LIFEGUARD: Locating Internet Failure Events and Generating Usable Alternate Routes Dynamically*

Colin Scott

A senior thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Science
With College Honors

Computer Science & Engineering
University of Washington

Abstract

The Internet is far from achieving the "five 9s" (99.999%) of availability provided by the public telephone network. Based on anecdotal evidence as well as a first-of-its-kind measurement study assessing routing failures affecting Amazon's EC2 data centers, we find that network outages are surprisingly common and highly problematic. These outages incur large costs, disrupt critical services and applications, and are generally difficult to detect and isolate.

In this thesis we argue that it is possible to isolate wide-area network faults at a router-level granularity without requiring control over both end-points. We present *LIFEGUARD*, our prototype system for Locating Internet Failure Events and Generating Usable Alternate Routes Dynamically. *LIFEGUARD* leverages distributed network measurements, a continually updated historical path atlas, and novel measurement tools such as spoofed forward traceroute to infer the direction of failures, isolate faults, suggest alternate working paths, and provide views into routing behavior from before the onset of outages. We evaluate *LIFEGUARD* with a four month study monitoring real outages on the Internet, and present specific examples and aggregate characteristics of the failures we observed.

Presentation of work given on June 2nd, 2011

Thesis and presentation approved by:

Date:

* This document presents my original thesis as I submitted it in June, 2011. Other researchers and I subsequently incorporated this work into a larger system, also known as LIFEGUARD (to appear in SIGCOMM 2012). That system uses this thesis work to locate failures, but includes additional mechanisms to automatically route around the discovered failure locations.

Contents

1 Introduction

As the Internet evolves to adapt to new applications such as cloud services, telephony, and financial services, the need for network reliability is steadily increasing. Without working Internet paths, for example, users of cloud-based storage may find themselves completely unable to access their data.

Despite the importance of reliable connectivity, network outages are a common occurrence on the Internet. Outages are time consuming and disruptive, resulting in high operational costs, damaged reputations, and frustrated customers. Wide-area outages are particularly problematic, since a failure in a network outside of a network operator's domain can nonetheless disrupt significant portions of traffic.

At the very least, network operators would like to know which autonomous system is responsible for a failure so that they can contact the accountable party. More ideally, users and operators would benefit from a mechanism for automatically reacting to outages without requiring human intervention.

With the set of techniques and tools available today, even the initial problem of isolating wide-area network faults is difficult if not impossible. In particular, state-of-the-art tools such as `traceroute` provide insufficient information for identifying the failed link between two hosts. As a result, operators have little visibility outside their local network, and may have little idea as to the extent of a problem, its location, or its root cause. Operators often resort to posting to mailing lists when confronted with outages, loss, or degraded performance, asking others to issue traceroutes and otherwise weigh in with their view of the problem [1, 2].

Previous research systems have addressed aspects of this problem in different ways. Architectural approaches propose new protocols or routing schemes intended to prevent or avoid network reachability problems [3]. Deploying new architectures is extremely difficult in practice however, and it is highly doubtful whether a single scheme can prevent all possible types of network reachability problems. Another set of approaches attempts to solve the problem by building on top of the existing Internet infrastructure. Some attempt to detect and isolate reachability problems by passively monitoring live border gateway protocol (BGP) feeds [4, 5, 6]. However, BGP monitoring misses a large portion of network reachability problems, since data-plane anomalies do not always show up in the control-plane [7]. Lastly, monitoring systems [8, 7] use data-plane measurements to monitor outages at Internet-scale, but are largely focused on studying and classifying aggregate properties of outages rather than isolating and circumventing specific problems.

In this thesis we argue that it is possible to isolate wide-area network faults at a router-level granularity without requiring control over both end-points. We present *LIFEGUARD*, our prototype system for Locating Internet Failure Events and Generating Usable Alternate Routes Dynamically.

LIFEGUARD builds on results from previous work, including Hubble [7], reverse traceroute [9], and iPlane [10]. In particular, we employ distributed coordination of network measurements, continually updated historical path atlases, and novel measurement tools such as spoofed forward traceroute.

In addition to isolating the exact router or link causing an outage, *LIFEGUARD* attempts to facilitate remediation by identifying working alternate paths usable by network operators to route around failures. While the design of *LIFEGUARD* is primarily focused on reachability problems, we believe that the principles of our system are applicable to performance and other wide-area network problems.

This work makes three research contributions. First, it presents a study on network outages affecting Amazon EC2 data centers [11]. Second, it presents new tools for measuring Internet paths in the presence of reachability problems, and applies these tools in an algorithm for isolating forward, reverse, and bidirectional path failures at an IP-link granularity. Lastly, it presents an

in-depth analysis of the network reachability problems observed by our system.

The remainder of this thesis is organized as follows. We present a study on network reachability problems and define *LIFEGUARD*'s focus in Section 2. In Section 3 we describe the design and implementation of *LIFEGUARD*. We present an evaluation of our system in Section 4, and use it to study outages in Section 5. We provide a discussion of our work in Section 6, and present related work in in Section 7. Finally, we conclude in Section 8.

2 Motivation

This section motivates the need for a wide-area network fault isolation system. We begin by surveying the frequency, duration, and general characteristics of network outages. We then demonstrate that currently available tools for isolating and circumventing reachability problems are inadequate. Finally, we draw on these findings to define the specific types of outages our system focuses on and the information we hope to provide.

2.1 Network Reachability Problems

LIFEGUARD is motivated primarily by the lack of reliability in the Internet. Stories of network faults abound. Consider, for example, the elderly Georgian woman who managed to disconnect Armenia from the Internet for more than four hours while scavenging for copper [12]. From first hand experience as well, I found at an internship with a content distribution network that operators spend a surprisingly high portion of their time in triage of network connectivity problems.

2.1.1 EC2 connectivity

We argue that outages are prevalent and widespread, affecting both public and commercial networks. First we demonstrate that Amazon EC2 data centers [11], a critical piece of cloud infrastructure, are subject to both short- and long-term network connectivity problems. To the extent of our knowledge, this is the first public study of its kind.

For this study we rented EC2 instances in each of the four AWS regions for the two month period starting July 25th, 2010. Our vantage points issued pings every 30 seconds to 250 targets, consisting of 5 routers each within the 50 ASes of highest degree according to UCLA's AS topology data [13].

Figure 1 shows the duration of outages observed during the month of August. On the x-axis we show the duration of the outages we observed in minutes, and on the y-axis we show the cumulative fraction of outages. Defining an outage event as four or more dropped pings seen from a single vantage point, we observed 12,371 outages during this period. Roughly 90% of these outages were shorted-lived, lasting less than 10 minutes. The remaining 10% of outages lasted significantly longer. We find this to be a shocking breach of reliability, considering that these ˜1,200 outages were observed along paths to highly-connected routers in the core of the Internet.

In Figure 2 we show the correlation of outages across our four vantage points. Note that these are the same outages shown in Figure 1, bucketed by the number of vantage points experiencing a reachability problem for a given target during the same period. We see that 79% of the reachability problems observed during this period were partial, with one or more vantage points maintaining connectivity with the destination.

A number of interesting conclusions follow from this experiment. First, the long tail of outage durations suggests that we may need different approaches to combating short- and long-term reachability problems. While recent architectural solutions excel at preventing transient packet loss

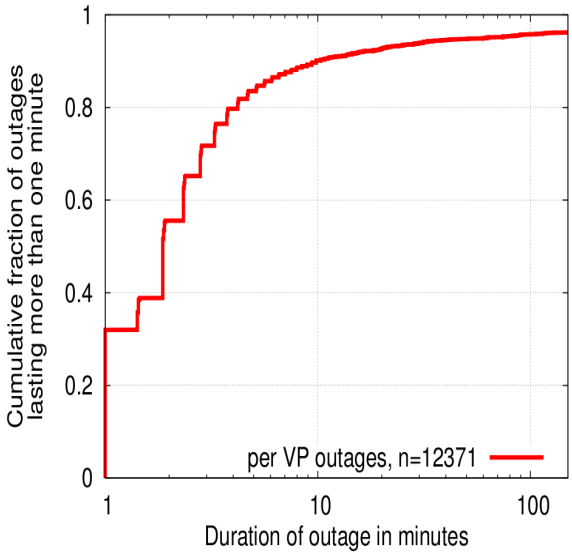


Figure 1: Fraction of per-VP outage events versus outage duration in minutes. Note that the x-axis is on a log-scale. We found that 10% of outages lasted longer than 10 minutes

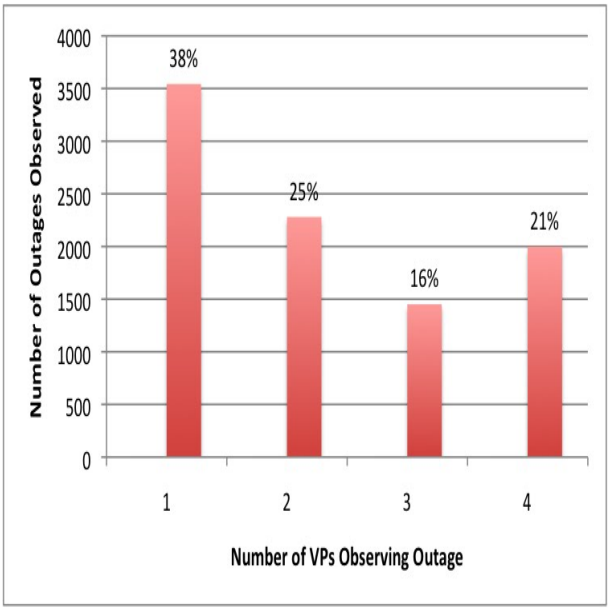


Figure 2: Correlation of VP observations. Two VPs are considered correlated if they observe dropped pings to the same destination within the same 90 second window. Roughly 80% of observed outages were partial.

[3] a measurement-based approach appears to be necessary for dealing with long-lasting failures. Second, this experiment shows that Hubble's characterization of network outages generalizes to at least one commercial network. This is significant, since the reliability of cloud infrastructure is increasingly important as more applications and data move to the cloud. Moreover it shows that wide-area network fault isolation is a particularly difficult problem, since companies such as Amazon are demonstrably unable to prevent or quickly remediate network outages despite vested interest in maintaining reliable network connectivity for their customers and clients.

2.1.2 Further Lessons from Hubble

We argue that the reachability problems affecting EC2 are representative of a wide range of networks. With Hubble, a system for monitoring and classifying long-term network outages across the Internet [7], we also found that:

Outages are prevalent. Over a three-week period Hubble observed more than 30,000 outages lasting longer than 15 minutes. These outages affected roughly 10% of all prefixes monitored by Hubble, where Hubble's target list included 92% of edge prefixes and all core prefixes on the Internet. Based on these findings, network outages clearly constitute a significant impediment to the reliability of the Internet.

Most do not show up in BGP feeds. 63% of the outages detected by Hubble were not visible in public BGP feeds [14]. This implies that data-plane measurements are necessary to gain reasonable coverage of outages.

Most are partial. 67% of the outages detected by Hubble were partial, meaning that some vantage

points had connectivity with the destination while others did not. Partial outages imply that working paths exist, suggesting that it may be possible to route around the problem. Furthermore, connected vantage points can issue or receive measurements on behalf of disconnected vantage points during partial outages, enabling more accurate isolation of failures.

Many are long lasting. More than 60% of the outages detected by Hubble lasted longer than 2 hours, and ˜10% lasted longer than a day. Non-transient outages require manual intervention, yet the duration of these outages clearly demonstrate that existing troubleshooting techniques are inadequate.

Problems are not just in the destination AS. All of the outages observed over the three-week period involved a failure outside of the destination AS. With failures occurring in the core of the Internet, network operators need an interdomain troubleshooting approach.

Most partial outages are unidirectional. Hubble was able to infer that failures were located on either the forward or reverse path in 80% of the partial outages it observed. A path that is compliant with routing policies in one direction should generally also be compliant in the other, making these problems particularly interesting.

Some of these results have been corroborated by several other studies. For example, the probability of encountering a major routing pathology along a path has been found to be between 1.5% and 3.3% [15, 16].

2.2 Measurement Tools

The prevalence of long-lived outages suggests that network operators have limited ability to detect, isolate, and avoid failures outside the local domain. We argue that this is largely due to limitations of currently available measurement tools.

`traceroute`, the most widely used network diagnostic tool today, allows operators to infer the path traversed by packets to arrive at a given destination. In short, `traceroute` works by issuing packets with incrementally increasing "expiration timers" (TTLs), evoking an error message from each router in-turn along the forward path.

Because Internet paths are often asymmetric, differing in the forward and reverse direction between endhosts, `traceroute` provide incomplete or misleading information in the presence of network failures; with control over only one endpoint, operators have no way to distinguish reverse and forward path failures when observing a sequence of TTLs yielding no responses. Isolating and avoiding failures on the reverse path is particularly difficult with `traceroute`; even if network operators could infer the direction of a failure, without control of the destination `traceroute` provides no visibility into the path taken by return packets.

We illustrate these difficulties with a specific outage reported on the Outages.org mailing list. On December 16th, 2010, an operator reported that "It seems traffic attempting to pass through Level3's network in the Washington, DC area is getting lost in the abyss. Here's a trace from VZ residential FIOS to www.level3.com:"

```
1 Wireless_Broadband_Router.home
2 L100.BLTMMD-VFTTP-40.verizon-gni.net
3 G10-0-1-440.BLTMMD-LCR-04.verizon-gni.net
4 so-2-0-0-0.PHIL-BB-RTR2.verizon-gni.net
5 0.ae2.BR2.IAD8.ALTER.NET
6 ae7.edge1.washingtondc4.level3.net
7 vlan80.csw3.Washington1.Level3.net
8 ae-92-92.ebr2.Washington1.Level3.net
9   *    *    *
```

```
1 192.168.1.1
2 l100.washdc-vfttp-47.verizon-gni.net
3 g4-0-1-747.washdc-lcr-07.verizon-gni.net
4 0.ae1.br1.iad8.alter.net
5 ae6.edge1.washingtondc4.level3.net
6 vlan90.csw4.washington1.level3.net
7 ae-71-71.ebr1.washington1.level3.net
8 ae-6-6.ebr2.chicago2.level3.net
10 ae-1-100.ebr1.chicago2.level3.net
11 ae-3-3.ebr2.denver1.level3.net
12 ge-9-1.hsa1.denver1.level3.net
13 * * *
```

The dropped probes in this trace after the 8th hop, ae-92-92.ebr2.Washington1.Level3.net, suggest that the problem may be in Washington DC. Several minutes later though, a second operator replied with another trace to the same destination making it appear that the problem may be near Denver (12 ge-9-1.hsa1.denver1.level3.net). With only this information, the operators are left with several questions unanswered: are they even seeing the same problem? Are the forward and reverse paths symmetric? What do the reverse paths look like? Can other vantage points reach the destination? Clearly for this particular outage, the measurement tools available to the operators did not suffice.

2.3 Problem Definition

We now define the focus of this thesis. While the goal of Hubble was to monitor the entire Internet and learn aggregate properties of outages, this work seeks to apply Hubble's lessons in order to pinpoint and circumvent specific problems. We focus on (i) wide-area, (ii) long-lasting, (iii) partial, and (iv) high-impact outages. Our emphasis on (i) is motivated by operators' lack of visibility into problems outside of their own domain, where network faults may obstruct traffic from a wide range of sources; (ii) and (iii) ensure that we are identifying relevant and actionable problems, where problems are not resolving themselves and working paths exist in the network; and (iv) optimizes the value of our system's results.

We take a measurement-based approach; while architectural solutions excel at preventing certain types of network outages, we believe that long-lasting failures and misconfigurations are ultimately unavoidable. To improve the reliability of the Internet network operators require a mechanism for identifying and reacting to inevitable network faults.

Thus, we seek a system with the following properties:

Accuracy: Ideally the system should consistently identify the exact failed IP-link or router causing an outage, regardless of whether the failure lies on the forward or reverse path. Accuracy is especially important for two reasons. First, attempts to route around failures are ineffective without correct location information. Second, network operators must be extremely circumspect when claiming that another domain is responsible for an outage.

Scalability: Not all networks have the resources necessary for wide-area fault isolation. Thus, network operators need a global information service capable of monitoring outages at Internet-scale. For operators to effectively react to problems however, the system must be able to detect and isolate outages within seconds or minutes. Therefore the system's measurement and computational requirements should feasibly scale. Minimizing measurement overhead is especially important since measurement traffic may exacerbate problems along already problematic Internet paths.

Relevance: As with any monitoring system, alarms should not be generated so often as to be ignored; the outages isolated by our system should be actionable and highly relevant. Furthermore,

to facilitate remediation of failures, the reports generated by our system should include additional information such as alternate working paths that may be used by network operators to route around failures.

Deployability: We seek a system that is deployable today, without requiring changes to infrastructure within the network. Moreover, since outages may occur anywhere on the Internet, the system should not rely on control of the destination.

3 System Design and Architecture

LIFEGUARD is our system for addressing the problems outlined in Section 2. In this Section we begin by motivating the measurements needed for IP-link isolation. We then describe the architecture of *LIFEGUARD* in detail.

3.1 Required Measurements

LIFEGUARD attempts to isolate the exact IP-link causing an outage. Here we outline the measurements needed to achieve this goal by walking through a particular outage detected by our system on February 24th, 2011. At the top of Figure 3 is our source vantage point, plgmu4.ite.gmu.edu, and at the bottom is our destination, 77.221.192.1 (ASN 42676).

To detect the outage, plgmu4.ite.gmu.edu and other vantage points acted as **ping monitors (details in 3.3.2).** In *LIFEGUARD*, ping monitors issue probes to selected targets at regular intervals. A centralized controller then pulls state from these vantage points and triggers additional measurements when a previously pingable target consistently fails to respond to at least one vantage point. In this case, plgmu4.ite.gmu.edu suddenly observed consecutive dropped pings for the destination 77.221.192.1.

After detecting the outage, our system attempts to isolate the location of the failure. As described in Section 2.2, `traceroute` does not provide accurate information in the presence of failures. In Figure 3, a `traceroute` issued from the source is denoted with the set of solid black lines. Note that the traceroute halts at ZSTTK-Omsk-gw.transtelecom.net. This seems to suggest that the failure may be directly adjacent to this hop: between ASes 20485 and 21127 (ASNs not shown in the figure). However, without further information network operators cannot be sure, since the probes may have been dropped on a reverse path back from a hop beyond ZSTTK-Omsk-gw.transtelecom.net.

To isolate the direction of a failure, *LIFEGUARD* makes use of **spoofed pings (Not shown in figure. Details in 3.4.1)**. Leveraging a technique proposed by Hubble, plgmu4.ite.gmu.edu issues pings towards the destination, with the source address set as other vantage points in our system. In this way we ensure that the probes traverse only the forward path. Likewise, other vantage points with connectivity to the destination spoof probes as the source, ensuring that the probes traverse only the reverse path.

In our example, spoofed probes sent from the source reached our receivers, but no probes reached the source, implying a reverse path failure. Having drawn this conclusion, our system then attempts to measure the working forward path. This is accomplished with **spoofed traceroute (details in 3.4.2).** Generalizing the technique for isolating the direction of a failure, plgmu4.ite.gmu.edu issues spoofed TTL-limited probes towards the destination, allowing us to measure 3 additional forward hops including the destination. The spoofed traceroute is denoted with the blue edges in the Figure 3.

To further isolate the location of the failure *LIFEGUARD* leverages the data given by its **topology mapper (details in 3.5.1).** If *LIFEGUARD* only launched measurements when it first

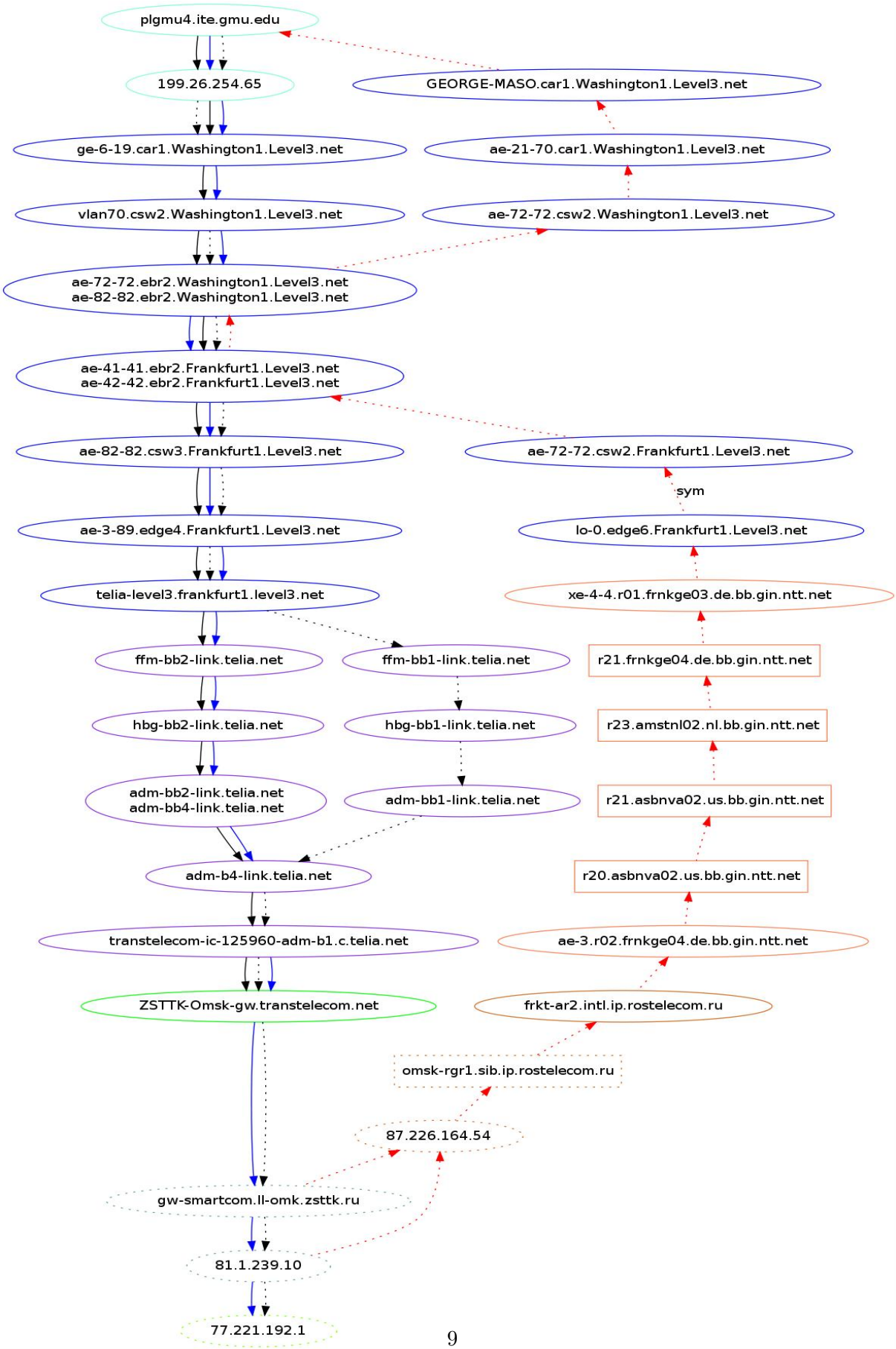


Figure 3: Example Outage.

suspects a problem, it would not be able to obtain a view of what paths looked like before the problem started. Our system thus maintains an atlas of forward and reverse traceroutes for every target and intermediate hop.

In the graph above, historical forward and reverse traceroutes are denoted with the dotted black and red lines, respectively. We see that the two hops closest to the destination share a common historical reverse path, i.e. 87.226.164.54 → omsk-rgr1.sib.ip.rostelecom.ru.

As our final isolation step, *LIFEGUARD* employs **router liveness inference.** We have our source issue pings to all hops along historical paths to identify those that are reachable, helping us further isolate the location of the failure and identify possible alternative working paths. We note that because some routers are configured to ignore ICMP pings, we maintain a database of historical pingability to distinguish connectivity problems from routers not configured to respond to ICMP probes.

In this case, we found that all hops before omsk-rgr1.sib.ip.rostelecom.ru (AS 12389) were reachable (denoted with solid outlines), while all those beyond were not, although they had been ping responsive in the past. Because both historical reverse paths from unresponsive forward hops traversed omsk-rgr1.sib.ip.rostelecom.ru, it seems highly likely this is the point of failure. This is further supported by the fact that all unreachable hops except omsk-rgr1.sib.ip.rostelecom.ru were pingable from other vantage points (denoted with dashed lines), indicating that other outgoing paths were not affected.

In this case, *LIFEGUARD* was unable to measure the reverse path. This is to be expected, since reverse traceroute requires that the source be able to receive probes from the destination. In the case of forward outages however, *LIFEGUARD* attempts to measure the working reverse path with **reverse traceroute (details in 3.4.3).** We note that since the source cannot reach the destination in the presence of outages, our system employs a spoof-only version of the reverse traceroute algorithm.

3.2 System Architecture

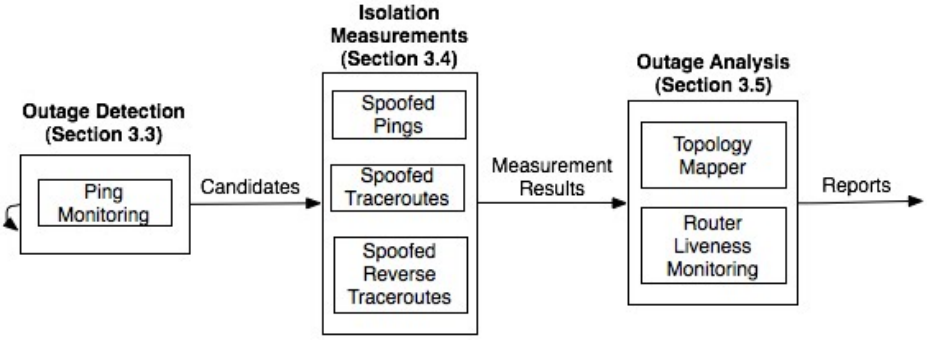


Figure 4: Diagram of the *LIFEGUARD* archicture

Having motivated the measurements needed for isolation, we now present the overall architecture of *LIFEGUARD*. As depicted in Figure 4, *LIFEGUARD* consists of three main components: outage detection, isolation measurements, and outage analysis.

3.3 Identifying Possible Outages

As implied by the results in Section 2.1.2, control-plane measurements do not suffice to cover all reachability events. Therefore *LIFEGUARD* uses an active measurement approach to detecting outages. The targets monitored by *LIFEGUARD* are carefully chosen to gain the greatest possible coverage of outage events with a relatively small target list. Here we describe the details of our target selection strategy and how the active ping monitoring component of our system works.

3.3.1 Target selection

To inform target selection, we first focus on collections of routers within ISPs known as Points-of-Presence (PoPs). Because PoPs represent a collection of interfaces at the same location, they allow us to avoid redundant probing while still gaining widespread coverage of outages.

To form our target list we generate a list of PoPs and member routers using the iPlane dataset. We next rank these PoPs by the number of paths in the iPlane atlas that intersect each PoP. To reduce bias from vantage points with many paths intersecting the same PoP, we omit all paths from a VP to a PoP if those paths represent more than 1% of the VP's paths. We find that, even with this strict threshold for excluding paths from the analysis, the top 500 PoPs are intersected by 63% of the paths in the iPlane atlas. This result implies that large portions of Internet paths traverse small numbers of PoPs; therefore, the characteristics of paths to these PoPs are likely to affect large portions of Internet traffic.

In addition to highly-connected PoPs, we also probe routers within stub ASes lying on paths intersecting these core PoPs. This decision is motivated by earlier results showing that a large portion of network reachability problems occur in the lower tiers of the AS hierarchy [8, 7].

Given the overhead of our measurements, *LIFEGUARD*'s target selection strategy is crucial for achieving our goal of Internet-scale outage monitoring. We admit that there is large opportunity for improvement with regards to our current approach. We discuss this issue further in Section 6.

3.3.2 Ping monitoring

LIFEGUARD's active ping monitoring component serves to detect medium- to long-term outages. Relative to more heavy-weight approaches such as pervasive traceroutes, the minimal measurement overhead of pings allows us to react to outages in a timely manner.

We design the ping monitors to detect as many outages as possible, while minimizing spurious isolation measurements sent to targets that are reachable or experiencing transient packet loss. We therefore consider a target unreachable only after it has failed to respond to 4 or more consecutive pings. Moreover, because we focus on avoidable and partial reachability problems, we require that at least two vantage points maintain connectivity with the destination before issuing further measurements. In our experience these heuristics are effective at eliminating most transient or complete outages.

To avoid exacerbating network problems, the centralized controller ensures that the isolation algorithm and corresponding measurements are executed at most once every hour to any given target. The system also examines logs once a day and replaces ping monitors that frequently fall behind, or fail to receive ping responses from an improbably high portion of targets.

3.4 Isolation Measurements

Given a list of targets potentially experiencing network outages, *LIFEGUARD* triggers additional measurements in an attempt to measure working paths and isolate the probable source of the problem.

3.4.1 Spoofed pings

LIFEGUARD employs spoofed probes to differentiate forward, reverse, and bi-directional path failures. We briefly recapitulate the technique taken from Hubble here. To ensure that probes only traverse the forward direction, we have our source issue probes towards the destination with the source address set to other vantage points in our system, such that replies are not returned to the source. Similarly, auxiliary vantage points issue probes with the source address set to the source, ensuring that probes only traverse the reverse direction. If the spoofed probes are received on either end, we can infer that the corresponding path is working. Note that because we only make inferences from the receipt of spoofed packets, not their absence, we avoid drawing false conclusions.

3.4.2 Spoofed forward traceroute

As described in Section 2, `traceroute` provides incomplete information in the presence of outages. For *LIFEGUARD*, we have developed a distributed version of `traceroute` employing spoofed TTL-limited probes to measure the full forward path in the case of reverse path failures.

When a reverse path failure is detected, our centralized controller begins by choosing a set of vantage points with working paths to the destination to act as receivers for spoofed probes. The source then issues spoofed probes with incrementing TTL's as each of the receivers in turn. Because routers erase the original TTL when sending a ICMP time-exceeded message, we encode the TTL in the sequence number field of the packet header which is then reflected in the reply. In this way we ensure that we infer the correct order of hops along the forward path.

We note that in our experience deploying *LIFEGUARD*, we have found that normal traceroutes can occasionally yield more hops than a spoofed traceroute. Our hypothesis is that intermediate hops occasionally have working reverse paths back to the source, but not working forward path to auxiliary vantage points. Therefore to gain the most possible coverage, *LIFEGUARD* employs both normal and spoofed traceroutes.

Measuring the forward path is useful for two reasons. First, the source can ping all of the forward hops to see which have paths back to it, and issue reverse traceroutes for all that do, thus determining a set of working paths and helping narrow down where the reverse path might be failing. Second, because working paths in one direction should be compliant with the business policies of intermediate ASes in the other, the forward path could be used as a feasible alternative return path from the destination AS.

3.4.3 Reverse traceroute

LIFEGUARD makes use of the reverse traceroute algorithm for measuring reverse paths [9]. In short, reverse traceroute pieces together path segments with record route and timestamp probes to successive intermediate hops from the destination, issuing spoofed probes from nearby vantage points when the source is too far away from the next hop. When reverse traceroute is unable to measure the reverse path for a particular hop, it assumes that the last link on a forward traceroute to the hop is symmetric. In general, the number of symmetry assumptions made in a reverse traceroute affects the accuracy of the result.

In the context of reachability problems our source may not be able to probe the destination. For this reason we also make use of a slight variation of the reverse traceroute algorithm described in [9], wherein we issue only spoofed measurements from connected vantages points, and base symmetry assumptions off historical forward or reverse traceroutes rather than traceroutes issued directly from the source.

Relative to traceroute, reverse traceroute incurs a fairly high measurement and coordination overhead; reverse path measurements require roughly 4 times the measurement probes, and take on the order of hundreds to thousands of seconds to complete. Without optimizations, this stands as an impediment to our goal of scalability. We briefly describe two sets of optimizations we have made to the reverse traceroute system to achieve the large-scale reverse path measurements required by *LIFEGUARD*.

First, in the initial version of reverse traceroute, all vantage points in the system issued spoofed probes to intermediate hops until a vantage point happened to be in close enough proximity to elicit reverse path information. Because many destinations rate-limit after only a few consecutive probes, this meant that probes sent from in-range vantage points may have been dropped. Furthermore, sending large amounts of probes to each intermediate hop was clearly prohibitive for scalability. To overcome these issues, we built a system which regularly generates a "map" of the Internet by issuing probes from all vantage points to selected targets. With the resulting data, the system generates an ordering of vantage points which are most likely to yield reverse hops for any given prefix. Generating an ordering which covers the widest possible area of a prefix using the minimum number of vantage points is an instance of the well-known set cover optimization problem. With the standard greedy algorithm, we were able to show that in the median case a single vantage point covers 95% of the addresses in a prefix. Reverse traceroute now uses these predictions to choose the optimal vantage points for issuing probes.

Second, reverse traceroute previously did not persistently cache information from previous measurements. Reverse traceroute now maintains a feedback loop for retaining reverse path information, including a path-segment caching layer, a new batching and staging system for scheduling measurements, and a database of router behavior.

In a small measurement study over 2000 randomly chosen (source,destination) pairs, we found that the initial version of reverse traceroute took 9.8 hours to complete all measurements in a single batch. With optimizations in place, the measurements were completed in approximately 45 minutes, representing a 13-fold speedup. We believe this benchmark to be an accurate representation of our workload, since *LIFEGUARD* issues a large number of measurements in parallel to maintain historical path atlases.

3.5 Fault Analysis

Having detected a reachability problem, network operators would like to know (a) which AS is responsible for the problem, so that they can know who to contact, (b) which router or link is likely causing the issue, and (c) any alternative working paths which may used to circumvent the outage. Section 3.5.2 shows how *LIFEGUARD*'s measurements begin to address (a) and (b). In Section 3.5.3 we describe our approach to addressing (c). First, we outline some additional measurements that aid fault analysis.

3.5.1 Topology mapper

The vantage points in *LIFEGUARD* continually issue forward and reverse traceroutes to all targets and intermediate forward hops. This is in addition to the on-demand measurements issued in the event of a reachability problem. The resulting database provides *LIFEGUARD* with a set of relatively recent paths to compare against when a failure occurs; if we probed a path only when it was experiencing problems, we would not know what the working path looked like before the problem.

This information is used for several purposes. First, historical forward and reverse hops act as a "seed" for spoofed reverse traceroutes, allowing us to make symmetry assumptions when the

source is not able to issue a successful traceroute to the destination. Historical path information is also useful for fault analysis; by pinging hops along historical paths the source can determine which routers are reachable and which aren't, helping to determine the location of the failure. With reachability information we can identify possible alternate working paths, infer path changes, and locate suspected failed routers in the case of reverse path outages where we have no visibility into the current reverse path.

Several studies have found that Internet paths are fairly stable. For example, Paxson [15] found that 87% of routes persisted for more than 6 hours. Thus, we believe that historical path information provides valuable additional information for isolation, so long as the atlas is refreshed on a regular basis.

Algorithm 1 Identifying the suspected failed link

```
if REVERSE PATH FAILURE then
  suspects = {destination's historical reverse path}
  for all h ∈ pingable forward hops do
    suspects -= {h's measured reverse path}
    for all r ∈ h's historical reverse path do
      if r is pingable from source then
        suspects -= {r}
      end if
    end for
  end for
  if suspects == ∅ or all suspects are pingable (reverse path change) then
    return destination
  else
    return unresponsive hop adjacent to the responsive hop farthest away from S in suspects
  end if
else
  m = last ping responsive spoofed forward hop
  n = last non-spoofed forward hop
  // measurements sometimes differ in length
  return maxttl(m, n)
end if
```

3.5.2 Failure Isolation Approach

LIFEGUARD aims to identify the exact IP-link causing a network reachability problem. Note that we use the term "IP-link" loosely to encompass a broad range of failures, including routers with incomplete routing tables, misconfigurations, hardware failures, and fiber cuts. We also note that identifying the exact problem may not always be achievable; there may be multiple problems at once, multiple problems with the same root cause, failures on new paths we are unable to measure, evolving problems as operators react or problems cascade, or unexpected router behavior. Consider for example, a path S-A-B-C-D. If the link C-D fails, it is possible that B will drop its routing table entry for D and all subsequent packets destined for D upon discovering that the current path is not working. A traceroute from S will then stop at B, leading us to the false conclusion that the link B-C is down. *LIFEGUARD*'s approach to handling these complications is to identify the hop that seems to explain the majority of lost probes, either because we have verified that the failure

lies on a path that the router has recently started using, or because it lies at a convergence point where all hops beyond are unreachable from the source.

Our isolation algorithm is depicted above. The specifics of the algorithm differ depending on the direction of the failure inferred with spoofed probes:

Forward and bidirectional failures. For forward and bidirectional failures, it is likely that the last ping responsive router on S's forward path is adjacent to the failed link. We identify this router by taking the union of the spoofed forward traceroute and the normal traceroute issued by S and returning the hop closest to D.

Reverse path failures. In the case of reverse path failures we can issue pings to the hops on historical reverse path from D to S in order to further isolate the outage. When the destination is still using the same reverse path, the failed link is likely adjacent to the pingable hop furthest from the source. Otherwise if all of the hops on the historical reverse traceroute are pingable, it is likely that the destination has switched over to a new reverse path.

We can further constrain the set of possible failed routers by issuing reverse traceroutes to all reachable forward hops. This provides us with a set of working paths, helping us further distinguish failed hops.

It is worth noting that we are not capable of directly measuring failed reverse paths, since spoofed probes will consistently fail to reach S. As a consequence, we have limited ability to differentiate failed reverse links from reverse path changes. Therefore we conservatively conclude that the reverse path suspect is either adjacent to the failure, or has switched over to a new faulty reverse path.

Multiple failures. Algorithm 1 assumes that an outage is caused by a single router or link. In the unlikely event that multiple points of failure exist, our technique will tend to identify the failure closest to the source. In these cases we can simply iterate: after having fixed the failure closest to the source, subsequent attempts will identify the next closest failure, and so on.

3.5.3 Identifying alternate paths

Having isolated the location of an outage, network operators would benefit from the ability to route traffic along alternate paths while a failure is being repaired. Our system attempts to identify working alternate by checking the reachability of hops along the (i) historical forward path, (ii) historical reverse path, and (iii) current working direction in the case of uni-directional failures. Any paths that are entirely reachable from the source are returned as possible alternatives.

It is worth noting that the last hop of all paths, *viz.* the destination, will by definition not be reachable from the source. Therefore it is possible that our technique may output false positives in the case that all hops except the destination are pingable yet the link directly adjacent to the destination is down. We argue that the possibility of false positives is relatively unimportant. Network operators have limited ability to affect the return path chosen by other ASes, so alternate reverse path information would primarily be used to provide the responsible party with additional information. In the case of a failed link directly adjacent to the destination, the destination AS should often be able to use a different egress while the link is being repaired.

This technique ensures that a working set of *links* exists in the network. However, it does not ensure that a valid BGP route exists for this path; without access to BGP feeds, our system has no way to verify whether alternate routes exist in practice. Nonetheless, even if a path existed in the control-plane, network operators would need to verify that the data-plane path was functional.

LIFEGUARD also employs a second method for identifying alternate paths. In historical forward and reverse paths, we often find to our surprise that hops directly adjacent or within the

destination AS are pingable from the source. The paths to these hops must differ in some respects. Therefore we issue traceroutes to any such hops to discover how their paths differ. If the paths are valid, a network operator could feasibly route traffic to the destination along the same path.

We have also considered generalizing this approach: rather than identifying pingable routers only on historical measurements from a particular source, it could be fruitful to make use of a full-blown topology map encompassing a mesh of historical paths. For example, suppose that a destination AS has 6 ingress routers. If we kept track of these ingress points, we could ping them whenever an outage occurs. Suppose we find that the current ingress has failed, but others are working. Network operators could feasibly route around the failure by choosing a different egress point out of their own network in an attempt to affect the ingress used to enter the destination AS.

Ideally our system should return paths compliant with the routing policies of intermediate ASes. In the case of a uni-directional failure, we posit that the working direction should also be compliant in the opposite direction, at least with respect to the Geo-Rexford policy model [17]. If the direction of traffic is reversed, it is still the case that no stub ASes transit traffic, at most one peering link is traversed, etc.

For all partial failures, we know that at least one alternate path exists, *viz.* disconnected vantage point → central controller at UW → connected vantage point → destination. While this path is certainly not policy compliant (UW is a stub AS), it suggests the possibility of an overlay detouring service, where nodes with connectivity could provide uninterrupted service between arbitrary end-hosts.

3.5.4 Filtering Heuristics for Relevant and Actionable Results

We have designed *LIFEGUARD* to present relevant and actionable results to network operators. *LIFEGUARD* therefore applies the following filtering heuristics before generating reports in order to avoid alarming on an overwhelming number of transient or unimportant outages:

- 2 or more sources without connectivity must have observed 4 or more consecutive dropped pings. Similarly, 1 or more source must have been able to reach the destination for 4 or more measurement rounds. This is to filter out cases of lossy links or transient routing loops.
- 1 or more sources without connectivity must have seen at least one ping response from the destination in the past. This is largely intended to filter out misconfigured PlanetLab nodes.
- We also exclude any sources receiving ping responses from less than 10% of targets to filter out misconfigured or failed PlanetLab nodes.
- The destination must not have become reachable from the source during or directly after measurements are issued. Similarly, we exclude cases where both directions between the source and destination were successfully traversed with spoofed probes. This is to ensure that reachability problems persist by the time network operators examine *LIFEGUARD*'s reports.
- We observed a large number of cases where the failed link appeared to be directly adjacent to the destination. We ignore cases where the forward measurements reach the destination's AS, since we are primarily interested in wide-area network

4 System Evaluation

LIFEGUARD is now a fully automated, continuously running system for detecting and isolating outages. In this section we describe the details of its current deployment and evaluate how well it has met our design goals. In Section 5 we characterize the reachability problems we found with our system.

4.1 Experimentation Platform

LIFEGUARD currently employs 12 spoofing PlanetLab [18] nodes as measurement issuers and receivers. We find that a dozen geographically distributed vantage points are sufficient for measurement coordination and outage detection. These vantage points issue pings to all targets every 2 minutes, update historical forward paths every 5 minutes, and refresh historical reverse path information approximately every 4 hours.

Recall that our intent with *LIFEGUARD* is to focus on (i) wide-area, (ii) long-lasting, (iii) partial, and (iv) high-impact outages. With this in mind, we chose to monitor the following targets:

- 16 cache servers located in each of the points of presence (PoPs) of a medium sized content distribution network (CDN). These targets allow us to correlate ping reachability with application-level reachability, as described in Section 5. They also allow us to continue evaluating the connectivity of cloud infrastructure, similar to the EC2 study outlined in 2.
- 83 routers in highly-connected PoPs located in the core of the Internet based on iPlane's traceroute atlas [10].
- 185 targets located on the edge of the Internet, with paths traversing at least one of the 83 centralized PoPs. These targets are statically chosen; we do not re-check that these PoPs still lie on intersecting paths.
- 76 PlanetLab nodes. These targets provide us with a mechanism for validating our results, as described in Section 4.2.

For this and the subsequent section we consider a snapshot of our logs ending May 25th consisting of 27,291 partial outages, 788 of which passed filtering heuristics. The vast majority of the 27,291 partial outages failed to pass filters because the target had become reachable by the time measurements were finished.

4.2 Accuracy

At 11:36 PM on May 25th, three of our vantage points observed a forward path outage to three distinct locations. All of these outages were isolated to ae-3-3.ebr1.Chicago2.Level3.net, a core router in Level3's Chicago PoP. Several minutes later, a network operator posted the following to the Outages.org mailing list: "Saw issues routing through Chicago via Level3 starting around 11:05pm, cleared up about 11:43pm.". This was later confirmed by several other network operators. This example illustrates that our isolation approach is capable of identifying important failures. However, we do not claim that this example validates our isolation results, since Level3 did not verify the isolation.

In general, obtaining ground truth for wide-area network faults is difficult: emulation of failures is not feasible, only a handful of networks publicly post outage reports, and network operators take offense to spam sent from researchers. At the time of this writing we did not believe that *LIFEGUARD* was mature enough to justify direct contact with network operators. In the near

future we plan to set up a url for the project, post a public solicitation to Nanog, and attempt to elicit confirmation of our isolation results from specific networks.

In lieu of ground truth data, we also attempted to evaluate our results by comparing against cases where the destination was under our control and able to issue measurements back to the source, thereby yielding a more complete view of the network topology. Unfortunately, due to an undetected concurrency bug, we were only able to gather a small number of such cases passing filtering heuristics, most of which were inconclusive. The concurrency bug is now fixed, and we hope to verify additional cases in this manner in the near future.

4.2.1 Outage Categorization

We additionally evaluate *LIFEGUARD*'s accuracy by categorizing the outages we observed according to our confidence in the isolation results. The six categories we chose represent broad classes of failure scenarios where *LIFEGUARD* has differing levels of visibility into the problem. We generated these categories by examining individual cases by hand, and then automatically classifying the remaining cases.

The first three categories pertain to forward and bidirectional outages. They are illustrated in Figure 5.

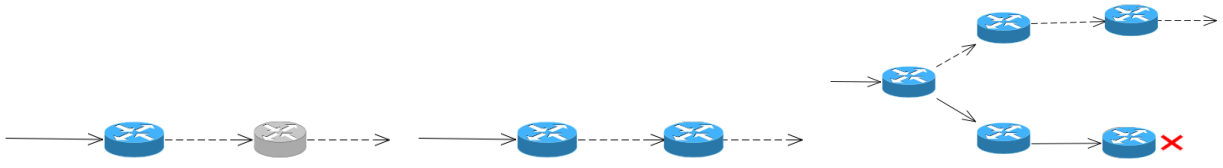
No path change, next historical hop down: In this classification, the normal and spoofed forward traceroute terminate at the same hop, and follow the same path as the historical traceroute. Moreover, the next hop on the historical traceroute is not pingable from the source.

No path change, next historical hop pingable: In this classification, the measured forward paths follow the same path as the historical traceroute, and the normal and spoofed forward traceroute terminate at the same hop. However, the next hop on the historical forward path is reachable from the destination. This suggests that the last hop may have changed forward paths, or otherwise has a destination-based routing problem.

Path change: Here the measured paths differ by one or more prefixes from the historical traceroute. As a result, we have limited visibility into what hops lie beyond the last pingable hop of the traceroute, and whether they are reachable from the source.

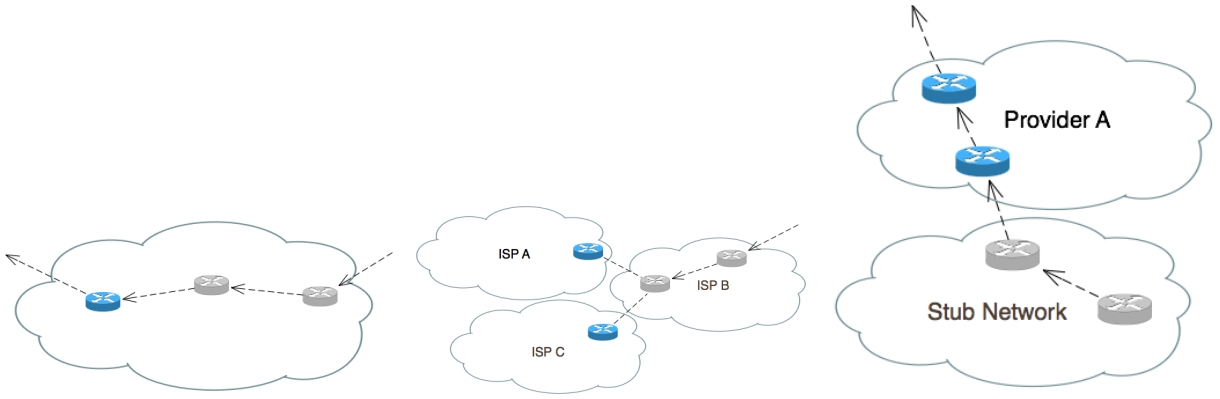
Note that for bi-directional failures, if paths are symmetric, historical reverse information may further support our hypothesis that the hop directly after measured forward path is adjacent to the failure.

The last three categories pertain to reverse path failures, and are illustrated in Figure 6. For these categories, we use the term "reachability horizon" to denote the point at which there is a clear distinction between hops on the historical reverse paths that are reachable from the source and hops closer to the destination that are not. We note that for reverse path failures we have no way to detect path changes, since the current reverse path from the destination cannot be measured. Thus, if paths have not changed, our system correctly returns the failed router, else we identify the router that has switched over to a new faulty reverse path.



5 (a) Next hop unreachable 5 (b) Next hop reachable 5 (c) Path change

Figure 5: Classes of forward and bidirectional path outages. Measured paths are shown in solid lines, and historical paths are shown with dotted lines.



6(a) Horizon confined to one AS 6(b) Horizon crosses AS boundaries 6(c) Stub network, provider link down

Figure 6: Classes of reverse path outages. Routers reachable from the source are shown in blue, and unreachable routers are shaded. Note that 6(c) is a special case of 6(b).

Reachability horizon confined to one AS: In this category, the boundary between reachable and non-reachable hops is confined to one or two PoPs within the same AS. Even if there was an intra-domain path change since the historical reverse path was measured, the fact that the reachability horizon is currently confined to a single AS gives is moderate confidence that the failure is within close proximity of the horizon. Moreover, for these cases we have information about which PoPs within the AS are reachable and which are not. For the next category, it is usually the case that all PoPs within AS containing the failure are not reachable from the source.
Reachability horizon spans multiple ASes: The classification is similar to the last, except that the reachability horizon is found at a peering link between ASes such as those found at an Internet eXchange Point (IXP). It is possible that the current reverse path now passes through a different AS than the historical reverse path, so we place low confidence on our results in these cases.
Broken provider link for multi-homed stub AS: For this category, the reachability horizon is found at a border router directly adjacent to the destination AS. Because other vantage points have connectivity with the destination, it must be the case that other working ingress links exist. The accuracy of the historical reverse traceroute for these cases is fairly unimportant, since the fact that most or all hops on historical paths within the destination AS are not pingable from the source strongly suggests that the provider link is down. We note that this category is a special case of the previous.

Table 1 shows how many problems were assigned to each class. Note that without accurate historical reverse path information, our system is unable to accurately isolate reverse path failures: although we can identify where the failure is *not* by issuing reverse traceroutes to pingable forward hops, we have no way to identify unreachable reverse hops. The large number of reverse path failures without accurate historical reverse path information is largely due an undiscovered bug in our system described later in this section.

4.2.2 Additional metrics

Since traceroute is the most used tool by network operators, examining how often our isolation results differ than those given by a simple traceroute provides insight into the marginal benefit

Category	Number of outage events
No path change, next historical hop down	195 (25.0%)
No path change, next historical hop reachable	229 (29.4%)
Path change	163 (20.9%)
Reachability horizon confined to one AS	10 (1.2%)
Reachability horizon spans multiple ASes	22 (2.8%)
Broken provider link for multi-homed stub AS	37 (4.7%)
Inaccurate or missing historical reverse traceroute	128 (16.4%)

Table 1: Counts of outage classifications. For the purposes of this table, categories are not overlapping, and no outages were unclassified.

of the extra machinery provided by *LIFEGUARD*. Note with traceroute alone, operators do not have a way to infer the direction of the failure, and thus cannot tell whether the last hop of the traceroute is indeed adjacent to the failure. For the 663 filtered cases where a normal traceroute was successfully issued, 201 (30%) of the suspected failures given by our system were not adjacent to the last traceroute hop. 152 (23%) were in a different autonomous system. Recall that for forward outages, our isolation algorithm identifies the last pingable hop along the traceroute as the suspected failure. As we describe later on, we found 28% of failures to be on the forward path. These results indicate that a normal traceroute provides correct isolation information for an additional 42% outages, either bidirectional or on the reverse path. For the remaining 30% of outages, the extra machinery implemented in *LIFEGUARD* is necessary for accurate isolation. Note that the last hop of the traceroute may differ from the suspected failure either because the failure is on the reverse path, or because the spoofed traceroute yielded more hops.

For optimal accuracy, it is essential for our system to have a full set of historical and on-demand measurements. Over the course of 4 months, our system did not successfully measure a spoofed traceroute in 3% of outages, a spoofed reverse traceroute in 46%, did not have a working historical forward traceroute in 10%, and did not have a historical reverse traceroute from the destination in 53%. Unsuccessful spoofed reverse traceroutes are fairly common due to progress constraints we place on measurements: if a reverse traceroute takes more than 40 minutes to issue, or requires more than 3 consecutive symmetry assumptions, we kill it and return a failed query. We believe the latter constraint is justified, since inaccurate reverse traceroutes are of little value to our isolation algorithm. Missing historical reverse traceroutes are largely due to a bug discovered in our system on May 24th, 2011, causing some queries to the historical data store to incorrectly return empty results. Historical reverse traceroutes may also have been missing if the reverse path had not been refreshed in a sufficiently long time, or the reverse path atlas was not communicating properly with our centralized controller.

4.3 Scalability

Given *LIFEGUARD*'s current measurement overhead, could we feasibly scale to cover the entire Internet? There are two related issues here: the probing capacity of vantage points, and the time for results to be returned.

From a small experiment, we found that PlanetLab nodes with average load are able to issue and receive ˜1000 ping probes in one minute. This means that with a period of 10 minutes, our system's ping monitors could feasibly monitor up to 10,000 targets. With more than 10,000 targets, the vantage points would be unable to monitor at a 10-minute granularity. Although we

Measurement	Average time (seconds)
Spoofed pings	13.92
Spoofed traceroute	86.60
Traceroute	32.85
Pings	7.81
Spoofed reverse traceroute	1281.44
Total measurement time for bidirectional and reverse path outages	141.18
Total measurement time for forward path outages	1336.02

Table 2: Average measurement times for each component

may increase this rate by using more threads or PlanetLab slices, we believe that our monitoring rate is ultimately constrained by speed of our nodes' NICs.

If we ignore the probing overhead of maintaining historical path atlases, fault isolation requires approximately 280 probes. Let y be the number of receivers with connectivity and z be the number of hops on historical paths. Then $2 * y$ spoofed pings (direction), $30 * y$ spoofed TTL-limited pings (spoofed traceroute), 30 normal TTL-limited pings (normal traceroute), ˜40 IP-options (spoofed reverse traceroute), and $(y + 1) * z$ pings (reachability inference) are issued in a single iteration of our fault isolation algorithm. A typical partial outage has 2 to 6 connected nodes, and 30 distinct historical hops. As we show later on, isolation for forward and bidirectional outages takes 140 seconds on average. Thus, if we again assume a maximum probing rate of 1000 probes per second, a vantage point in our system could feasibly isolate up to 15 outages in parallel at any one time without queuing measurements. Note that it is possible that issuing spoofed probes to all connected receivers may not be necessary; we plan to conduct a study in the near future on the marginal benefit of adding in additional receivers.

The historical path atlases represent the majority of our system's overhead. With ˜45,000 (source,destination) pairs including intermediate hops, we currently refresh our reverse paths approximately every four hours. This constitutes a considerable bottleneck for achieving our goal of global coverage. In ongoing work, we are investing techniques to improve the coverage and speed of reverse traceroute. It is worth noting that some paths may be refreshed much faster than four hours, and paths that take longer to measure do not impede the refresh rate of other paths. We discuss this issue further in Section 6.

To enable repair and circumvention of reachability problems, it is crucial for *LIFEGUARD* to return its results in a timely manner. Table 2 breaks down the average time for each measurement type. Although most measurements can be parallelized, spoofed reverse traceroutes require substantially more time than the other measurements. This is partly because spoofed reverse traceroutes are often forced to make symmetry assumptions after all other IP-option measurement attempts have failed. To help minimize this overhead, we withhold spoofed reverse traceroute requests if the failure is either bi-directional or on the reverse path. Exploring alternate progress requirements for spoofed reverse traceroutes, such as capping the total number of symmetry assumptions, has the potential to further improve this time.

4.4 Relevance of Outage Reports

Over the course of four months, our system detected 27,291 partial outages passing the initial set of filtering heuristics. After issuing isolation measurements, 778 (2.85%) passed the final set of filters. This translates into roughly 5 outage reports per day. We found these filtering heuristics to

be effective in reducing the number of uninteresting or inconclusive outage reports.

To evaluate how well the reports generated by *LIFEGUARD* facilitate the avoidance and repair of outages, we examine the number of cases where we were able to identify alternate working paths. We note that even if all hops along a reverse path are pingable from the source, a reverse path with too many symmetry assumptions may not be valid. For this study we considered a reverse path to be valid if it required no more than 3 symmetry assumptions overall, and did not have any sequence of 2 or more symmetry assumptions in a row. First, we found that *LIFEGUARD* was able to measure the entire opposite direction for 265 (64%) of 413 uni-directional failures. The remaining 36% were forward path outages where the spoofed reverse traceroute was not measured successfully. Over all 778 filtered outages, we were able to identify alternate working paths in either direction for 383 (49%). The alternate paths consisted of 189 historical reverse paths, 42 historical forward paths, 197 measured forward paths, and 40 measured reverse paths.

4.5 Deployability

We have designed to *LIFEGUARD* to be easily deployable, such that it can be replicated by any domain with access to a handful of vantage points. Excluding the infrastructure required by the reverse traceroute system, *LIFEGUARD* currently employs 12 vantage points.

5 Outage Characterization

After demonstrating *LIFEGUARD's* effectiveness in achieving our design goals, we now present an analysis of the outages observed by *LIFEGUARD* over the four-month period starting February 12th 2011.

5.1 Case studies

We first present a few case studies as illustrations of the problems uncovered with *LIFEGUARD*'s diagnostic information. Later we provide a more general characterization of the outages we observed.

Forward path failure, multi-homed AS. One of *LIFEGUARD*'s vantage points in China found that the measured forward path towards a failed router in Romania followed the historical forward path, but stopped at a border router within AdNet Telecom, one of the destination's providers. We were able to successfully measure the entire reverse traceroute back from the destination, and found that the destination was routing return packets through one of its other providers, NTT. We later verified by hand that other vantage points with connectivity to the destination were using NTT rather than AdNet as an ingress into the destination AS, further supporting our hypothesis that the provider link to AdNet was down. If network operators were receiving alternate BGP advertisements through NTT, they could easily switch paths to route around the failure.

Reverse path failure, confined to one AS. *LIFEGUARD* found that a subset of vantage points were unable to reach 208.23.225.1, a GoDaddy router in Chicago. A normal traceorute from one of our vantage points in Korea stopped within Level3's San Jose PoP, but a spoofed forward traceroute yielded 7 more hops through Level3's Dallas and Chicago PoPs, proceeding all the way to the destination. This suggests a reverse failure on the paths from all hops beyond San Jose. We found with historical reverse traceroutes that the return path from the destination passed through Level3 in Chicago, and on through Level3's Denver PoP. Since Denver was pingable from the source, and all hops including the destination were pingable from at least one other vantage point, we suspect that the failure was an incoming link to Denver. Even if a path change occurred since

Dataset	Number of targets	Number of outages observed
CDN cache hosts	16 (4.4%)	85 (0.3%)
PlanetLab nodes	76 (21.1%)	17505 (64.4%)
Core PoPs	83 (23.1%)	1224 (4.5%)
Edge routers	185 (51.4%)	6867 (25.16%)

Table 3: Outages observed per dataset

Rank	Number of Located Failures
Tier 1	66..126 (8.4 - 16.1%)
Tier 2	244..408 (28.8 - 52.4%)
Tier 3	72..187 (9.2 - 24.0%)
Stub	94..193 (12.0 - 24.8%)

Table 4: Distribution of filtered outage locations across ISP tiers

the time the historical reverse traceroutes were issued, the information provided by *LIFEGUARD* seemed to indicate that the failure was confined to Level3.

Forward path change, historical path reachable. From one of our vantages points, we found that the last hop of a measured forward path towards an unresponsive target in New York was located in asianetcom.net (AS 12271). Interestingly, although the historical forward path passed through the same router, all hops except the destination along the historical path were pingable from the source. Moreover, *LIFEGUARD* issued a number of additional traceroutes to hops beyond the failure, all of which passed successfully through the last forward hop and exited asianetcom.net's network. Thus, we have evidence that asianetcom.net made an internal routing change from a working path to a failed path, and was not failing over properly to alternate working paths within their AS. The existence of other working paths within asianetcom.net's network suggests that other networks could bypass the failure by entering asianetcom.net through a different ingress.

Failure(s) correlated across multiple sources and destinations. A vantage point in Albany and a vantage point in Delaware observed simultaneous outages to three destinations: a core router in XO's Texas PoP, a core router in XO's Chicago PoP, and an edge router in Ohio. Both of the outages observed from Albany were identified as reverse path failures, and the outages from Delaware were identified as bi-directional and reverse. All reverse path failures were isolated to 207.88.14.197.ptr.us.xo.net, an XO router in Dallas, and the bidirectional failure was isolated to p5-2.IR1.Ashburn-VA.us.xo.net, an XO router in Virginia. Several operators subsequently posted to the Outages.org mailing list reporting problems with XO in multiple locations, including Los Angeles, San Diego, New York and New Jersey. This example demonstrates that failures in the core affected multiple paths, and suggests that there may be value to correlating failures across vantage points and destinations.

5.2 Location

Where do failures occur on the Internet? Here we consider how often outages are observed for each of our datasets, as well as the location and direction of failures.

Table 3 shows the number of unfiltered outages observed per dataset. Not surprisingly, relatively few outages were observed along paths to well-maintained CDN cache hosts and core routers.

Table 4 shows the distribution of filtered outage locations across ISP tiers. We consider only filtered outages here, since for the non-filtered cases the problem may have resolved itself by the time

the isolation measurements were complete. Note that we enumerate a range of possible locations. This is because the suspected failed link was found at the boundary between two ASes on historical paths, and it is well known that the IP addresses of border routers are often borrowed from the address space of their neighbor [19]. We found 325 (41%) such inter-domain failures, and 462 (59%) intra-domain failures. This matches closely with a related study by Feamster et. al [20], which found that 62% of failures were intra-domain.

We also consider the direction of failures detected by our system. *LIFEGUARD* found that 197 failures (25%) were on the reverse path, 216 (28%) were on the forward path and the remaining 375 (47%) outages were bi-directional. This is closely in line with the results given by Hubble [7], where 62% of outages were isolated to either the forward or reverse path.

5.3 Application-level connectivity

To evaluate whether ICMP ping losses are correlated with application-level connectivity problems, we periodically issued HTTP GET queries for a URL known to be hosted by the content distribution nodes described in Section 4.1. In addition to issuing GET queries to the 16 raw IP addresses in our ping target list, we also performed a DNS resolution for the URL every round, and issued GET queries to the IP addresses listed in the A record response. It should be the case that ping losses correspond to dropped GET queries for the hardcoded IP addresses. If none of the IP addresses in the A record responded to the query, we confirm the content-distribution network was unable to redirect clients around the outage. Queries were issued from the same nodes and at the same rate as ICMP pings, albeit from a separate process.

For the study, we filtered all outages where there weren't simultaneous HTTP and ping measurements. From those, we examined all ping outages and checked whether HTTP outages occurred at the same time for the same PoP. Further, we correlated this with the PoP that the VP was directed to through DNS.

Over the two months we issued HTTP requests, we observed 6215 ping outages for the content distribution nodes, and 364 (5.86%) simultaneous HTTP outages. Of these simultaneous outages, 18 (0.02%) were for the PoP currently . This data suggests that (i) the content-distribution network was quite reliable in terms of application-level connectivity, and (ii) it must have been the case that the content distribution nodes went through periods of availability where they were unresponsive pings.

It is a known concern that nodes may be running, yet periodically unresponsive to pings. This suggests that ping losses alone are not a reliable indicator of outages. Nonetheless, *LIFEGUARD*'s measurements from multiple vantage points and consideration of historic reachability information all give stronger evidence that the detected outages are actually affecting traffic.

5.4 Alternate paths

How do alternate working paths differ from broken paths? We consider a series of related questions here.

First, we examine the percentage of time where alternate working paths pass through the same AS as the failure. Surprisingly, out of 223 unidirectional failures where the working direction was successfully measured, the working direction passed through the same AS as the suspected failure for 189 (84%). Thus, when both the forward and reverse path traverse the same AS, the paths tend to traverse different routers within that AS. This suggests that even if network operators are not able to route around the responsible ISP, they may be able to find a working path to the destination by entering the ISP through a different ingress.

Rank	At least one reachable target (unique ASes)	No reachable target (unique ASes)
Tier 1	161 (10)	0 (0)
Tier 2	29 (7)	5 (3)
Tier 3	61 (12)	7 (2)
Stub	14 (2)	7 (4)

Table 5: Breakdown of prefix reachability within the destination AS.

Second, we performed a study to answer the following question: when there is a failure for a destination D, are other prefixes advertised by D's AS reachable from the source? For this study we maintained one pingable IP address for every prefix advertised by one of the ASes in our target set. Whenever an outage was detected for a destination D from a source S, we issued pings to these IP addresses from S and all vantage points with connectivity to D. We use the term "reachable" to mean that that the source can ping a target, and "pingable" to mean that the source cannot ping the target, but some other vantage can.

Our study spanned two days and 334 outages, affecting 51 unique destinations. For 284 of these outages (47 unique destinations), the destination was unreachable, but pingable. We found that of these 284 outages, 265 (42 unique destinations) had at least one reachable target in the AS. Moreover, of the 284 outages, we found that 188 (15 unique destinations) have all pingable targets reachable. Lastly, for 19 of the 284 outages, 19 (10 unique destinations) had no reachable targets in the AS. Table 5 breaks down these cases by AS rank. Although these numbers are too small to draw any real conclusions, it does seem that (as expected) larger ASes are more likely to have other reachable targets than smaller ASes.

6 Discussion

In this section we discuss lessons we learned from building *LIFEGUARD*, as well as proposed future work.

6.1 Lessons Learned

If we could give one piece of advice to someone building a similar system, it would be this: the abstract model of network behavior in your head almost never matches reality exactly. *LIFEGUARD* constantly surprised us with unexpected measurement results.

In a similar vein, we learned that it is not always possible to accurately isolate failures with heuristics alone; network failures can manifest themselves in many ways, and it's very difficult to encompass all behavior patterns with a single isolation algorithm. *LIFEGUARD* does make an effort to reduce noise in measurement results, e.g. by waiting and re-probing if a path change is detected before isolation measurements have finished. However, we believe that for some types of outages, especially those found in the wide-area where network operators' do not have complete access to the internal state of routers, making educated guesses and presenting diagnosis information as clearly as possible to humans is the best a computer system can do. That said, as long as the failure can be isolated to the correct AS, automated remediation systems built on top of *LIFEGUARD* may still be feasible.

6.2 Future Work

Our work on *LIFEGUARD* is far from complete. Our long-term goal for *LIFEGUARD* is to see it deployed at Internet-scale and put into real use by network operators. Several challenges and avenues of investigation need to be addressed to achieve this goal:

- The coverage of our system could potentially be improved by choosing targets more intelligently. To evaluate the efficacy of our current target selection strategy, we plan to examine the correlation between outages in centralized PoPs and outages seen at edge routers beyond those PoPs. If it is the case that outages are well correlated, then we will have shown that wide coverage of outages can be achieved by only monitoring a select set of routers.

- The accuracy of our system is not yet well evaluated. We hope to corroborate with network operators in the future to gain confidence in our approach.

- Feldmann et al. [21] correlated outages across time, vantage points, and destinations in order to gain more accurate isolation of network faults. Although our system generates all of the necessary data to do the same, we currently only consider a single viewpoint. We are excited to pursue this line of investigation. Basically, our idea is to assume a single failure in the network. Then, we'll generate a "hypothesis space" consisting of all routers along historical and measured paths. Finally, we'll widdle down the size of this hypothesis space by identifying routers that are clearly not the point of failure. We would like to be able to reason about the following cases:

  1. Multiple vantage points observing an outage for a single destination.
  2. A single vantage points observing simultaneous outages for multiple destinations.
  3. Multiple vantage points observing simultaneous outages for multiple destinations.

  In addition, it might be fruitful to programmatically track the behavior of outages over time.

- Our isolation algorithm assumes only one network fault. We would like to verify whether this assumption holds in practice.

- When identifying alternate working paths, we currently only verify that a set of working *links* exists in the network. In practice, it may not be the case that network operators have access to alternate BGP routes towards the destination. We would like evaluate how often alternate paths are in fact viable.

- A number of optimizations can still be made to the reverse traceroute system. Improving the scalability of reverse traceroute will be essential if we are to transform *LIFEGUARD* into a full-blown Internet monitoring system.

- We were unable to perform a number of measurements studies in the interest of time. It would be valuable to further examine the characteristics of the outages we observed, e.g. failure-induced path changes.

In addition to seeing our system widely deployed, we would also like to investigate applications which could potentially benefit from isolation information. For example, we are currently exploring *PAMELA*, Poisoning ASes for Machiavellian Event Location Avoidance, as a methodology for routing around failures once they have been detected and located. We are also interested in applying our isolation techniques to more general performance problems.

7 Related Work

We group related work into three categories: active probing, passive BGP monitoring, and failure avoidance.

Active probing: Several studies have characterized reachability problems with active measurements. In influential early work, Paxson used traceroutes to investigate properties of Internet routing, including stability, asymmetry, and failures [15]. Govindan and Paxson [22] later used source spoofing to estimate router processing time over paths they also probed without spoofing; we use it to send packets over paths we otherwise would be unable to probe.

Feamster et. al. [23] measured Internet failures between vantages points using periodic pings and triggered traceroutes. They consider the location of a failure to be the last reachable hop in traceroute. They also used one-way ping to distinguish forward and reverse failures. Our work generalizes this approach by detecting and isolating outages in the wide-area without requiring control over the destination. We also handle reverse path outages in a more systematic way.

PlanetSeer's [8] measurement coordination closely resembles *LIFEGUARD*; a problem detected at one vantage point triggers measurements at others in order to further isolate the location of a failure. However, PlanetSeer uses remote vantage points only to identify cases of partial reachability, with all further probes and analysis focusing only on the local measurements. Our system uses more sophisticated measurement coordination.

We draw much of the inspiration for *LIFEGUARD* from our previous work. We motivate the design of *LIFEGUARD* with findings from Hubble [7], a system for monitoring reachability problems at Internet-scale. *LIFEGUARD* also makes heavy use of reverse traceroute [9] for maintaining historical path atlases and measuring working paths. *LIFEGUARD* can be seen as an application of these results; while the focus of Hubble was studying gross properties of Internet outages and providing rudimentary root-cause information, *LIFEGUARD* aims to facilitate repair and circumvention of specific reachability problems by isolating outages at a much finer level of granularity and providing auxiliary information such as working alternate paths.

Passive BGP monitoring: Several papers have studied the use of BGP monitoring for detecting and studying reachability problems. Labovitz et al. [24] found that routers may take tens of minutes to converge after a failure, and that end-to-end disconnectivity accompanies this delayed convergence. Mahajan et al. [25] showed that router misconfigurations could be detected with BGP feeds. Ave et al. [26] analyzed the temporal properties of failures, such as mean time to fail, mean time to repair and failure duration. They found that 40% of the failures are repaired in 10 minutes and 60% of them are resolved in 30 minutes. Caesar et al. [5] proposed techniques to analyze routing changes and infer why they happen. Feldman et al. [21] were able to correlate updates across time, across vantage points, and across prefixes by assuming that the failure lies either on the old best path or the new best path; they can pinpoint the likely cause of a BGP update to one or two ASes. Together, these studies developed techniques to reverse-engineer BGP behavior, visible through feeds, to identify network anomalies. However, there are limits to this approach. Though it is possible to infer reachability problems by passive monitoring [6], often times the presence of a BGP path does not preclude reachability problems and performance bottlenecks. Further, BGP data is at a coarse, AS-level granularity, limiting diagnosis.

Failure avoidance: Overlay detouring systems seek to avoid partial reachability problems by routing traffic through intermediary peers whenever failures are detected. For example, RON [27] nodes are able to react to failures on the order of seconds by monitoring all paths to all nodes in the system and routing through peers identified to have working paths. Other systems such as one-hop source routing [28] provide a more scalable detouring solution by routing traffic through a randomly chosen intermediary. Our work takes a complementary approach, providing

troubleshooting information for debugging and repairing the failures themselves, on the order of minutes rather than seconds.

Our system differs from all of the previously mentioned work in several ways. First, *LIFEGUARD* isolates wide-area network faults at a much finer level of granularity than was previously achievable, especially with respect to reverse path failures. It accomplishes this by leveraging historical path atlases, reverse path information gathered with the reverse traceroute system, and novel tools such as spoofed forward traceroute. Second, it facilitates repair and circumvention of reachability problems by providing auxiliary information such as possible alternate working paths.

8 Conclusion

In this thesis we argued that proactive isolation of wide-area network faults has the potential to improve the reliability of the Internet. We began by demonstrating that network outages are prevalent, difficult to isolate and repair, and highly problematic. We then presented our prototype system, *LIFEGUARD*, as a proof-by-example that failures can be isolated at an IP-link link granularity, regardless of direction. Finally, we argued that the measurement techniques and algorithms used in *LIFEGUARD* improve significantly on currently available diagnosis tools, allowing network operators to repair and remediate outages more quickly and effectively.

Last but not least, we found that network measurement systems are fantastically fun!

Acknowledgements

I would like to thank Ethan Katz-Bassett, Dave Choffnes, Italo Cunha, Arvind Krishnamurthy, Jeff Rasley, Allison Obourn, Justine Sherry, and Tom Anderson for all their contributions and support.

References

[1] http://isotf.org/mailman/listinfo/outages, "Outages mailing list."

[2] http://www.merit.edu/mail.archives/nanog/, "North American Network Operators Group mailing list."

[3] J. P. John, E. Katz-Bassett, and T. Anderson, "Consensus routing : The internet as a distributed system," in *NSDI*, 2008.

[4] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating internet routing instabilities," *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '04*, p. 205, 2004.

[5] M. Caesar, L. Subramanian, and R. H. Katz, "Root cause analysis of Internet routing dynamics," tech. rep., Univ. of California, Berkeley, 2003.

[6] C. Labovitz, A. Ahuja, and M. Bailey, "Shining Light on Dark Address Space."

[7] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson, "Studying black holes in the Internet with Hubble," in *NSDI*, 2008.

[8] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, "Planetseer: internet path failure monitoring and characterization in wide-area services," in *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, (Berkeley, CA, USA), pp. 12–12, USENIX Association, 2004.

[9] E. Katz-Bassett, H. V. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, A. Krishnamurthy, and T. Anderson, "Reverse traceroute," in *NSDI*, 2010.

[10] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *OSDI*, 2006.

[11] "ec2." `http://aws.amazon.com/ec2/`.

[12] "Georgian woman cuts off web access to whole of armenia." http://www.guardian.co.uk/world/2011/apr/06/georgian-woman-cuts-web-access.

[13] "Ucla as topology dataset." `http://irl.cs.ucla.edu/topology/`.

[14] http://www.routeviews.org/, "RouteViews."

[15] V. Paxson, "End-to-end routing behavior in the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 26, pp. 25–38, Oct. 1996.

[16] Y. Zhang, V. Paxson, and S. Shenker, "The stationarity of internet path properties: Routing, loss, and throughput," tech. rep., In ACIRI Technical Report, 2000.

[17] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 681–692, December 2001.

[18] "Planetlab." `http://www.planetlab.org`.

[19] H. Chang, S. Jamin, and W. Willinger, "Inferring AS-level internet topology from router-level path traces," in *SPIE*, 2001.

[20] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek, "Measuring the effects of Internet path faults on reactive routing," in *SIGMETRICS*, 2003.

[21] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating Internet routing instabilities," in *SIGCOMM*, 2004.

[22] R. Govindan and V. Paxson, "Estimating router icmp generation delays," in *Proceedings of the Passive and Active Measurement (PAM) workshop*, 2002.

[23] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek, "Measuring the effects of internet path faults on reactive routing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, p. 126, June 2003.

[24] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," in *SIGCOMM*, 2000.

[25] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *SIGCOMM*, 2002.

[26] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of internet stability and wide-area backbone failures," in *in Proc. International Symposium on Fault-Tolerant Computing*, 1998.

[27] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, (New York, NY, USA), pp. 131–145, ACM, 2001.

[28] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall, "Improving the reliability of internet paths with one-hop source routing," in *In OSDI*, pp. 183–198, 2004.