 50% of ASes provide LGs we see diminishing returns.

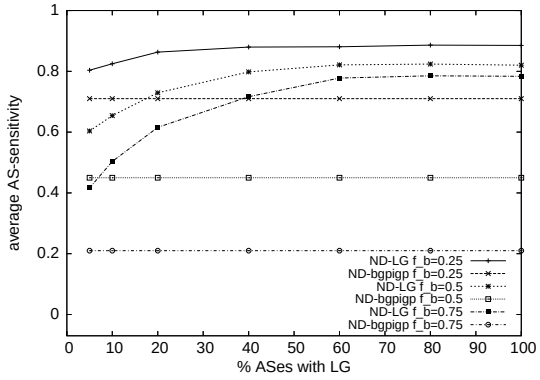

Figure 12: The effect of Looking Glass servers

6. DISCUSSION

A deployment of NetDiagnoser requires several practical issues to be resolved. The first issue relates to the *detection* of an unreachability problem by the sensors. Events such as link flaps could affect the measurements, causing transient events to be treated as failures. This can be overcome by using a more robust detection algorithm. For example, the troubleshooter could raise an alarm only if the failure manifests itself in several successive measurements. So far, we have assumed that the sensors perform measurements at approximately the same time, which requires some form of clock synchronization. Approximate clock synchronization can be achieved in practice using NTP. Also, we assume that the time required to perform a measurement is small compared to the time between measurements, and that the network state does not change during the measurement. The measurement duration can be made small (on the order of a few round-trips) using a modified form of traceroute that probes each hop of the path *in parallel*. The probability of a network event occuring during this small measurement period is then negligibly small. In [5], we examine further issues related to a practical deployment of NetDiagnoser, and present some ideas.

7. RELATED WORK

The Tomo algorithm is an instance of network tomography. There are a number of tomography solutions to detect network-internal characteristics such as topology [4], or link delay and loss rate [3, 8]. These solutions do not apply here because our goal is not to determine the properties of every link, but rather diagnose a reachability failure observed by end-hosts.

The area of Boolean tomography is most similar to our work. Duffield [7, 6] presents a simple algorithm to detect the smallest set of failed links that explains a set of end-to-end observations on a tree topology. Steinder et al. [27] use a "belief network" to find the links that have the highest likelihood of being faulty. A similar approach is adopted to find Shared Risk Link Groups [16]. Kompella et al. [17] present an algorithm similar to Tomo for failure localization using measurements between edge routers in a network. All these algorithms assume that the topology is completely and accurately known, and hence apply mostly to intradomain scenarios. Another approach to failure identification uses Bayesian techniques [15, 28, 24, 14, 22]. These studies assume that the link failure probabilities are known. The state of the art in this area [23], introduces a technique to learn the link failure probabilities from multiple measurements over time. In our work, we make no assumption about the probability of link failures, and only assume that the smallest set of potentially failed links is most likely to explain the observations. None of the tomography studies considers the multi-AS environment or uses control-plane information in the diagnosis.

Several measurement studies correlate end-to-end performance degradation with control-plane events. Feamster et al. [9] measure the location of path failures, their durations, and their correlation with routing protocol messages. More recently, Wang et al. [29] studied the causal effects between routing failures and end-to-

end delays and loss rates, and the impact of topology, policy and BGP configuration on these effects. These studies, however, do not present algorithms to identify the location of those failures.

Tulip [18] and PlanetSeer [30] are two systems for Internet path diagnosis. Tulip uses end-to-end probes to track Internet path performance. Its approach is different from ours because the goal there is to identify the location of the "bad" links with respect to loss rate, queuing, or reordering. PlanetSeer's goal is to detect paths with bad performance, and not to identify the location of the path failure. In fact, PlanetSeer could be used as the sensor infrastructure for NetDiagnoser.

In the area of interdomain routing root-cause analysis, Feldmann et al. [11] use passive BGP measurements to find the root-cause of BGP-visible routing changes. However, this approach can only diagnose path failures that are visible at some BGP collection points.

8. CONCLUSIONS

Troubleshooting network unreachability problems is a challenging task, especially when multiple ASes are involved. In this paper, we take some first steps towards addressing this problem by proposing NetDiagnoser, an algorithm for multi-AS network troubleshooting. We use as our starting point the Boolean tomography approach introduced by Duffield [7], and significantly extend it to work in a multi-AS scenario. In particular, we introduce techniques to make use of the information obtained from rerouted paths, BGP and IGP messages, and Looking Glass servers. NetDiagnoser can also locate failures due to router misconfigurations. We find that NetDiagnoser performs very well in a variety of failure scenarios, such as link failures, router failures and router misconfigurations. The hypothesis set almost always contains the actually failed links, with a small number of false positives. In cases where some ASes block traceroutes, NetDiagnoser is able to identify the AS(es) responsible for the failed links, again with a small number of false positives and false negatives.

9. REFERENCES

[1] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *Proc. Internet Measurement Conference*, 2006.

[2] C. Boutremans, G. Iannacconne, and C. Diot. Impact of Link Failures on VoIP Performance. In *Proc. of NOSSDAV workshop*, 2002.

[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] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang. Maximum Likelihood Network Topology Identification from Edge-based Unicast Measurements. In *Proc. ACM SIGMETRICS*, 2002.

[5] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot. NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data. Thomson technical report CR-PRL-2007-02-0002 `www.cc.gatech.edu/~amogh/CR-PRL-2007-02-0002.pdf`.

[6] N. Duffield. Simple network performance tomography. In *Proc. Internet Measurement Conference*, 2003.

[7] N. Duffield. Network Tomography of Binary Network Performance Characteristics. *IEEE Trans. Information Theory*, 52, 2006.

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

[9] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek. Measuring the Effects of Internet Path Faults on Reactive Routing. In *Proc. ACM SIGMETRICS*, 2003.

[10] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proc. USENIX NSDI*, 2005.

[11] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs. Locating Internet Routing Instabilities. In *Proc. ACM SIGCOMM*, 2004.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[13] D. Johnson. Approximation Algorithms For Combinatorial Problems. In *Jnl. of Comp. and System Sciences*, 1974.

[14] 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)*, 2005.

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

[16] R. Kompella, J. Yates, A. Greenberg, and A. Snoeren. IP Fault Localization via Risk Modeling. In *Proc. USENIX NSDI*, 2005.

[17] R. Kompella, J. Yates, A. Greenberg, and A. Snoeren. Detection and Localization of Network Black Holes. In *Proc. IEEE INFOCOM*, 2007.

[18] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet Path Diagnosis. In *Proc. ACM SOSP*, 2003.

[19] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz. Towards an accurate AS-level traceroute tool. In *Proc. ACM SIGCOMM*, 2003.

[20] W. Muhlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig. Building an AS-topology Model that Captures Route Diversity. *Proc. ACM SIGCOMM*, 2006.

[21] NANOG. Looking Glass Sites. `http://www.nanog.org/lookingglass.html`.

[22] H. Nguyen and P. Thiran. Using End-to-End Data to Infer Lossy Links in Sensor Networks. In *Proc. IEEE INFOCOM*, 2006.

[23] H. Nguyen and P. Thiran. The Boolean Solution to the Congested IP Link Location Problem: Theory and Practice. In *Proc. IEEE INFOCOM*, 2007.

[24] V. Padmanabhan, L. Qiu, and H. Wang. Server-based Inference of Internet Link lossiness. In *Proc. IEEE INFOCOM*, 2003.

[25] B. Quoitin and S. Uhlig. Modeling the Routing of an Autonomous System with CBGP. In *IEEE Network Magazine*, 2005.

[26] Routeviews. University of Oregon Route Views Project. `http://www.routeviews.org`.

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

[28] C. Wang and M. Schwartz. Fault Detection With Multiple Observers. *IEEE/ACM Trans. Networking*, 1(1), 1993.

[29] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush. A Measurement Study on the Impact of Routing Events on End-to-end Internet Path Performance. *Proc. ACM SIGCOMM*, 2006.

[30] 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*, 2004.