Diagnosing Internet Failures Using End-to-End Measurements and Routing Data

Ovidiu Mara

NAL, I&C, EPFL

Abstract—The scale and the distributed nature of the Internet make it difficult for Internet Service Providers and end-users alike to identify the causes of failures that affect their networking services. Among the variety of problems that can occur, failures on the IP forwarding path are the hardest to troubleshoot. Under these circumstances, a number of network failure diagnosis techniques have emerged over the last decade, having the goal of detecting link or router level failures indirectly, using end-to-end measurements. Other available data, such as routing information, may also be used to improve some of these methods.

Index Terms—network failure diagnosis, network tomography, performance, inference

I. INTRODUCTION

This paper presents three diagnosis techniques that can be used to identify failures on the IP forwarding path in a computer network. Such failures are generally hard to troubleshoot, because they might occur on devices for which the affected users have no administrative rights, and thus no possibility to debug or test directly. In this situation, diagnosis techniques can only rely on the limited debugging functionality offered by the current Internet infrastructure (traceroute, ping), on end-to-end probing between a small number of hosts to which they have remote access to, and on a creative use of other available information, such as BGP updates.

We will first present network performance tomography, a technique for inferring performance characteristics of the devices within a network using end-to-end measurements. It is used to infer the quality of physical links, such as loss rate, delay, or —in the case of binary or “Boolean” tomography [1], [2]— to simply decide whether a link is congested or not. Section II shows a simple algorithm that applies binary tomography on tree network topologies.

Section III shows how binary tomography can be adapted to realistic network topologies: multi-AS environments. We present the algorithm “Tomo” from [3], which applies binary tomography on graph topologies. We then show how it was enhanced for a realistic setting, becoming a diagnosing system called “NetDiagnoser”.

Section IV is focused on Hubble [4], a system that operates continuously to find Internet black holes —failures that cause complete and permanent unreachability on the paths towards routable destinations. Black holes are an important problem because they violate the central property of the Internet: global reachability, the ability to send traffic from every address to every other address. We discuss the main qualities of Hubble, and how it is different from network tomography.

Finally, section V shows the current research direction of my thesis, with respect to the background presented in this paper.

II. BINARY NETWORK TOMOGRAPHY ON TREE TOPOLOGIES

Network performance tomography is a technique for inferring performance characteristics of the devices within a network using end-to-end measurements. It is used to infer the quality of physical links, such as loss rate, delay, or — in the case of binary or “Boolean” tomography — to simply decide whether a link is congested or not. In this section, we present a simple algorithm that applies binary tomography on tree network topologies, called “Smallest Consistent Failure Set” [1].

The scenario used is the following. The network is represented as a directed tree, consisting of a set of nodes connected by links. The root of the tree and the leaves are hosts; all the other nodes are routers. Also, the root node is connected to the rest of the network with exactly one link (i.e. it has one child). The root node acts as a server, running an application that
EDIC RESEARCH PROPOSAL

sends continuously traffic to the leaves, while each leaf node represents a client, running another application that receives and records the traffic sent by the server. We define as path a contiguous, ordered sequence of links between the root node and a destination node.

During an experiment, the server sends unicast traffic to all the destinations. As a result, some of the links might experience performance degradations, such as packet loss. We call links that show such problems “failed”, while the links that perform well are “good.” For every source-destination pair, the performance degradation over the path depends on the performance of each link along the path. We define a binary metric that describes the quality of each link; we encode with “1” the fact that the link is good, and with “0” that it failed (e.g. if its loss rate exceeds a certain threshold). Similarly, we define a binary metric that describes the quality of the paths.

The system can measure the performance of the paths by keeping track of the packets sent by the server and what was received by the clients. Packet counters can be used to measure the loss rate on each path. The path qualities are, thus, known. The unknowns are the link qualities, and the goal of the tomography algorithm is to infer them.

### A. The SCFS Algorithm

The algorithm we are presenting is called “Smallest Consistent Failure Set.” Its design is based on three assumptions: (i) that links are fair for the flows that traverse them, (ii) that failure events are rare, and (iii) that failures are independent and equally likely to occur on any link in the network.

Link fairness means that every flow that traverses a link experiences the same performance degradation, or lack thereof. Therefore, if a link is experiencing performance problems, all of the flows that traverse the link will be affected equally, and thus all the paths that traverse the link will consistently show problems. This allows us to use end-to-end measurements in order to make two type of deductions. First, if a path is good, we can safely conclude that all the links traversed by that path are good. Second, if several paths that share common links have failed, we could correlate the information in order to determine which of the links have most likely failed, if we had some extra information that helped us determine which combination of failed links had the highest probability to occur.

The assumptions that (ii) failures are rare events and (iii) they are distributed uniformly help exactly in this regard: under these assumptions, the smallest set of failed links that explains the failed paths is considered the most likely solution.

Hence, the SCFS algorithm works as follows. First, if all the paths in the network have failed, then we conclude that the link that connects the server with the rest of the network has failed, that every other link is good, and the algorithm finishes. If this is not the case, we decide that the link at the root must be good (since there is at least one good path traversing it), and then we apply the algorithm recursively for each of the subtrees rooted at the server’s children, until there are no more failed paths to explain. Consequently, the algorithm will always pick as failed only the links nearest to the root that are consistent with the observed pattern of failed paths, as shown in Figure 1.

![Fig. 1. Operation of the SCFS algorithm. Top left: the topology with the path quality assignments (crosses represent failed paths). Top right: the link qualities computed by SCFS. Link 2 has been picked as failed because it explains three failed paths; as the path failures are explained, links 3, 4 and 5 are marked as good, although their actual status is not known. Link 8 is also picked as failed. Bottom left: a different configuration of failed links consistent with the path qualities; edges 3, 4 and 5 are failed, and edge 2 is good. The SCFS algorithm cannot detect this situation. Bottom right: yet another configuration consistent with the path qualities: links 2 and 4 are both failed. This is also not detectable by SCFS.](image)

### B. Discussion

The SCFS algorithm demonstrates a simple and effective principle behind binary tomography, however it has a few problems.

First, links picked as failed may hide other failed links, that will be denoted as good. For the algorithm, it is impossible to tell when this occurs without extra information. It is thus expected that the algorithm gives a high false negative rate.

Second, assumption (iii), that failures are equally likely to occur on any link in the network, may not hold. Failures at the edge of the network are more likely in practice than failures in the Internet core, due to over-provisioning in the core. Thus failures are indeed rare, but not distributed uniformly. The algorithm would wrongly pick a link closer to the core of the network in case several leaves experienced congestion (Figure 1, bottom left).

Third, the fairness assumption may not hold. Our network emulations showed that, in case of drop-tail link queues without ECN, which is the most common setting in the Internet nowadays [5], [6], flows that send smaller packets reach lower loss rates than flows with full sized packets, in

---

1Assumption (iii) is not specified explicitly in [1].
case of congestion. This has also been suggested as a result of black-box analysis of several models of Cisco routers in their default configuration [5]. Care must be taken in case of both passive and active measurements in order to prevent mixing measurement data obtained from different types of traffic.

Finally, it might seem that with higher tree branching ratios the algorithm would perform better. Counter-intuitively, this is not the case, because the impact of an incorrect decision also increases with the branching ratio, counterbalancing the advantage. In [1] it is also shown that variations of the tree depth have no effect on the performance of the algorithm, as long as the fraction of bad paths is held constant. As a result, the main factor that influences the accuracy of the SCFS algorithm is the validity of the assumptions given above.

III. NETWORK TOMOGRAPHY ON GENERAL TOPOLOGIES

In this section we will show how binary tomography can be adapted to realistic network topologies: multi-AS environments. We first present the algorithm “Tomo” from [3], which applies binary tomography on graph topologies. We then show how it can be enhanced for a realistic setting, becoming a diagnosis system called “NetDiagnoser”.

The scenario we use is a multi-AS environment in which a provider has access to an overlay of troubleshooting sensors located at end hosts in multiple ASes. The sensors can send probing traffic to each other, and the goal is to diagnose failures that cause some pairs of sensors to become unreachable. Specifically, we want to identify which links have failed, or if not possible, at least the AS in which the failure originated.

Since the network consists of several ASes, a complete and accurate network topology is not known. However, it is possible to obtain an IP-level topology of the paths between the sensors by using tools like traceroute. For each pair of sensors, the forward and the return path between them are determined using traceroute, and then the entire set of paths must be merged together to form a graph.

The paper [3] incorrectly states that this graph does not require router aliasing, motivating that the set of links returned by traceroute is sufficient, and determining if different network interfaces belong to the same router is not necessary. However, this is not true, because routers send ICMP time exceeded replies using the IP address of the interface on which the reply is returned, and not the interface that would have forwarded the traffic. This fact is clearly stated in RFC1812 [7]. An example where this causes problems is given in Figure 2. It can be seen that if two paths share exactly one common link, but have different prefixes and suffixes, it is impossible to detect the overlap without router aliasing. The problem also manifests when paths intersect over more than one link: although the mapping detects that paths do intersect, the common set of links is always missing a link. Such inconsistencies can impact the accuracy of the tomography algorithm not only by introducing unnecessary extra unknowns, but also through the fact that these unknowns are correlated. It is thus clear that router aliasing should be applied to the topology in order to remove duplicate links, before running the tomography algorithm.

A. The Tomo Algorithm

The main reason why the SCFS algorithm cannot be applied to this scenario is not related to its working principle, but to the implementation. SCFS was designed exclusively for tree topologies, and it cannot work on graphs. Therefore we need a new algorithm, based on the same principle as SCFS, which can be used for graphs. This algorithm, described in [3], is called “Tomo.”

Tomo works under exactly the same assumptions as SCFS: that (i) links are fair for the flows that traverse them, that (ii) failure events are rare, and that (iii) failures are independent and equally likely to occur on any link in the network. The working principle is also identical: under these assumptions, the smallest set of failed links that explains the failed paths is considered the most likely solution.

Before showing how Tomo works, we need to give a formal definition of the problem. The topology graph $G = (V, E)$ is the directed graph obtained by uniting all the paths identified through traceroutes between sensors. A subset of the links in the graph may fail, which in turn, under the fairness assumption, causes the paths that traverse these links to fail. As with SCFS, we associate with each link and each path a variable that represents its status. The status is encoded with 1 for “good”, and 0 for “failed”.

Let $y$ be a vector encoding the status of each path in the graph. Then $y_i = 1$ if path $p_i$ is good, and $y_i = 0$ otherwise. Similarly, let $x$ be a vector that denotes the status of the links, with $x_i = 1$ if link $l_i$ is good, and $x_i = 0$ otherwise.

Under the fairness assumption, if $y_i = 1$, then $x_j = 1$ for each link $l_j$ in path $p_i$. Conversely, for each failed path $p_i$, one or more of the links $l_i$ in $p_i$ must have failed. Thus we consider the links on failed paths as candidates of being failed links, and we denote for each failed path $p_i$, the set $\cup_{l \in p_i, p_i\text{ failed}} = L_i$ as the candidate failure set$^3$ of the path. Let $\mathcal{L}$ be the set of all the candidate failure sets, and let $U = \cup_{l \in p_i, p_i\text{ good}} = G$, the set of candidate failed links. We denote with $\cup_{l \in p_i, p_i\text{ good}} = G$, the set of links known to be good.

$^3$This was called in [3] a “failure set”, however we believe that the name “candidate failure set” is more appropriate, as the links in the set are only candidate failed links.
The binary tomography problem is defined as follows: find the smallest set of failed links \( H \), which is consistent with the observations \( y \). This means that \( H \) is the smallest set of links that intersects with each candidate failure set \( L_i \), i.e. \( H \cap L_i \neq \emptyset, \forall L_i \in L \). Also, \( H \) must not contain any link that is known to be good, i.e. \( H \cap G = \emptyset \).

This problem is known as the Minimum Hitting Set, an NP-hard problem [8]. For practical reasons, we will not solve the problem optimally, and instead will use a greedy heuristic that approximates the solution within an approximation ratio of \( \log |U| \) [9]. This approximation algorithm will be called “Tomo.”

The algorithm works as follows. We first remove from the set of candidate failed links \( U \) and from all the candidate failure sets in \( L \) all the good links, \( G \). Then, we maintain a set of unexplained failure sets, which are sets from \( L \) that are not explained by the current partial solution \( H \) (initially empty). From this point, we proceed iteratively, at each step picking from the set of candidate failed links \( U \) the links that appear in the highest number of unexplained failure sets. This means that these links appear in more failed paths than any other link, and thus are considered the source of these path failures, so we add them to the solution \( H \). The failure of each path traversing one or more of these links is considered as explained, therefore all the other links belonging the paths are removed from \( U \), and the candidate failure sets corresponding to the explained paths are removed from the set of unexplained failure sets. The algorithm stops when all the path failures have been explained.

**B. Practical Limitations**

There are several problems that affect the effectiveness of Tomo in practice.

Link unfairness can occur due to incorrect BGP policies, packet filtering, and to some extent due to queuing. Unfairness can cause a link to behave as failed with respect to some paths, and as good for others. Because of the strict reasoning used in the Tomo algorithm, its accuracy can be affected by such anomalies.

An enhanced version of the Tomo algorithm is presented in [3]. This version relaxes the assumptions required by Tomo, by allowing link unfairness to occur on inter-AS links, thus taking into account the possibility of unfairness caused by BGP misconfigurations or by filtering at the domain edges.

The improved method works in two stages. First, all the routers in the topology are annotated with AS information, based on their IP address, using the technique from [10]. This is done to allow the identification of inter-AS links. Then, each inter-AS link is split into multiple parallel links, one for each neighbor AS towards which the AS of the link destination endpoint is forwarding traffic. After splitting, we have a different link on each AS path; due to the fact that BGP policies are generally specified on a neighbor AS basis [11], this should be sufficient in order to guarantee fairness in the modified topology, albeit at the expense of introducing extra unknowns. An example is shown in Figure 3.

The result of the transformation is a new topology, represented as a multigraph. The original version of the Tomo algorithm can be applied on this graph in order to identify the failed links.

Rerouting can also be problematic, because Tomo works on a fixed topology, which does not change when failures appear. However, real networks are much more dynamic, and usually try to recover automatically from reachability problems by attempting to compute new routes around the failed links. Tomo should be enhanced in order to not only take into consideration topology changes, but also by taking advantage of this additional information in order to provide more accurate results.

If a path fails and is detected as such, but after a short time the network manages to reroute and find a new working path around the failure, then all the links on the new path must be working. The set of working links \( G \) in Tomo should contain these links and not the old ones. We can assume that some of the old links—that appear in the path before rerouting, but not afterwards—have probably failed, so we could create additional candidate failure sets containing such links.

In addition to performing end-to-end measurements continu-
uously, the diagnosis system would also have to retrace the paths every time there is a reason to suspect that the routes have changed. An enhanced version of the Tomo algorithm would work in rounds, performing periodically both the end-to-end measurements and topology mappings using traceroute. Together with the improvement that takes into account link unfairness, this forms the solution bearing the name ND-edge in [3].

Incomplete topology mapping is another common problem in practice, caused by routers that are sometimes configured to ignore or block traceroute, or instead respond with private IP addresses. In both cases, parts of the paths are missing, and the topology is incomplete. We will call the hops that could not be identified unidentified hops, and the links that have at least one endpoint in such hops unidentified links.

Unidentified links affect the diagnosis in two ways. First, if a link failure occurs on one of these links, it is unlikely that Tomo will identify the failed link correctly. The reason is that the algorithm is looking for links that are part of many failed paths, but there is no way for the algorithm to know that an unidentified link that appears in multiple paths is actually the same one. From the perspective of the observer, each unidentified link appears in only one path, and thus will almost never be picked as failed.

Second, due to the fact that the IP addresses of unidentified hops are not known, it is not clear to which AS they belong, hindering the enhancements offered by ND-edge. An enhancement to Tomo is proposed in [3], that makes use of Looking Glass servers [12] in order to attempt to (1) identify the AS in which unidentified hops are located and (2) match unidentified links that are instances of the same link. The technique is called ND-LG and works in the following way. For each path with unidentified hops, the system searches for an identified hop that precedes the unidentified ones, and resides in an AS that offers a Looking Glass server. If such a hop is found, the Looking Glass server is queried in order to obtain the AS path to the destination of the path. By correlating the AS path with the AS information of the identified hops on the path, the unidentified hops can be tagged with the ASes they might belong to. It would be ideal to identify precisely in which AS a hop is located, but in case of ambiguities, a hop may be tagged with a set of several possible ASes.

After tagging the unidentified hops with AS information, we proceed to attempting to match duplicate instances of unidentified links. Unidentified links from different paths are considered identical if their endpoints have identical tags, and if they appear in the same number of failed paths. Although this matching is not precise, it is now possible to diagnose failures that occur on unidentified links using Tomo. Even if the exact failed link cannot be located, there is a chance of identifying the AS that caused the failure.

C. Discussion

The Tomo algorithm and its enhanced versions have been evaluated in the original article [3] using simulations over a relatively large topology with 165 ASes and over 200 links. However, the fraction of failed links was tiny — only one, two or three failed links were used. The quality of the results proved to be highly dependent on the sensor placement: it was found that random placements may not give very satisfactory results, while the best placements were those with large numbers of sensors located in a few ASes, for instance placing half of the sensors in an AS and the other half in another.

Obtaining a favorable sensor distribution would be difficult and expensive, if one were to deploy this tomography technique in today’s Internet. By contrast, a random placement, where a few cooperating ISPs and/or end users are maintaining a small number of probing hosts, seems more realistic. For this reason, the authors have decided to focus on evaluating the performance of the algorithms using random sensor placements.

Tomo has proven useful in locating only one link failure, while in experiments with more failed links it performed very poorly, even in the absence of unfairness. Its enhanced version, ND-edge, was able to give satisfactory results, because it was taking advantage of the information obtained from rerouting.

In our own experiments, we evaluated Tomo (not the enhanced versions) using network simulation and emulation. The results have been consistent with the above.

We have seen that one of the most important problems of Tomo is the propagation of errors: any good link that is identified incorrectly as failed leads to marking the truly failed link as good. Any other paths that might traverse the latter would lead to more errors, because in turn other good links might be marked as failed instead, to explain the path failures. In other words, small errors trigger other errors, which in turn cause even more errors. It is important to remember that it is impossible to solve this problem with perfect accuracy in any practical scenario, due to the fact that the number of unknowns (links) exceeds the number of constraints (paths). The algorithm is thus inherently error prone, in which case being susceptible to error propagation is a significant drawback.

Under this consideration, instead of computing just the likeliest solution, we believe it might be better to attempt to compute a set of several solutions that have a high probability of explaining the failures. Another approach could be to not opt for exact solutions at all, and instead compute a fuzzy view of the network, like a weather map, that would point out the most likely segments which could have caused the failures, as done in [13], [14].

Another issue we would like to point out is that in ND-edge, unfairness is dealt with by artificially increasing the number of unknowns. This is a solution that the authors themselves pointed out that it scales poorly, and it can also be applied only on a limited number of links. If we assumed that unfairness could occur on any link in the network, and therefore attempted to split all the links, the problem would become unsolvable, because every link would be part of no more than one failed path. This would make it impossible to decide which of the candidate failed links are indeed failed, and which are good.

Rather than working around unfairness, we propose that it might be preferable to instead study its nature in practice, find out to what extent it usually occurs, and use this information
to build a diagnosis system that is robust to a certain amount or type of unfairness.

IV. BLACK HOLE DETECTION

This section presents Hubble [4], a system that operates continuously to find Internet black holes — failures that cause complete and permanent unreachability on the paths towards routable destinations. Black holes are an important problem because they violate the central property of the Internet: global reachability, the ability to send traffic from every address to every other address [15]. These outages have the following specific properties:

- **Not caused by routing failures.** Black holes affect the traffic towards destination prefixes that are advertised via BGP; reachability problems where the prefix has been withdrawn from BGP tables or has never been routable are not considered black holes.
- **Persistence.** Short term, transient packet losses caused by routing changes or other short-lived events do not qualify as black holes. Only persistent failures, for instance lasting more than half an hour, are considered.
- **Affect the forwarding infrastructure of the Internet.** Lack of reachability caused by end hosts, or problems in the local network of end hosts are not considered black holes; only failures that appear in the core of the Internet qualify as such.

A concrete, practical criterion to determine that a certain end host is unreachable is to run traceroutes towards it from various points around the Internet and observe that more than 10% of them fail to reach it. It is pointed out in [4] that if this is the case, it is very likely that the problem is much more serious. For example, in half of the cases when a destination qualified as unreachable, it could not be reached from more than 50% of the vantage points that were probing it.

In order to identify black holes, Hubble uses several sub-systems connected together, each having a specific purpose:

- **Pingable address discovery.** This subsystem extracts all the routable prefixes in the Internet from BGP feeds, and selects a set of targets that can be probed using ping.
- **Identification of unreachable targets.** To identify prefixes that might be unreachable, the system uses two sources of information: (i) failure to ping a destination that was previously pingable and (ii) BGP routing changes which could be caused by failures. The destinations selected in this stage are considered potential targets, and are then fed to the other subsystems of Hubble, to be diagnosed using traceroute and other methods.
- **Reachability analysis.** Targets that were selected as potentially having problems are probed using traceroute from several vantage points.
- **Problem classification.** Information from the traceroutes, combined with extra data, such as from an Internet topology updated daily, router liveness monitoring and from specially crafted spoofed probes are used in order to identify and classify the problem.

The above subsystems of Hubble are presented in the following subsections.

A. Address Discovery and Identification of Targets

The pingable address discovery consists of selecting each .1 address from every /24 prefix present in a BGP snapshot obtained from RouteViews [16]. These addresses are served to the ping monitoring system.

Ping probes are sent by Hubble from various vantage points on the Internet, towards each destination address. Care is taken to prevent flooding the targets; overall, the system never pings an address more often than once in two minutes.

The goal of ping monitoring is to discover previously reachable prefixes that stop sending replies, which might signal a failure. If an address does not reply 6 times in a row, it is considered as potentially unreachable, and is reported to the traceroute probing system for diagnosis. The detection threshold has been chosen empirically, in order to eliminate the possibility of a transient problem.

In addition to active monitoring through ping, Hubble also uses passive BGP monitoring to identify potential problems. BGP updates are received from more than 40 locations every 15 minutes, and suspect changes are used to supplement the ping monitoring by identifying additional candidate prefixes. BGP changes can be either a symptom of the problem, such as when routers try to find alternative paths to a destination that is no longer reachable, or the cause, as in the case of misconfigured BGP policies that lead to outages.

The triggered traceroutes are sent to the candidate targets every 15 minutes, from a few PlanetLab hosts around the Internet. Not all the vantage points send traceroutes to a destination at the same time, to prevent flooding the target. The result of the traceroutes is reported to the analysis system.

B. Traceroute Analysis

The result of the traceroutes has a dual purpose: to decide whether a destination is indeed unreachable, and to analyze and identify the cause of the failure.

It is common for traceroutes not to reach their targets even if they are not suffering from any problems, for a variety of reasons: the end host may be down, a firewall might be blocking either the probes or ICMP timeout replies etc. In all these cases, the destination prefix might be very well reachable for other destination addresses or other types of traffic. In this situation, a robust test must be used to determine if a target prefix is unreachable. A method that is simple and works generally well is to check whether the traceroute has reached any address from the target prefix. The target is considered reachable if and only if it passes this test.

After identifying an unreachability problem, the only thing that is known is the address prefix experiencing it. The next step in the diagnosis is to identify the location and cause of the problem, and report the result to an operator. To be useful, such a report must contain at least the AS where the problem is located, optionally also the IP addresses of the routers involved, and the paths for which the problem manifests.

C. Problem Classification

To analyze and explain a failure event, Hubble requires the working topology before the the failure appeared; this
is necessary because a complete post-mortem analysis using traceroute is not possible due to the failure itself. For this purpose, a map of the entire Internet, as seen from the Hubble vantage points, is generated every day. The mapping is obtained using a combination of techniques: traceroute, ping with record route and router interface aliasing [17], [18].

A simple classification scheme is used to diagnose the problem, based on the last observable hop, the expected next hop and the ASes of these hops in the traceroutes. The most common sources of failure in the Internet are broken into 9 different classes, each characterized by a specific pattern of which traceroutes fail to reach the destination and which ones succeed. These classes are based on hand analysis performed by the Hubble authors, and are used in an automatic fashion by the Hubble system to diagnose failures.

The following criteria have been used to define the classes of failures:

- Is the problem local to the origin AS, or does it occur in a third party? In case of the latter, is it located in a direct provider or further upstream?
- Is the problem caused by an entire AS that does not forward traffic, or just a few isolated routers?
- If the problem was caused by a single router: did the router fail completely, or did it just stop forwarding the traffic to our target? Is it a router that previously worked, or a new router used the first time after a path change?
- Is the unreachability complete, or are there paths that can still reach the destination?

Various combinations of the above properties lead to the 9 classes of unreachabilities:

**Problem local to the origin AS:**
1. The destination AS is reachable, however the destination prefix is not. The problem lies in the destination AS; this is not a very interesting case.

**Problem caused by another AS, with complete failure:**
2. The destination AS is single-homed, and its provider stopped forwarding traffic to it.
3. The destination AS is multi-homed, and all its providers stopped forwarding traffic to it.
4. An upstream provider stopped forwarding traffic.

**Problem caused by another AS, with partial failure:**
5. The destination AS is multi-homed, and some of its providers stopped forwarding traffic to it.
6. The destination AS is multi-homed, and some of the upstream providers stopped forwarding traffic to it.

**Problem caused by a router:**
7. A router that previously worked stopped forwarding traffic to the destination. The router is still up.
8. A router that previously worked is no longer reachable (probably down).
9. A path change occurred, and a new router does not forward traffic to the destination.

**D. Discussion**

Although both Hubble and the network tomography techniques presented in sections II and III are used for diagnosing network failures, the two exhibit striking differences.

Most importantly, Hubble uses a very simple diagnosis approach in comparison to the more complex algorithms used for tomography. The main difference is that while tomography uses tools like traceroute for topology mapping, and then enters a new stage in which end-to-end measurements are used in combination with an inference algorithm, Hubble lacks this second stage completely, performing both the mapping and the diagnosis using traceroute. This is only possible because the two techniques actually target different types of failures: Hubble diagnoses persistent, complete drops of traffic, while tomography targets “softer” failures, such as congestion. Because with soft failures, some of the traceroute traffic might still be forwarded successfully, the approach in which Hubble probes with traceroute cannot be applied in tomography.

It can be said that in contrast with tomography, Hubble is at an advantage, as the problem is easier to solve, using readily available tools. However, the system deserves considerable credit for integrating these tools in an ingenious, effective and scalable manner. The degree of automation of Hubble is also impressive.

One of its best accomplishments is that it has been used to monitor a large fraction of the Internet, namely targets from more than 110,000 prefixes that have pingable addresses. These account for 85% of the edge prefixes in the Internet, thus we can safely state that almost the entire Internet is covered. It is surprising that over a three week period, more than 10,000 prefixes experienced one or several unreachability events, totaling to over 30,000 events. This means that around 10% of the monitored prefixes were affected by such problems. Another surprising fact is that the duration of the events is relatively long, with a median of 2.75 hours for partial unreachabilities and 3.5 hours for complete unreachabilities, while 10% of the events lasted more than 24 hours.

**V. Conclusion and Research Proposal**

Diagnosing link failures is an ill-posed problem, because the number of unknowns (the status of each link in the network) is always much higher in practice than the number of constraints, or equations (the status of each path in the network). End-to-end measurements alone are not sufficient for solving these problems.

Two approaches that address this issue have been presented. On one hand, Hubble offered a practical solution, by probing not only end hosts, but also the routers between them. Although the solution is highly effective, it can only be used for a very specific kind of failure: complete network partition. There are other types of problems that can affect the performance of network links without cutting off traffic completely, most common being packet loss or large queuing delays. These problems can affect networking applications by reducing or varying throughput, or by increasing latency, and it would thus be desirable to be able to use methods for detecting and diagnosing them. However, probing routers does not work in this case, for several reasons. For example, measuring packet loss on the path to a router requires sending a large number of probes, such as ping requests or low TTL packets, but many ISPs limit or even filter completely both the traffic that targets
their routers and the replies. Unfortunately, such an approach cannot be used reliably for network tomography.

The other tools that we presented attempt to work around the problem by providing extra information to the inference algorithm. The data falls into two main classes: (1) apriori knowledge of some characteristics of the network topology and the failure scenario and (2) information extracted from other sources, such as BGP tables, routing updates etc. The latter is usually scarce, and not primarily planned to be used in this kind of diagnosis, so its usefulness is limited. The former however has a great potential, because it has been shown that computer networks tend to share common features, for instance planarity [19]. It is thus likely that other characteristics, particularly the mechanisms through which failures occur, follow similar patterns in different networks.

This is also suggested by the results obtained by Hubble and presented in Section IV, which showed that 82% of the failures were following a small number of patterns.

It is essential that such assumptions made about the network are in agreement with reality. If they are not valid, the algorithms fail to diagnose the problem properly. For this reason, there has been recent work to relax the assumptions used in network tomography, such as the work in [20] and [13].

Despite these advancements, we believe that there are still several key questions that need to be addressed:

**Do congestion events and soft failures also follow specific patterns in the Internet? What are the most common sources of these failures? Can we use this information to obtain better diagnosis techniques?** We have seen that many works make the assumption that failures are rare and are equally likely to occur anywhere in the network. This contradicts the intuition that failures at the edge of the network are more likely in practice than failures in the Internet core, due to over-provisioning in the core. In this case failures would not be distributed uniformly and, as we have seen in section II, the algorithm would wrongly pick a link closer to the core of the network in case several paths intersecting there experienced congestion.

**How unfair are the links in real networks? What is causing unfairness, and how much does it affect tomography? Can we design algorithms that are robust when facing certain amounts of unfairness?** Most tomography algorithms are in essence solving a system of equations. Unfairness can cause inconsistencies between these equations, and as a result it is a source of errors. We have pointed out that through propagation of errors, the accuracy of an algorithm can be sometimes significantly affected. It would be desirable to find out if it is possible to detect these errors before they propagate, and design algorithms that minimize their effects.

**In addition to computing the solution of network tomography problems, is it possible to also compute automatically an estimate of how reliable this solution is?** Would it be possible to detect that certain assumptions made by the algorithm were likely invalid, and would it be feasible to design an algorithm that adapts its operation depending on observed characteristics of the network? The answer to this question depends on what is found for the previous two questions. Ideally, some specific properties of failure events could be identified, which would allow us to have a better understanding of how the performance of the algorithms varies in various scenarios. This result could be used to predict how accurate an algorithm is for a specific data set. For example, if we can show that unfairness, quantified with a certain metric, does not exceed in practice a certain threshold, and by having a model of how an algorithm is affected by unfairness, we might be able to compute bounds for the accuracy of this algorithm.

**REFERENCES**


