Quantifying Violations of Destination-based Forwarding on the Internet

Tobias Flach
University of Southern California
Los Angeles, CA 90089, USA
flach@usc.edu

Ethan Katz-Bassett
University of Southern California
Los Angeles, CA 90089, USA
ethan.kb@usc.edu

Ramesh Govindan
University of Southern California
Los Angeles, CA 90089, USA
ramesh@usc.edu

ABSTRACT

Initially, packet forwarding in the Internet was destination-based – that is, a router would forward all packets with the same destination address to the same next hop. In this paper, we use active probing methods to quantify and characterize deviations from destination-based forwarding in today's Internet. From over a quarter million probes, we analyze the forwarding behavior of almost 40,000 intermediate routers. We find that, for 29% of the targeted routers, the router forwards traffic going to a single destination via different next hops, and 1.3% of the routers even select next hops in different ASes. Load balancers are unlikely to explain these AS-level variations, and in fact we uncover causes including routers inside MPLS tunnels that otherwise employ default routes. We also find that these violations can significantly affect the results of measurement tools that rely on destination-based forwarding, and we discuss some ideas for making these tools more robust against these violations.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management*; C.2.6 [**Computer-Communication Networks**]: Internetworking—*Routers*

Keywords

Routing, Measurements, Forwarding, Ping, Traceroute

1. INTRODUCTION

Historically, most IP routers determined the next hop of a packet based solely on the destination of that packet [8]. As the routing fabric has evolved, networks have deployed several mechanisms that violate destination-based forwarding. Many Internet routers employ load balancing, with the majority of load balancers splitting traffic based on the flow identifier [3]. Tunneling using encapsulation (e.g., MPLS label switching or IP-in-IP encapsulation) can produce different routes than destination-based forwarding, as routers in the tunnel forward packets based on a label or the IP address of the tunnel endpoint. Finally, policy-based forwarding mechanisms may determine next hops using policies specified on arbitrary header fields to realize, for example, differentiated service.

Despite the existence of such mechanisms, a number of measurement systems still assume destination-based forwarding [6, 10, 15]. These systems are agnostic to whether routers forward based on destination or based on, say, labels. However, their correctness depends on the resulting routes being observably destination-based, in that all packets from a router to a destination follow the same path.

For example, reverse traceroute assumes forwarding is destination-based in order to stitch together a path out of segments of multiple paths to a common destination [10]. This stitching could yield false paths if routers forward based on factors other than just the destination address.

As another example, Doubletree attempts to minimize the number of probes required for topology discovery [6]. It uses hop-by-hop probing, like traceroute, to find (partial) paths between source-destination pairs, but stops probing for a pair if it observes an interface which was encountered earlier while probing on a different path to the same destination. In that case, Doubletree assumes that the remainder of the path is the same for both pairs. This may not hold if forwarding is not strictly destination-based.

Figure 1 illustrates how violations of destination-based forwarding can significantly distort path measurements. This figure shows the result of two probes sent to a PlanetLab node at the University of Auckland, New Zealand. We sent the first probe from a PlanetLab host at the University of Texas in Arlington, and we injected the second probe at one of the intermediate nodes observed on the first probe's path. The two probes take completely different AS-level paths because we injected the second probe at a router inside an MPLS tunnel. Because the injected packet lacked an MPLS label, the router forwarded it via a default route [14].

In this paper, we make two contributions. First, we devise a general methodology to detect destination-based forwarding violations; prior work (e.g., [4]) has described methodologies tailored towards specific kinds of violations like load balancing. Combining and adapting two existing probing techniques [7, 10], our technique uses source-spoofed IP *record-route* pings to quantify violations of destination-based forwarding. By injecting packets in routers in the middle of a source-destination path, we are able to detect violations caused by MPLS tunnels. In addition, these injected packets have different contents, so they can exercise load balancers or certain violations due to policy-based forwarding.

Our second contribution is an analysis of three research questions: How often is destination-based forwarding violated (Section 3)? What are the causes of these violations (Section 3 & 4)? What effect do these violations have on the accuracy of mea-

IMC'12, November 14–16, 2012, Boston, Massachusetts, USA.

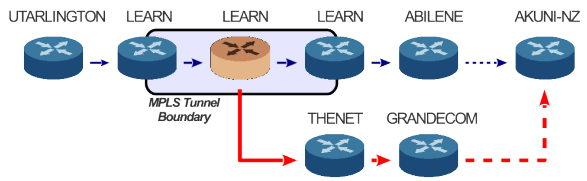

Figure 1: Observed partial paths for two probes to the same New Zealand destination. The first probe (thin/blue arrows) traverses the LEARN network via an MPLS tunnel. We inject the second probe (thick/red arrows) at the colored router inside the tunnel. The router forwards it based on the IP table entry for the destination. Each node is labeled with its AS. Dashed arrows indicate elided paths.

surement systems that assume the prevalence of destination-based forwarding (Section 5)? To answer these questions, we employ our probing methods on PlanetLab and conduct large-scale experiments, measuring paths between PlanetLab hosts and from PlanetLab to the broader Internet. We measure over 250,000 paths spanning over 3,700 ASes and find that a *non-negligible fraction (4.4%) of probed ASes exhibit violations that cause AS-level differences in paths*. Moreover, we find that violations of destination-based forwarding can significantly skew the results obtained by tools like reverse traceroute and Doubletree. We also find that a tool like reverse traceroute can easily be made more robust to such violations, but it is less obvious how to make Doubletree more robust.

2. MEASUREMENT METHODOLOGY

In this paper, we use active probing to find routes that observably violate destination-based forwarding. We first measure a path $h_1, h_2, \ldots h_n$ from a PlanetLab source to a destination D. Then, for each router h_i on that path, we want h_i to send a packet to D, to test whether it forwards to the same next hop h_{i+1} observed in our original path. By forcing h_i to send the packet – that is, by changing the point where the packet is injected into the network[1] – we guarantee that it does not route based on encapsulation originated by another router, thus giving us visibility inside tunnels. Since we do not have direct access to h_i, we have to induce it into sending a packet to D, which we can do by sending it an ICMP Echo Request and measuring the path that h_i's response uses.

While traceroute cannot measure this path, an IP Record Route ping (henceforth *RR ping*) can [10]. An RR ping is an ICMP Echo Request with the Record Route (RR) IP option enabled. The IP standard requires that, if a packet has the RR option set, routers record the IP address of one of their interfaces into the packet header. However, the RR field only has space for nine entries, and some routers do not honor the standard. Thus, only the first nine routers that implement the RR option will record one of their interfaces, and later routers will not overwrite these entries. A router that receives an RR ping will copy the RR field into the ICMP Echo Response, and hops on the reverse path will fill the remaining slots. Many routers support the RR option [17], and the global distribution of PlanetLab allows probes to reach many IP addresses within the nine-hop limit [9]. Nevertheless we acknowledge that the limitations of RR might introduce some bias to our data.

Baseline Experiments: Our experiments use PlanetLab nodes

[1]In this study, we only modified the point of injection (and therefore the source address of the packet), to keep the methodology concise. Thus, we do not detect violations caused by differing port numbers, protocol types, etc.

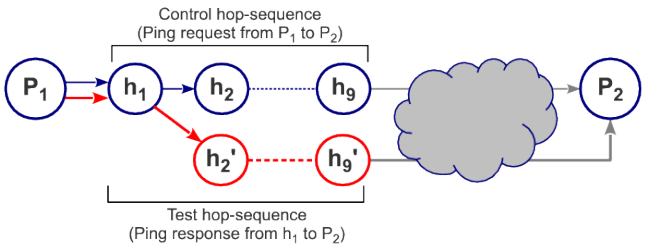

Figure 2: Example for two comparable paths to the same destination P_2. The thin/blue path (top) is traversed by an RR ping request sent directly to P_2. The thick/red path (bottom) is traversed by a spoofed RR ping request sent to h_1 and the response forwarded to P_2. Note that only the first nine hops can be recorded in either case, thus the remainder of the path to the destination is unknown.

to send RR pings and receive the responses. To describe our methodology, we discuss our baseline experiments conducted between source-destination pairs $\langle P_1, P_2 \rangle$ both on PlanetLab, resulting in a data set we label *Set PL/PL* (PlanetLab-to-PlanetLab). Later, we discuss extensions to this methodology.

For each such pair, P_1 first sends an RR ping request to another PlanetLab host P_2. Depending on the hop distance between the pair, the RR field in the response that P_1 receives will include all or part of the forward path to P_2 and may include all or part of the reverse path back to P_1. The hops on the *forward* path of this probe constitute what we call a *control hop-sequence*.

To detect violations to destination-based forwarding, we compare this hop sequence with sequences generated by sending *test* probes as follows (Figure 2). P_1 sends an RR ping to each forward path hop h_i, but *spoofs the source address as* P_2. Spoofing ensures that the response traverses the path between h_i and P_2. The response records hops in this path, when header space permits, and we use these hops to form what we call a *test hop-sequence*. This technique is strongly influenced by the methodology of reverse traceroute [10] and a prior technique used to estimate ICMP generation delays [7].

Thus, while the control probe is routed *through* the forward hops, we inject the i-th test probe *at* h_i. If forwarding were strictly destination-based, the control probe and the response to the test probe would follow the same path from h_i to P_2 because they are both destined to P_2, even though they have different source addresses (the response has h_i as the source address). On the other hand, if the next hop h'_{i+1} in the test hop-sequence is different from the next hop in the control hop-sequence, the hop h_i violates destination-based forwarding. In this case, we say that a *path fork* occurred at router h_i and that h_i satisfies the *fork condition*. However, the converse is not true: h_i may violate destination-based forwarding, but this violation may not always be visible in the way packets are forwarded at h_i.

Since a fork condition may be caused by a routing change between the control and the test probes, we repeat the pair of probes to ensure that the fork condition is reproducible. If a hop h_i reliably satisfies the fork condition, it is said to be a *violator*. Note that a violator need not violate destination-based forwarding for all traffic through it; our definition classifies the router as a violator if it exhibits the fork condition for at least one source-destination pair.

Extended Experiment Sets: As seen in the *Set PL/PL* part of Figure 3, only the first eight routers on a path between two PlanetLab nodes can be classified, and the source node must support spoofing. To target routers that are not observed on paths between

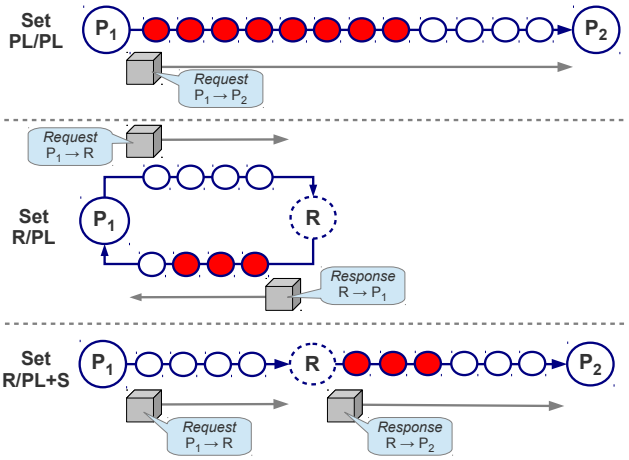

Figure 3: Overview of the coverage of nodes by each of the experiment sets and the ping packets associated with it. Violations can only be detected in filled nodes due to the RR hop limit.

PlanetLab nodes, we designed two additional sets of experiments. In *Set R/PL* (Router-to-PlanetLab) we send an RR ping from a PlanetLab node P_1 to a node R and use the hops visited by the response along the *reverse* path back to P_1 as the control hop-sequence. Then, P_1 sends an RR ping to each such hop as the test probe, again selecting the hops visited by the response as the test hop-sequence. Since we designed this approach to analyze paths back to the source of the ping, it has the advantage of not requiring spoofing. However, it has the drawback that R can be at most seven hops away from the source to leave space for at least two reverse hops in the RR field. We inject the test probe at the first reverse hop and use the second reverse hop to test the fork condition. The *Set R/PL* part of Figure 3 illustrates which hops this methodology can test. To determine which nodes are suitable choices for R for each PlanetLab node, we had the node send an RR ping to the .1 address in each routable /24 prefix and retained those that were responsive and within range.

Our third set, *Set R/PL+S* (Router-to-PlanetLab with spoofing), uses source spoofing to expand the coverage of the second set, as seen in Figure 3. Each PlanetLab node P_1 probes each nearby router R that it probed in R/PL, but P_1 sends multiple RR pings to R, spoofing as each of the other PlanetLab nodes in turn. This technique measures the (partial) path from R to a PlanetLab node P_2 even if P_2 is not near R. We use the hops $h_1, ..., h_m$ traversed by the response from R to P_2 as the control hop-sequence. We produce the test hop-sequences by sending probes from P_1 to $h_1, ..., h_m$ directly and extracting the hops visited by the response.

Establishing Fork Causality: In addition to determining violators, we also want to establish causality for forks. We explore two causes, load balancing and MPLS tunnels. Attributing violations to other causes is left for future work.

To characterize a violation as induced by load balancing, we employ two techniques. First, we measure the paths between our source-destination pairs with Paris traceroute, a traceroute variant that measures all load balancing paths to the destination [4]. Since traceroute and RR may record different IP addresses for a given router [17], we align the Paris traceroutes with our RR pings using IP alias data [12]. If a violating node is an alias for a load balancer seen by Paris traceroute on the same route, we classify the violation as being caused by load balancing. Since the alias data is likely incomplete, we use a second technique to attribute additional load balancing. Many routers act as packet-based load balancers for packets with IP options, including RR pings [17]. We can pinpoint these nodes by recording the control hop-sequence for a source-destination pair multiple times; we do this by issuing RR pings repeatedly. To find all possible paths with high probability we send 100 probes for each pair, and then compute the confidence interval for observing all next hops [4]. To avoid misclassifications in our analysis, we only used measurements with at least 99% confidence. Our results indicate that the 100 probes we sent per pair were sufficient to reach this confidence threshold in almost all cases. We mark a node as a load balancer if we observed at least two different next hops after that node on measurements to a single destination.

If a router inside an MPLS tunnel forwards the control probe based on a tunnel label, rather than based on the destination, it may choose a different route when it generates and sends the test probe towards the destination. To detect violations correlated with MPLS tunneling, that is cases where the violating node is observed in an MPLS tunnel, we use two detection mechanisms.

First, we rely on traceroute measurements which exhibit MPLS labels for nodes in *explicit MPLS tunnels* [5]. However, up to 50% of the tunnels have configurations which prevent this means of detection. That is, either the nodes do not appear in traceroute measurements at all (opaque or invisible tunnels) or their MPLS labels are obfuscated (implicit tunnels) [5, 18]. As such violations caused by them remain unclassified.

Second, we enhance our measurements to find possible instances of *default routing* regardless of tunnel visibility. Some routers in the middle of tunnels do not carry full routing tables and rely on tunnels to forward packets routed through them [14]. A router configured in this manner will use a default route to forward any packet it sources, and so all packets sent from the router will have the same next hop, regardless of the destination. To detect evidence suggesting such behavior, we sent a number of RR pings to a router, each using a different PlanetLab node as the spoofed source, thus causing the router to send responses to each of those PlanetLab destinations. If all these responses have the same first hop, regardless of the destination, and this hop differs from the next hop we observe on a route *through* the router, we assume the route through the router was traversing a tunnel.

Violation causes such as policy-based forwarding are harder to establish, and we have left these to future work.

3. RESULTS

Analysis methodology: We recorded all experimental datasets between December 16, 2011 and March 12, 2012. Before we conducted each experiment, we compiled a list of suitable PlanetLab nodes by selecting at most one active node per AS. For sets PL/PL and R/PL+S we also removed nodes from the sender set which employ spoofing filters since these experiments use spoofed source addresses in some probes. Since set PL/PL only investigates direct routes between PlanetLab nodes, we encountered fewer than 2,000 unique IPs on the paths between these nodes; sets R/PL and R/PL+S encountered about 25,000 IPs each when targeting arbitrary destinations within seven hops distance. Overall, we recorded more than 260,000 traces using 179 PlanetLab sites, 65 of which do not filter source-spoofed packets.

After we recorded the measurements, we compared each test hop-sequence with its corresponding control hop-sequence. We analyzed the data at two levels of granularity: IP addresses and AS numbers. To map an IP address stored in the RR field to its cor-

Traces recorded	262,034
Involved PlanetLab nodes	179
IP addresses targeted	39,699
IP addresses violating (IP forks)	11,487 (28.9%)
IP addresses violating (AS forks)	505 (1.3%)
ASes targeted	3,777
ASes w/ viol. IP addr. (IP forks)	669 (17.7%)
ASes w/ viol. IP addr. (AS forks)	165 (4.4%)

Table 1: Overview of combined results from all experiment sets (PL/PL, R/PL, R/PL+S)

Type	Freq. (IP)	Freq. (AS)
Non-violating	74.8%	97.8%
Load balancing	16.1%	0.0%
Explicit MPLS tunnel	0.4%	0.0%
Default routing	0.3%	0.3%
Unclassified	6.9%	1.7%
Invalid	1.5%	0.0%

Table 2: Node type frequencies in Set PL/PL for IP and AS forks

responding AS number, we used prefix-to-AS mappings provided by an iPlane dataset [15].

On the IP level, we classified a targeted node as a violating router if, for at least one pair of paths to a common destination, the next hop differed. In this case we observed an *IP fork*. If the next AS, different from the AS of the targeted node, differed between the control and test hop-sequences, we labeled it an *AS fork*.

Frequency of violations: Table 1 summarizes the experimental results. *Out of nearly 40,000 targeted IP addresses, 28.9% caused IP forks and 1.3% of the nodes caused an AS fork.* The existence of a non-trivial fraction of AS forks is interesting, and to our knowledge, has not been documented elsewhere. Our results also indicate that most of these forks occur at edge routers. It is not surprising that the fraction of nodes causing AS forks is significantly smaller than the number of nodes responsible for IP forks for the following reason. We expect the majority of IP forks to be the result of load balancing, and prior studies have shown that most load balancers only distribute traffic across multiple paths inside their own AS boundaries [3]. Thus, forks caused by load balancing usually converge before leaving the current AS and rarely translate to AS forks.

Causes of Violations: As discussed earlier, violations can have different causes, e.g., load balancing, MPLS tunneling, or other routing policies. In this section we classify forks by cause.

Table 2 provides a classification of the targeted nodes in set PL/PL (classifying violations in sets R/PL and R/PL+S is difficult since this requires traceroutes from the R router, which we could not obtain for these sets). The middle column shows statistics for all forks, whereas the right column only considers forks which also result in differing AS traversals. Table 3 shows statistics for the violation types observed on paths between source-destination pairs.

We determined *load balancing*, *Explicit MPLS tunnel*[2], and *default routing* violations as described in Section 2. We classified a node as *invalid* if we suspected routing instabilities or if the confidence of having observed all interfaces was below 99%. Any violation that did not fit in any of these categories is labeled as *unclassified*.

[2]Our traceroute data as well as results from an earlier study [18] indicate that about 25% of the paths contain an MPLS tunnel, but in our experiments most of them do not trigger a violation.

Type	Freq. (IP)	Freq. (AS)
Without violation	72.6%	89.9%
Load balancing	22.3%	0.0%
Explicit MPLS tunnel	0.2%	0.0%
Default routing	0.2%	0.2%
Unclassified	11.8%	10.0%

Table 3: Percentages of paths between source-destination pairs where a violation of particular type (or no violation at all) was observed in the PL/PL set (only considering first nine hops)

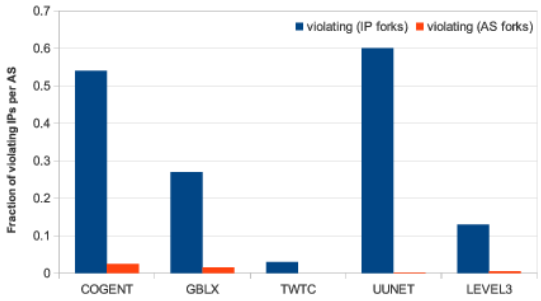

Figure 4: Fraction of violating IP addresses in an AS, that are violators for the most frequently targeted ASes (all experimental sets)

The distribution suggests that we can attribute the majority of IP forks to load balancing. However, for more than 6% of the cases we were unable to determine a violation cause. For many of these cases, we did not observe the violator on any recorded traceroute, e.g., due to ICMP error message filtering by some routers or missing IP address alias mappings. Thus, a fraction of these violations could still be the result of load balancing. However, *we did not observe any load balancers causing AS-level divergences*. All cases of *default routing caused AS forks*, as did a significant portion of the unclassified violations. Additional causes such as routing policies incorporating source addresses may be responsible for some path forks. In Section 4 we discuss some specific instances of the unclassified violations in more detail.

These frequencies may be a lower bound on the likelihood of observing violations on paths, because RR's nine hop limit restricts the coverage of our methodology. For example, as we discuss below, traceroutes from PlanetLab nodes to randomly selected destinations revealed that over 70% of the paths include at least one load-balancing node. However, many of these load balancers were in core networks that were located more than nine hops from our sources.

AS Characteristics: Next, we analyzed which ASes exhibited violations and with what frequency. Table 1 shows that we observed IP-level forks in 17.7% of the visited ASes and *AS forks in 4.4% of the ASes assessed*. However, the distribution of violating IP addresses across ASes is not uniform, with the number of IP addresses in an AS observed to cause a fork varying from 1% to over 70%. In Figure 4, we show the violation frequencies for the ASes in which we probed the most targets. While 55% of the targeted COGENT nodes are violating routers, less than 5% of the targeted TWTC addresses are responsible for a path fork. A likely explanation for these differences across ASes is the variability in AS engineering practices. For example, Sommers et al. found that COGENT is one of the top-10 ASes in terms of total MPLS tunnels

AS Name	Type	obs. IPs	viol. IPs
COGENT	Large ISP	3303	86
AR-TAST-LACNIC	Large ISP	62	18
SWISSCOM	Large ISP	600	13
GNAXNET-AS	Stub	35	12
GBLX	Tier-1	1119	12

Table 4: Statistics for ASes with the largest number of violating IP addresses (viol. IPs) causing AS forks (all experimental sets; obs. IPs describes the number of observed IPs in that AS in our measurements)

Diamond type	μ	σ	Median
Single AS	2.5	2.0	2
Multiple ASes	4.9	2.5	5

Table 5: Diamond length statistics from traces with violations from Set PL/PL

per AS, which likely contributes to the high number of violations in that AS [18].

We obtained similar results when considering only nodes responsible for AS forks. Table 4 shows the statistics for the ASes with the largest number of violating IP addresses and gives the UCLA Internet Topology Project's AS-type classification for the ASes [19]. As the table shows, AS forks can be found even in large ISPs and a Tier-1 provider, possibly due to the usage of MPLS and policy-based routing for traffic engineering.

Diamond lengths: In this section we analyze how much two paths deviate from each other once they traverse a violating router, before reconverging on their path to the common destination. The deviating path portion is termed a *diamond* [4]. To do so, we count the number of hops between the violating router and the first interface common to the two paths after the violating router. We refer to the common interface as the *merging router*, and we refer to the number of hops between the violating router and the merging router as the *diamond length*.[3] Note that due to the nine hop limit of the RR field, we may not observe the merging router. In obtaining these results, we omitted R/PL and R/PL+S datasets since we typically only observe a small number of hops past R, making it rarer to observe complete diamonds. In PL/PL cases when we do not observe the merge, we assume that the next (unrecorded) hop would be the merging router. While this likely results in an underestimation of the diamond lengths, the small number of such cases has a negligible effect on the overall outcome.

As shown in Table 5, diamonds where the forking and merging routers belong to the same AS have an average length of 2.5 nodes. This result roughly coincides with the average diamond length of load-balanced paths studied by Augustin et al. [3]. In contrast, *diamonds induced by AS forks are roughly twice as long on average*. When a path diverges at the AS level, it will usually traverse multiple nodes inside this AS before exiting to another AS where it could reconverge with the path traversed by the control probe.

4. CASE STUDIES

In Section 1 we briefly discussed a routing violation we found with our measurements (see Figure 1). In the example, the targeted IP is not globally routable. Normally, it only carries MPLS-encapsulated packets and local network management traffic. Because it normally only forwards traffic based on MPLS labels, it does not carry a full IP routing table. Instead, it has a default route pointing to the University of Texas System (THENet) that suffices to carry management traffic back to the network operators. In our case, a record route probe from a PlanetLab node at the University of Texas, Arlington (UTA), uncovered the router. Even though the IP was not globally routable, routes to it exist from the University of Texas system, and so our test probe from UTA reached the IP. However, the UTA node spoofed the test probe to claim it came from a different PlanetLab site. The target, lacking a non-default route to this other site, responded to the test probe via its default route. Despite the fact that the control and test probes visit completely different ASes, they do travel via similar locations. More concretely, in this example all probes follow routes via Arlington, Dallas, Houston and Los Angeles before leaving the RR scope.

There are other cases in our dataset showing similar behavior and we omit presenting them separately here.

However, many violations are hard to classify if no load balancing or MPLS labels are observed. This is especially true in cases where we have no corresponding traceroute data available for the measured path, i.e. when we analyze reverse paths. For example, below we describe a violation observed in the R/PL set. The control probes always observed the following hops:

```
1 dinet-gw.spb.citytelecom.ru
2 te4-1-20-adelaida.spb.cloud-ix.net
3 runnet-gw.msk.citytelecom.ru
4 msk-1-gw.runnet.ru
5 CORE-VL42.radio-msu.net
```

In contrast, the test probe response traversed a different path:

```
1 dinet-gw.spb.citytelecom.ru
2 * (vsevnet.ru)
3 border.vsevnet.ru
4 CORE-VL42.radio-msu.net
```

The paths fork in an AS in St. Petersburg, Russia and merge in a different AS in Moscow, but observe different intermediate ASes. Based on the measurements, we can only rule out load balancing as a possible cause for the violation. However, we cannot determine whether the point of injection or the different source address of the test probe traversing the reverse path is causing the forking router to forward the probes to different next hops.

5. IMPACT OF VIOLATIONS

In this section we discuss the possible impact of routing violations on two existing systems: Doubletree and reverse traceroute.

Impact on Doubletree: Doubletree reduces probing overhead in large-scale distributed traceroute measurements by stopping probing a path once it reaches a hop already observed on the path from another monitor to the destination [6]. Consequently, forking paths are not explored and remain invisible to the system (Figure 5).

To quantify the impact, we issued Paris traceroutes from over 170 PlanetLab nodes to 200 randomly chosen destinations.[4] In total, we acquired 30,911 traceroutes.

We found that 22,582 of these (73.1%) include at least one load balancing node. This differs from results by Augustin et al. who found that only 39% of their recorded traces contain a load balancer [3]. Since their study was published in 2007, we suspect that,

[3]If the two forks have different lengths we use the smaller value.

[4]We executed multiple trials with different destination sets to ensure that the results are not biased by the number of destinations selected. We synchronized the probing mechanism, such that traceroutes for the same destination were issued within a time frame of five minutes on all PlanetLab nodes, thus minimizing the probability of route changes.

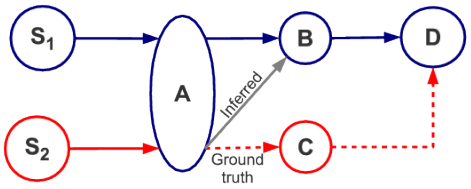

Figure 5: Scenario where Doubletree output deviates from the ground truth. Both paths from S_1 (blue edges) and S_2 (red edges) visit the common hop A. If Doubletree already observed the path $A \rightarrow B \rightarrow D$ it would not find the edge from A to C since the probing algorithm stops at the common node.

as part of the evolution of the Internet, more ASes have started to incorporate load balancing mechanisms into their topologies. This number is also larger than the results shown in Table 3. This discrepancy likely stems from the fact that traceroute provides data about all hops on a path, whereas RR has a 9 hop limit (see discussion of Table 3 in Section 3).

Of our recorded traces, 2997 (9.7%) visit a node with a violation unexplained by load balancing. Furthermore, 1260 traces (4.1%) encounter a node where traces towards the same destination forked into *different* ASes. These numbers are qualitatively consistent with Table 3. In that table, a little over 11% of the traces contained violations not explained by load balancing, with default routing and unknown causes usually resulting in AS forks.

We then determined the number of links that Doubletree would miss. Out of 49,272 observed links in the data, Doubletree would miss 11,222 links (22.8%) as a result of load balancers splitting traffic across multiple next hops. Furthermore, 601 links (1.2%) would remain unobserved due to other violations, with 150 unobserved links (0.3%) related to AS forks. Doubletree was shown to be unable to observe about 18% of links in an experiments [6]; our results are consistent with that number, but give some insight into the causes of unobserved links.

Impact on reverse traceroute: Reverse traceroute assumes forwarding is destination-based in order to incrementally piece together a path [10]. All the violations quantified in Section 3 would impact the accuracy of that technique; in particular, the statistics on diamond lengths give us a feel for the degree to which reverse traceroute can be affected.

Possible solutions: Both systems could employ Paris traceroute-like techniques to identify and incorporate load-balanced paths. However, Doubletree aims to reduce the number of probes required to establish a topology. Since Paris traceroute probes a node many times to ensure that all load balancing paths are explored, incorporating this technique to Doubletree would result in a significantly larger number of probes required to find the currently missing links. Thus, it is less clear how Doubletree should balance the tension between completeness and overhead reduction.

Additionally, systems could use the techniques in this paper to generate blacklists of violation candidates along a path, at the expense of additional probing overhead. Upon encountering these hops, systems could issue additional measurements to improve confidence in a result or flag it as suspect. For example, reverse traceroute could send additional spoofed RR pings, as described above, to find hops employing default routing that would otherwise distort results. Upon encountering these hops, reverse traceroute can easily adapt its IP timestamp-based technique to verify that hops discovered by its RR probes actually appear along the end-to-end path and not just on a default route. Similarly, in cases where an RR ping yields multiple reverse hops, reverse traceroute can then target each of those hops to check for consistent routes, as we did in this paper. Using a week of data from a reverse traceroute deployment monitoring routing failures [11], we analyzed probes sent for over 33,000 router-destination pairs. We found that 75% of them included multiple reverse hops, thus enabling the consistency check described above.

6. RELATED WORK

Systems such as reverse traceroute [10], Doubletree [6], and iPlane [15] assume destination-based forwarding. Other studies also make this assumption [2, 13, 16], but we omitted a detailed discussion of these due to space considerations.

Prior studies analyzed the prevalence of load balancing and MPLS deployments in the Internet. Augustin et al. describe measurements from fifteen RON nodes to over 68,000 destinations using Paris traceroute [3]. Their 2007 study found that about 39% of the paths between source-destination pairs contain at least one flow-based load balancer. In contrast, we have designed a generic mechanism to detect violations to destination based forwarding and demonstrate the existence of a non-trivial proportion of AS forks.

Using data from the CAIDA Archipelago project [1], Sommers et al. were able to detect MPLS tunnels based on traceroute data [18]. Besides extracting MPLS stacks from traceroute data, they inferred probable tunnels by comparing latency differences between consecutive hops. Their results indicate that about 25% of all paths in 2011 contained at least one tunnel. While their MPLS detection techniques are more advanced than ours, we did not incorporate their methodology due to the high number of false positives. Furthermore, our focus was on detecting violations to destination-based forwarding, and we use a different probing methodology because, as we have discussed above, not all MPLS tunnels trigger these violations.

Recently Donnet et al. demonstrated additional mechanisms to detect MPLS tunnels [5]. In their study they propose inference techniques for non-explicit tunnels based on using TTL signatures from traceroute and ping measurements. Incorporating their methods into our analysis would likely help to attribute some of the unclassified violations to non-explicit MPLS tunnels, and we leave this to future work.

7. CONCLUSION

We designed a probing methodology to uncover violations of destination-based forwarding. Our measurements reveal that an unexpectedly significant fraction of these violations cause forked paths to traverse different ASes as compared to the original path: 4.4% of targeted ASes and about 10.2% of paths exhibit such behavior. In general, these violations impact the accuracy of systems like Doubletree and reverse traceroute; the former can miss 22% of the links as a result of these violations. Finally, while it is conceptually easy to mitigate the impact of these violations in reverse traceroute by incorporating some of our mechanisms, it is less clear how to do so in Doubletree in a manner that also achieves one of that system's objectives, reducing probing overhead.

Acknowledgments

We would like to thank our shepherd, Peter Steenkiste, and the anonymous referees for their helpful comments on earlier versions of the paper. In addition we thank Byron Hicks from the Lonestar Education and Research Network (LEARN) for his insight on the causes of violations observed in his network. This work was funded in part by Google, Cisco, and the NSF (CNS-0121778 and CNS-0905568).

8. REFERENCES

[1] Archipelago IPv4 Routed /24 AS Links Dataset. http://www.caida.org/data/active/ipv4_routed_topology_aslinks_dataset.xml.

[2] B. Armbruster, J. C. Smith, and K. Park. A Packet Filter Placement Problem with Application to Defense Against Spoofed Denial of Service Attacks. *European Journal of Operational Research*, 176(2):1283–1292, 2007.

[3] B. Augustin, T. Friedman, and R. Teixeira. Measuring Load-balanced Paths in the Internet. In *IMC*, 2007.

[4] B. Augustin, T. Friedman, and R. Teixeira. Multipath tracing with Paris traceroute. In *E2EMON*, 2007.

[5] B. Donnet, M. Luckie, P. Mérindol, and J.-J. Pansiot. Revealing MPLS tunnels obscured from traceroute. *SIGCOMM Computer Communication Review*, 42(2):87–93, 2012.

[6] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Deployment of an Algorithm for Large-Scale Topology Discovery. *IEEE Journal on Selected Areas in Communications*, 24(12):2210–2220, 2006.

[7] R. Govindan and V. Paxson. Estimating router ICMP generation delays. In *PAM*, 2002.

[8] Internet Engineering Task Force. *RFC 1812: Requirements for IP Version 4 Routers*, June 1995.

[9] E. Katz-Bassett. *Systems for Improving Internet Availability and Performance*. PhD thesis, University of Washington, 2012.

[10] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. van Wesep, T. E. Anderson, and A. Krishnamurthy. Reverse Traceroute. In *NSDI*, 2010.

[11] E. Katz-Bassett, C. Scott, D. R. Choffnes, I. Cunha, V. Valancius, N. Feamster, H. V. Madhyastha, T. E. Anderson, and A. Krishnamurthy. LIFEGUARD: Practical repair of persistent route failures. In *SIGCOMM*, 2012.

[12] K. Keys, Y. Hyun, M. Luckie, and K. Claffy. Internet-Scale IPv4 Alias Resolution with MIDAR: System Architecture - Technical Report. Technical report, Cooperative Association for Internet Data Analysis (CAIDA), May 2011.

[13] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, and O. Lysne. Multiple routing configurations for fast IP network recovery. *IEEE/ACM Transactions on Networking*, 17(2):473–486, April 2009.

[14] Lonestar Education and Research Network (Austin, Texas): Private communication.

[15] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. E. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *OSDI*, 2006.

[16] A. Riedl and D. A. Schupke. Routing Optimization in IP networks Utilizing Additive and Concave Link Metrics. *IEEE/ACM Transactions on Networking*, 15(5):1136–1148, October 2007.

[17] R. Sherwood, A. Bender, and N. Spring. Discarte: A Disjunctive Internet Cartographer. In *SIGCOMM*, 2008.

[18] J. Sommers, P. Barford, and B. Eriksson. On the Prevalence and Characteristics of MPLS Deployments in the Open Internet. In *IMC*, 2011.

[19] UCLA Internet Topology Collection. http://irl.cs.ucla.edu/topology/.