PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-Area Services

Ming Zhang, Chi Zhang, Vivek Pai, Larry Peterson, and Randy Wang
Department of Computer Science
Princeton University

Abstract

Detecting network path anomalies generally requires examining large volumes of traffic data to find misbehavior. We observe that wide-area services, such as peer-to-peer systems and content distribution networks, exhibit large traffic volumes, spread over large numbers of geographically-dispersed endpoints. This makes them ideal candidates for observing wide-area network behavior. Specifically, we can combine passive monitoring of wide-area traffic to detect anomalous network behavior, with active probes from multiple nodes to quantify and characterize the scope of these anomalies.

This approach provides several advantages over other techniques: (1) we obtain more complete and finer-grained views of failures since the wide-area nodes already provide geographically diverse vantage points; (2) we incur limited additional measurement cost since most active probing is initiated when passive monitoring detects oddities; and (3) we detect failures at a much higher rate than other researchers have reported since the services provide large volumes of traffic to sample. This paper shows how to exploit this combination of wide-area traffic, passive monitoring, and active probing, to both understand path anomalies and to provide optimization opportunities for the host service.

1 Introduction

As the Internet grows and routing complexity increases, network-level instabilities are becoming more common. Among the problems causing end-to-end path failures are router misconfigurations [16], maintenance, power outages, and fiber cuts [15]. Inter-domain routers may take tens of minutes to reach a consistent view of the network topology after network failures, during which time end-to-end paths may experience outages, packet losses and delays [14]. These routing problems can affect performance and availability [1, 14], especially if they occur on commonly-used network paths. However, even determining the existence of such problems is nontrivial, since no central authority monitors all Internet paths.

Previously, researchers have used routing messages, such as BGP [16], OSPF [15] and IS-IS [12] update messages to identify inter-domain and intra-domain routing failures. This approach usually requires collecting routing updates from multiple vantage points, which may not be easily accessible for normal end users. Other researchers have relied on some form of distributed active probing, such as pings and traceroutes [1, 7, 19], to detect path anomalies from end hosts. These approaches monitor the paths between pairs of hosts by having them repeatedly probe each other. Because this approach requires cooperation from both source and destination hosts, these techniques measure only paths among a limited set of participating nodes.

We observe that there exist several *wide-area services* employing multiple geographically-distributed nodes to serve a large and dispersed client population. Examples of such services include Content Distribution Networks (CDNs), where the clients are distinct from the nodes providing the service, and Peer-to-Peer (P2P) systems, where the clients also participate in providing the service. In these kinds of systems, the large number of clients use a variety of network paths to communicate with the service, and are therefore likely to see any path instabilities that occur between them and the service nodes.

This scenario of geographically-distributed clients accessing a wide-area service can itself be used as a monitoring infrastructure, since the natural traffic generated by the service can reveal information about the network paths being used. By observing this traffic, we can passively detect odd behavior and then actively probe it to understand it in more detail. This approach produces less overhead than a purely active-probing based approach.

This monitoring can also provide direct benefit to the wide-area service hosting the measurement infrastructure. By characterizing failures, the wide-area service can mitigate their impact. For example, if the outbound path between a service node and a client suddenly fails, it may be possible to mask the failure by sending outbound traffic indirectly through an unaffected service node, using techniques such as overlay routing [1]. More flexible services may adapt their routing decisions, and have clients use service nodes that avoid the failure entirely. Finally, a history of failure may motivate placement decisions—a service may opt to place a service node within an ISP if intra-ISP paths are more reliable than paths between it and other ISPs.

This paper describes a monitoring system, PlanetSeer, that has been running on PlanetLab since February 2004. It passively monitors traffic between PlanetLab and thousands of clients to detect anomalous behavior, and then coordinates active probes from many PlanetLab sites to confirm the anomaly, characterize it, and determine its scope. We are able to confirm roughly 90,000 anomalies per month using this approach, which exceeds the rate of previous active-probing measurements by more than two orders of magnitude [7]. Furthermore, since we can monitor traffic initiated by clients outside PlanetLab, we are also able to detect anomalies beyond those seen by a purely active-probing approach.

In describing PlanetSeer, this paper makes three contributions. First, it describes the design of the passive and active monitoring techniques we employ, and presents the algorithms we use to analyze the failure information we collect. Second, it reports the results of running PlanetSeer over a three month period of time, including a characterization of the failures we see. Third, it discusses opportunities to exploit PlanetSeer diagnostics to improve the level of service received by end users.

2 Background

Although the Internet is designed to be self-healing, various problems can arise in the protocols, implementations [15], and configurations [16] to cause network instability. Even in the absence of such problems, routing updates can take time to propagate, so failures may be visible for minutes rather than seconds. Even though tools like *ping* and *traceroute* exist for diagnosing network problems, pinpointing failures and determining their origins is nontrivial for several reasons:

Network paths are often asymmetric. Paxson observed that 49% of node pairs have different forward and reverse paths which visit at least one different city [19]. Since *traceroute* only maps the forward path, it is hard to infer whether the forward or reverse path is at fault without cooperation from the destination. PlanetSeer combines passive monitoring results, path history information, and multi-point probing to isolate forward failures.

Failure origin may differ from failure appearance. Routing protocols, such as BGP and OSPF, may propagate failure information to divert traffic away from failed links. When a traceroute stops at a hop, it is often the case that the router has received a routing update to withdraw that path, leaving no route to the destination.

Failure durations are highly varied. Some failures, like routing loops, can last for days. Others may persist for less than a minute. This high variance makes it hard to diagnose failures and react in time.

Failure isolation requires broad coverage. Historically, few sites had enough network coverage to initiate enough traceroutes to identify all affected paths. The advent of public traceroute servers [24] provides some resources to help manually diagnose problems, and tools such as ScriptRoute [22] can help automate the process.

The advent of wide-coverage networking testbeds like PlanetLab has made it possible to deploy wide-area services on a platform with enough network coverage to also perform the probing on the same system. This approach addresses the problems listed above: (1) when clients initiate connections to the wide-area service, we obtain a forward path that we can monitor for anomalies; (2) PlanetLab nodes span a large number of diverse autonomous systems (ASes), providing reasonable network coverage to initiate probing; and (3) active probing can be launched as soon as problems are visible in the passively-monitored traffic, making it possible to catch even short-term anomalies that last only a few minutes.

While our focus is techniques for efficiently identifying and characterizing network anomalies, we must give some attention to the possibility of our host platform affecting our results. In particular, it has been recently observed that intra-PlanetLab paths may not be representative of the Internet [2], since these nodes are often hosted on research-oriented networks. Fortunately, by monitoring services with large client populations, we conveniently bypass this issue since most of the paths being monitored terminate outside of PlanetLab. By using geographically-dispersed clients connecting to a large number of PlanetLab nodes, we observe more than just intra-PlanetLab connections.

3 PlanetSeer Operation

This section describes our environment and our approach, including how we monitor traffic, identify potential path anomalies, and actively probe them.

3.1 Components

We currently use the CoDeeN Content Distribution Network [26] as our host wide-area service, since it attracts a reasonably large number of clients (7K-12K daily) and generates a reasonable traffic volume (100-200 GB/day, 5-7 million reqs/day). CoDeeN currently runs on 120

PlanetLab nodes in North America (out of 350 worldwide), but it attracts clients from around the world. CoDeeN acts as a large, cooperative cache, and it forwards requests between nodes. When it does not have a document cached, it gets the document from the content provider (also known as the origin server). As a result, in addition to the paths between CoDeeN and the clients, we also see intra-CoDeeN paths, and paths between CoDeeN and the origin servers.

PlanetSeer consists of a set of passive monitoring daemons (MonD) and active probing daemons (ProbeD). The MonDs run on all CoDeeN nodes, and watch for anomalous behavior in TCP traffic. The ProbeDs run on all PlanetLab nodes, including the CoDeeN nodes, and wait to be activated. When a MonD detects a possible anomaly, it sends a request to its local ProbeD. The local ProbeD then contacts ProbeDs on the other nodes to begin a coordinated planet-wide probe. The ProbeDs are organized into groups so that not all ProbeDs are involved in every distributed probe.

Currently, some aspects of PlanetSeer are manually configured, including the selection of nodes and the organization of ProbeD groups. Given the level of trust necessary to monitor traffic, we have not invested any effort to make the system self-organizing or open to untrusted nodes. While we believe that both goals may be possible, these are not the current focus of our research.

Note that none of our infrastructure is CoDeeN-specific or PlanetLab-specific, and we could easily monitor other services on other platforms. For PlanetSeer, the appeal of CoDeeN (and hence, PlanetLab) is its large and active client population. The only requirement we have is the ability to view packet headers for TCP traffic, and the ability to launch traceroutes. On PlanetLab, the use of *safe raw sockets* [3] mitigates some privacy issues – the PlanetSeer service only sees those packets that its hosting service (CoDeeN) generates. In other environments, we believe the use of superuser-configured in-kernel packet filters can achieve a similar effect.

In terms of resources, neither ProbeD nor MonD require much memory or CPU to run. The non-glibc portion of ProbeD has a 1MB memory footprint. The MonD processes have a memory consumption tied to the level of traffic activity, and is used to store flow tables, statistics, etc. In practice, we find that it requires roughly 1KB per simultaneous flow, but we have made no effort to optimize this consumption. The CPU usage of monitoring and probing is low, with only analysis requiring much CPU. Currently, analysis is done offline in a centralized location, but only so we can reliably archive the data. We could perform the analysis on-line if desired – currently, each anomaly requires a 20 second history to detect, one minute to issue and collect the probes, and less than 10ms of CPU time to analyze.

3.2 MonD Mechanics

MonD runs on all CoDeeN nodes and observes all incoming/outgoing TCP packets on each node using PlanetLab's tcpdump utility. It uses this information to generate path-level and flow-level statistics, which are then used for identifying possible anomalies in real-time.

Although flow-level information regarding TCP timeouts, retransmissions, and round-trip times (RTTs) already exists inside the kernel, this information is not easily exported by most operating systems. Since MonD runs as a user-level process, it instead derives this information by observing packet-level activity from tcpdump. It instead infers flow-level information—e.g., timeouts, retransmissions, and round trip times (RTTs)—from the sniffed packets, and aggregates information from flows on the same path to infer anomalies on that path.

MonD maintains path-level and flow-level information, with paths identified by their source and destination IP addresses, and flows identified by both port numbers in addition to the addresses. Flow-level information includes information such as sequence numbers, timeouts, retransmissions, and round-trip times. Path-level information aggregates some flow-level information, such as loss rates and RTTs.

MonD adds new entries when it sees new paths/flows. On packet arrival, MonD updates a timestamp for the flow entry. Inactive flows, which have not received any traffic in *FlowLifeTime* (15 minutes in the current system), are pruned from the path entry, and any empty paths are removed from the table.

3.3 MonD Behavior

MonD uses two indicators to identify possible anomalies, which are then forwarded to ProbeD for confirmation. The first indicator is a change in a flow's Time-To-Live (TTL) field. The TTL field in an IP packet is initialized by a remote host and gets decremented by one at each hop along the traversed path. If the path between a source and destination is static, the TTL value of all packets that reach the destination should be the same. If the TTL changes in the middle of the stream, it usually means a routing change has occurred.

The second indicator, multiple consecutive timeouts, signals a possible path anomaly since such timeouts should be relatively rare. A TCP flow can time out several times from a single unacknowledged data packet, and each consecutive timeout causes the retransmission timeout period to double [25]. The minimum initial retransmission timeout in TCP ranges from 200ms (in Linux) to 1 second (in RFC 2988 [25]). Thus, n consecutive timeouts means either the data packets or the corresponding acknowledgment packets (ACKs) have not been received during the last $2^n - 1$ periods (seconds or 200ms ticks).

Our current threshold is four consecutive timeouts, which corresponds to 3.2–16 seconds. Since most congestion periods on today's Internet are short-lived (95% are less than 220ms [27]), these consecutive timeouts are likely due to path anomalies. We can further subdivide this case based on whether MonD is on the sender or receiver side of the flow. If MonD is on the receiver side, then no ACKs are reaching the sender, and we can infer the problem is on the path from the CoDeeN node to the client/server, which we call **forward path**. If MonD is on the sender side, then we cannot determine whether outbound packets are being lost or whether ACKs are being lost on the way to MonD.

3.4 MonD Flow/Path Statistics

We now describe how MonD infers path anomalies after grouping packets into flows. We examine how to measure the per-flow RTTs, timeouts and retransmissions.

To detect timeouts when MonD is on the sender side, we maintain two variables, *SendSeqNo* and *SendRtxCount* for each flow. *SendSeqNo* is the sequence number (seqno) of the most recently sent new packet, while *SendRtxCount* is a count of how many times the packet has been retransmitted. If we use *CurrSeqNo* to represent the seqno of the packet currently being sent, we see three cases: If $CurrSeqNo > SendSeqNo$, the flow is making progress, so we clear *SendRtxCount* and set *SendSeqNo* to *CurrSeqNo*. If $CurrSeqNo < SendSeqNo$, the packet is a fast retransmit, and we again set *SendSeqNo* to *CurrSeqNo*. If $CurrSeqNo = SendSeqNo$, we conclude a timeout has occurred and we increment *SendRtxCount*. If *SendRtxCount* exceeds our threshold, MonD notifies ProbeD that a possible path anomaly has occurred.

A similar mechanism is used when MonD observes the receiver side of a TCP connection. It keeps track of the largest seqno received per flow, and if the current packet has the same seqno, a counter is incremented. Doing this determines how many times a packet has been retransmitted due to consecutive timeouts at the sender. When this counter hits our threshold, MonD notifies ProbeD that this sender is not seeing our ACKs. Since we are seeing the sender's packets, we know that this direction is working correctly. Note that this method assumes that duplicate packets are mostly due to retransmissions at the sender. This assumption is safe because previous work has shown that packets are rarely duplicated by the network [20].

Detecting TTL change is trivial: MonD records the TTL for the first packet received along each path. For each packet received from any flow on the same path, we compare its TTL to our recorded value. If MonD detects any change, it notifies ProbeD that a possible anomaly has occurred. Note that this case can aggregate information from all flows along the path.

3.5 ProbeD Operation

ProbeD is responsible for the active probing portion of PlanetSeer, and generally operates after being notified of possible anomalies by MonD. For the purpose of the following discussion, when an anomaly occurs, we call the CoDeeN node where the anomaly is detected the *local node*, and the corresponding remote client/server the *destination*. The active probing is performed using traceroute, a standard network diagnostic tool. ProbeD supports three probing operations:

Baseline Probes: When a new IP address is added to MonD's path table, the ProbeD on the local node performs a "baseline probe" to that destination. It is expected that the results of this traceroute reflect the default network path used to communicate with that destination under normal conditions. For actively-used communication paths, a baseline probe is launched once every 30 minutes. When PlanetSeer is run on CoDeeN nodes, these baseline probes are generated whenever a new client connects to a node, or when a node has to contact a new origin server.

Forward Probes: When a possible anomaly is detected by the local MonD and reported to ProbeD, it invokes multiple traceroutes from a set of geographically distributed nodes (including itself) to the destination; we call the traceroute from the local node the *local traceroute* or *local path*. This process is performed twice, generally within one minute, in order to identify what we term *ultrashort* anomalies. On particularly problematic paths, MonD might report possible anomalies very frequently, especially if the path is very unstable. To avoid generating too much probing traffic, ProbeD rate-limits the forward probes so that it does not probe the same destination within 10 minutes.

Reprobes: If the forward probes confirm an anomaly along a path to a destination, the local ProbeD that initiated the forward probes periodically reprobes that path to determine the duration and effects of the anomaly. We currently reprobe four times, at 0.5, 1.5, 3.5, and 7.5 hours after the anomaly detection time. These reprobes can compare their traceroute results with the original baseline probe as well as the forward probes.

3.6 ProbeD Mechanics

When ProbeD performs the forward probes, it launches them from geographically distributed nodes on PlanetLab. Compared with only doing traceroute from the local node, using multiple vantage points gives us a more

Category	Grps	Sites	Descriptions
US (edu)	11	70	US Universities
US (non-edu)	5	13	Intel, HP, NEC, etc.
Canada	2	11	Eastern & Western Canada
Europe	7	31	UK, France, Germany, etc.
Asia & MidE	4	14	China, Korea, Israel, etc.
Others	1	6	Australia, Brazil, etc.
Total	30	145	

Table 1: Groups of the probing sites

Tier #	Covered	Tier Size	Coverage
Tier 1	22	22	100%
Tier 2	207	215	96%
Tier 3	1119	1392	80%
Tier 4	1209	1420	85%
Tier 5	5906	13872	43%
Unmapped	1627		

Table 2: Path diversity

complete view of the anomaly, such as its location, pattern, and scope. Our ProbeDs are running on 353 nodes across 145 sites on PlanetLab, more than the number of nodes running CoDeeN. They are distributed across North/South America, Europe, Asia and elsewhere.

Since the set of ProbeDs must communicate with each other, they keep some membership information to track liveness. We note that an unavailable ProbeD only results in a degradation of information quality, rather than complete inoperability, so we do not need to aggressively track ProbeD health. Each ProbeD queries the others when it first starts. Thereafter, dead ProbeDs are checked every 8 hours to reduce unneeded communication. In the course of operation, any ProbeD that is unresponsive to a query is considered dead.

We divide the ProbeD nodes into 30 groups based on geographic diversity, as shown in Table 1, mainly to reduce the number of probes launched per anomaly. Probing every anomaly from every ProbeD location would yield too much traffic (especially to sites with conservative intrusion detection systems), and the extra traffic may not yield much insight if many nodes share the same paths to the anomaly. We also divide North America into educational and non-educational groups, because the educational (.edu) sites are mostly connected to Internet2, while non-educational sites are mostly connected by commercial ISPs.

When a ProbeD receives a request from its local MonD, it forwards it to a ProbeD in each of the other groups. The ProbeDs perform the forward probes, and send the results back to the requester. All results are collected by the originator, and logged with other details, such as remote IP address and the current time.

3.7 Path Diversity

We have been running PlanetSeer since February 2004. In three months, we have seen 887,521 unique clients IPs, coming from 9232 Autonomous Systems (ASes) (according to previous IP-to-AS mappings techniques [17]). Our probes have traversed 10090 ASes, well over half of all ASes on the Internet. We use a hierarchical AS classification scheme that has five tiers, based on AS relationships and connectivity [23]. The highest layer (tier 1) represents the *dense core* of the Internet, and consists of 22 ASes of the largest ISPs. Tier 2 typically includes ASes of other large national ISPs. Tier 3 includes ASes of regional ISPs. Finally, tiers 4 and 5 include ASes of small regional ISPs and customer ASes respectively. As shown in Table 2, we have very good coverage of the top 4 AS tiers, with complete coverage of tier 1 and nearly-complete coverage of tier 2.

4 Confirming Anomalies

Having collected the passive data from MonD and the traceroutes from ProbeD, the next step is processing the probes to confirm the existence of the anomaly. This section describes how we use this data to classify anomalies and quantify their scope. It also reports how different types of anomalies influence end-to-end performance.

4.1 Massaging Traceroute Data

Some of the data we receive from the traceroutes is incomplete or unusable, but we can often perform some simple processing on it to salvage it. The unusable hops in a traceroute are those that do not identify the routers or that identify the routers by special-use IP addressess [11]. The missing data is generally the absence of a hop, and can be interpolated from other traceroutes.

Identifying and pruning unusable hops in traceroute is simple: the unusable hops are identified by asterisks in place of names, and other than showing the existence of these hops, these data items provide no useful information. We simply remove them from the traceroute but keep the relative hop count difference between the existing hops.

Missing hops in a traceroute are slightly harder to detect, but we can use overlapping portions of multiple traceroutes to infer where they occur. We use a simple heuristic to identify the missing hops: we compare traceroutes that share the same destination, and if the hops leading to the destination differ by an intermediate hop that is present in one and missing in the other, we replace the missing hop with the address in the other trace. Put more formally, given two traceroutes from two sources to the same destination, suppose there are two subsequences of these two traceroutes, $X(X_1, X_2, ..., X_m)$ and $Y(Y_1, Y_2, ..., Y_n)$ ($m > 2$ and $n > 2$) such that $X_1 = Y_1$ and $X_m = Y_n$. We

TTL Change	Timeout	Fwd Timeout
994485 (44%)	754434 (33%)	510669 (23%)

Table 3: Breakdown of anomalies reported by MonD

define $hop(X_i)$ to be the hop count of X_i. Note that the number of hops between X_1 and X_m, $hop(X_m) - hop(X_1)$, can be greater than $m - 1$, because we have removed "*" and special-use IPs from the traceroutes. If $hop(X_m) - hop(X_1) = hop(Y_n) - hop(Y_1)$, it is very likely that all the hops in X and Y are the same since they merge at $X_1(Y_1)$ and follow the same number of hops to $X_m(Y_n)$. If there exists X_i such that the hop corresponds to $hop(Y_1) + hop(X_i) - hop(X_1)$ in Y does not exist because it has been removed as "*", we consider X_i a missing hop in Y and add this hop into Y.

Our approach to inserting missing hops helps us filter out the "noise" in traceroutes so that it does not confuse our anomaly confirmation using route change as described in Section 4.2. However, it may mask certain hop differences. For example, we sometimes see two paths X and Y merge at $X_1(Y_1)$, and diverge at some later hops before merging at $X_m(Y_n)$ again. This usually occurs between tightly-coupled routers for load balancing reasons, where a router selects the next hop from several parallel links based on the packet IP addresses and/or traffic load [19]. In this case, inserting missing hops may eliminate the different hops between the two traceroutes.

For our purposes, we do not treat such "fluttering" [19] as anomalies because it occurs as a normal practice. We detect fluttering by looking for hops X_i and Y_j such that $hop(Y_j) - hop(Y_1) = hop(X_i) - hop(X_1)$ but $X_i \neq Y_j$, and we merge them into the same hop in all the traceroutes. Note that this could also possibly eliminate the hop difference caused by path change and lead to underestimating the number of anomalies.

4.2 Final Confirmation

After we have processed the traceroutes, we are ready to decide whether an event reported by MonD is actually an anomaly. We consider an anomaly "confirmed" if *any* of the following conditions is met:

Loops: There is a loop in the local traceroute from the local node to the destination, which means the anomaly is triggered by routing loops.

Route change: The local traceroute disagrees with the baseline traceroute. Note that the baseline traceroute is no more than 30 minutes old. Given that 91% of the Internet paths remains stable for more than several hours [19], the anomaly is most likely caused by path change or path outage.

Non-Anomaly	Anomaly	Undecided
1484518 (66%)	271898 (12%)	503172 (22%)

Table 4: Breakdown of reported anomalies using the four confirmation conditions

Partial unreachability: The local traceroute stops before reaching the destination, but there exist traceroutes from other nodes that reach the destination. This could be caused by path outages.

Forwarding failures: The local traceroute returns an ICMP destination unreachable message, with code of net unreachable, host unreachable, net unknown, or host unknown. This usually indicates that a router does not know how to reach the destination because of routing failures [13].

Our confirmation process is very conservative—it is possible that some of the reported anomalies are real, but do not meet any of the above conditions. However, our goal is to obtain enough samples of anomalies for our analysis and we do not want our results to be tainted by false positives. Hence, we choose stricter conditions for confirming anomalies. Similarly, we confirm a reported anomaly as *non-anomaly* if the local traceroute does not contain any loop, agrees with the baseline traceroute, and reaches the destination. For a confirmed non-anomaly, we do not perform traceroutes at remote ProbeDs, in order to reduce measurement traffic.

In three months, we have seen a total of 2,259,588 possible anomalies reported, of which we confirmed 271,898. Table 3 shows the number of reported anomalies of each type. As we can see, TTL change is the most common type of reported anomaly, accounting for 44% of the reported anomalies. For the remaining anomalies triggered by timeouts, passive monitoring suggests that 23% are most likely caused by forward path problems.

Table 4 shows the breakdown of anomalies using the 4 confirmation conditions. The non-anomalies account for 2/3 of the reported anomalies. Among the possible reasons for the occurrence of non-anomalies are: ultrashort anomalies, path-based TTL, and aggressive consecutive timeout levels. Some anomalies, which we term ultrashort, are so short that our system is unable to respond in time. Since they often are in the process of being resolved when the forward probes are taking place, the traceroute results may be inconsistent. Many false alarms from path-based TTL changes are due to NAT boxes. When clients with different initial TTL values share a NAT box, their interleaved packets appear to show TTL change. Using flow-based TTL change would reduce these false alarms, but may miss real TTL changes that occur between flows since any path history

	Temporary	Persistent
Total	3565 (17%)	18000 (83%)
≤ 30min	N/A	54%
≤ 1.5 hrs	N/A	11%
≤ 3.5 hrs	N/A	6%
≤ 7.5 hrs	N/A	6%
> 7.5 hrs	N/A	23%
Single Loop	3008 (84%)	17007(94%)
Multiple Loops	557 (16%)	993 (6%)
1 AS	3021, (85%)	17895, (99%)
2 ASes	416, (12%)	101, (1%)
3 ASes	106, (3%)	4, (0%)
≥4 ASes	22, (0%)	0, (0%)
Tier-1 AS	510 (15%)	244 (2%)
Tier-2 AS	859 (25%)	789 (6%)
Tier-3 AS	1378(40%)	6263 (46%)
Tier-4 AS	197 (5%)	3899 (29%)
Tier-5 AS	538 (15%)	2401 (17%)
Total	3482	13596

Table 5: Summarized breakdown of 21565 loop anomalies. Some counts less than 100% because some ASes are not in the AS hierarchy mapping.

would be lost. Finally, our consecutive timeout conditions may be aggressive for hosts with short timeout periods. Excluding the non-anomalies, we confirm 271898 (35%) of the remaining anomalies and probe them from multiple vantage points. We use these anomalies for our analysis in the remainder of this paper.

5 Loop-Based Anomalies

This section focuses on analyzing routing loops, which can occur due to inconsistencies in routing state, misconfiguration, and other causes. We are interested in their frequency, duration, size, location, and effect on end-to-end performance.

We detect routing loops by observing the same sequence of routers appearing several times in a traceroute. Since loops in traceroute may reflect upstream routing changes rather than true routing loops [19], we choose to take a conservative approach and require that the same sequence appear in a traceroute **at least 3 times** before we confirm it as a routing loop.

Using this metric, we identify a total number of 21565 routing loops in our data. If we relax the loop condition to allow loops that have the same sequence only twice, this count increases to 119,936, almost six times as many. Using this approach, our overall confirmed anomaly count would increase 36% to over 370K.

Loops are separated into persistent and temporary loops [19] based on whether the traceroute was ultimately able to exit the loop. If the traceroute stays within the loop until the maximum number of hops (32 in our case), we classify it as persistent, while if the loop is resolved before the traceroute reaches the maximum hop, it is temporary. Temporary loops can occur due to the time necessary to propagate updated routing information to the different parts of the network, while persistent loops can be caused by various reasons, including router misconfiguration [16]. Persistent loops tend to last longer and may require human intervention to be resolved, so they are considered more harmful. About 83% of the observed loops are persistent, as shown in Table 5. Since temporary loops only exist for a short period, it may be harder to catch them.

We use the reprobes to determine duration of the persistent loops. The reprobes for some persistent loops are missing, often because the local PlanetLab node failed or rebooted before all reprobes completed. For those loops, we do not know when the loops were resolved. We only include the loops having all 4 reprobes in our analysis. Therefore, we show the percentage of loops in each duration instead of the exact numbers in Table 5. We can see many persistent loops are either resolved quickly (54% terminate within half an hour) or last for a long time (23% stay for more than 7.5 hours).

Previous research has noted that routing loops are likely to be correlated [19], because nearby routers usually share routing information very quickly. If some routers have inconsistent information, such information is likely to be propagated to other nearby routers and cause those routers to form loops. We observe a similar phenomenon, which is quantified in Table 5. We count the number of distinct loops in traceroutes from other ProbeDs during the same period. We find that 16% of the temporary loops are accompanied by at least one disjoint loop while only 6% of the persistent loops see them. We suspect the reason is temporary loops are more likely to stem from inconsistent routing state while persistent loops are more likely to be caused by other factors which may not be related to routing inconsistency.

Finally, using the persistent loop data, we can also get some insight into the relative distribution of AS quality by measuring how evenly-distributed the loops are. Since these loops are largely single-AS, they are very unlikely to arise from external factors, and may provide some insight into the monitoring/management quality of the AS operators. We use a metric, which we call *skew*, to provide some insight into the distribution. We calculate skew as the percentage of per-tier loops seen by the "worst" 10% of the ASes in that tier. A skew value of 10% indicates all ASes in the tier are likely to be uniform in the quality, while larger numbers indicate a wider disparity between the best ASes and the worst.

In tier 1, the top 2 ASes (10% of 22) account for 35% of the loops, while in tier 2, the top 21 ASes (10% of 215)

	2	3	4	5	6+
Persistent/All	97%	2%	1%	0%	0%
Persistent/Core	94%	4%	1%	1%	0%
Persistent/Edge	97%	2%	1%	0%	0%
Temporary/All	51%	29%	11%	7%	2%
Temporary/Core	45%	31%	13%	8%	3%
Temporary/Edge	53%	27%	12%	6%	2%

Table 6: Number of hops in loops, as % of loops

account for 97% of the loops. This skew may translate into different reliabilities for the customers they serve. The disparity in traffic must also be considered when judging how important these skew numbers are. With respect to the traffic that we observe, we find that these ASes account for 20% of tier 1 traffic and 63% of tier 2 traffic. The disparity between the loop rates and the traffic for these ASes would indicate that these ASes appear to be much more problematic than others in their tier.

5.1 Scope

Besides their frequency, one of the other factors that determines the effect of routing loops is their *scope*, including how many routers/ASes are involved in the loop, and where they are located. We use the term *loop length* to refer to the number of routers involved, and we show a breakdown of this metric in Table 6. The most noticeable feature is that temporary loops have much longer lengths than persistent loops. 97% of the persistent loops consist of only 2 routers, while the ratio is only 50% for temporary loops. Intuitively, the more routers are involved in a loop, the less stable it is. Therefore, most persistent loops exist between only two routers, while temporary loops span additional routers as the inconsistent state propagates.

We next examine the number of ASes that are involved in the loops. The breakdown is shown in Table 5, which shows that persistent loops overwhelmingly occur within a single AS, while 15% of temporary loops span multiple ASes. Ideally, BGP prevents any inter-AS loops by prohibiting a router from accepting an AS path with its own AS number in that path. However, BGP allows transient inconsistency, which can arise during route withdrawals and announcements [10], and this is why we see more temporary loops spanning multiple ASes. In contrast, persistent loops can occur due to static routing [19] or router misconfigurations [16]. Given how few persistent loops span multiple ASes, it appears that BGP's loop suppression mechanism is effective.

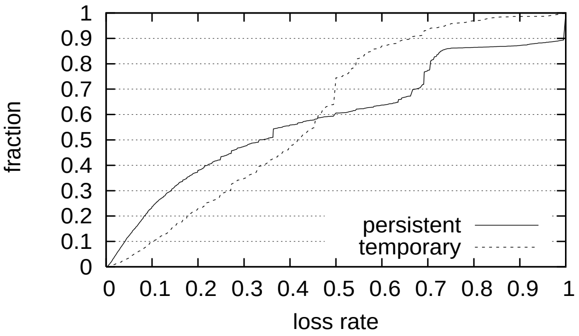

Figure 1: CDF of loss rates proceding the loop anomalies

To understand where loops arise, we classify them according to the hierarchy of ASes involved. In theory, we could calculate their depth [7], which would tell us the minimum number of hops from the routers to the network edge. However, since we cannot launch probes from the clients, this depth metric would be misleading. If the loop occurs on an AS that does not lie near any of our ProbeD locations, our probes travel through the network core to reach it, and we would believe it to be very far from the edge. If we could launch probes from the clients, network depth would be meaningful.

We map loops into tiers by using the tier(s) of the AS(es) involved. A loop can be mapped to multiple tiers if it involves multiple ASes. Table 5 shows the number of loops occurring in each tier. Tier-1 and tier-2 ASes have very few persistent loops, possibly because they are better provisioned than smaller ASes. A large portion of loops (40% for temporary and 46% for persistent) occur in tier-3 (outer core) ASes, which suggests that the paths in those large regional ASes are less stable. In Table 6, we compare the loops in the core network (tiers 1, 2, 3) or the edge network (tiers 4, 5). As the table shows, both temporary and persistent loops are likely to involve more hops if occurring in the core network.

5.2 End-to-End Effects

Loops can degrade end-to-end performance in two ways: by overloading routers due to processing the same packet multiple times [10] (for temporary loops), or by leading to loss of connectivity between pairs of hosts (for permanent loops). Since MonD monitors all flows between the node and remote hosts, we can use the network statistics it keeps to understand end-to-end effects.

When MonD suspects an anomaly, it logs the retransmission rate and RTT *leading up to that point.* Retransmission rates are calculated for the last 5 minutes. RTTs are calculated using an Exponentially Weighted Moving Average (EWMA) with the most recent value given a weight of 1/8, similar to that used in TCP. Figure 1 shows the CDF of the retransmission rate, and we see that 65% of the temporary loops and 55% of the persistent loops are preceded by loss rates exceeding 30%. Since the typical Internet loss rate is less than 5% [20], this higher loss rate will significantly reduce end-user TCP performance prior to the start of the anomaly.

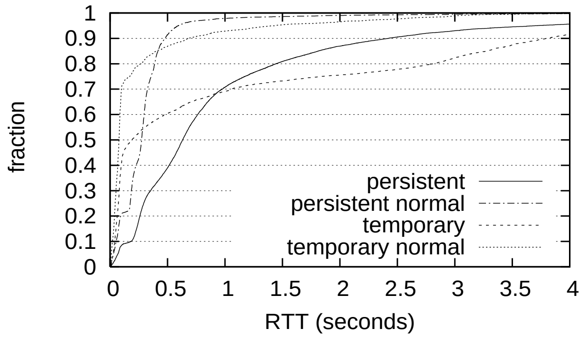

Figure 2: CDF of RTTs proceding the loop anomalies *vs.* under normal conditions

In addition to the high loss rates, loop anomalies are also preceded by high latency, as shown in Figure 2. High latency can be caused by queuing due to congestion or packets repeatedly traversing the same sequence of routers. We compare the RTT right before loops occur with the RTT measured in the baseline traceroute on the same path. It is evident that loops can significantly degrade RTTs.

6 Building a Reference Path

While loop-based anomalies are relatively simple to identify, they represent a relatively small fraction of the total confirmed anomalies. Classifying the other anomalies, however, requires more effort. This section discusses the steps taken to classify the non-loop anomalies, and the complications involved. The main problem is how to determine that the anomaly occurs on the forward path from the local node to the destination. Additionally, we want to characterize the anomaly's features, such as its pattern, location and affected routers.

To deal with these two problems, we need a *reference path* from the local node to the destination, which represents the path before the anomaly occurs. Then we can compare it against the local traceroute during the anomaly. This comparison serves three purposes: First, it can help us distinguish between forward-path and reverse-path anomalies. If the local path during the anomaly is different from the reference path, the route change usually indicates that the anomaly is on the forward path. Second, it can be used to quantify the scope of the anomaly. By examining which routers overlap between the local path and the reference path, we can estimate which routers are potentially affected by the anomaly. It can also be used to determine the location of the anomaly, in terms of the number of the router hops between the anomaly and the end hosts. Third, it is used to classify the anomalies. Based on whether the local traceroute reaches the last hop of the reference path, we can classify it as either path change or path outage.

Ideally, we could simply use the baseline traceroute as the reference path, if it successfully reaches the destination. If the local traceroute during an anomaly stops at some intermediate hop, we know it is an outage. If the local traceroute is different, but reaches the destination, we know it is a routing change. However, the baseline traceroute may not reach the destination for a variety reasons. Some of these include:

- The destination is behind a firewall which filters traceroutes. In this case, we still want to use it as the reference path, which can be compared with local traceroute to analyze anomalies.
- Some intermediate routers filter traceroutes. In this case, we do not have enough information about the hops after the last known hop on the forward path. When an outage occurs, we cannot quantify where it occurs since the anomaly may occur after the last known hop.
- The baseline traceroute is also affected by the anomaly and fails to reach the destination. In this case, we cannot use it as a reference because it usually does not provide more useful information than the local traceroute.

The rest of this section focuses on deciding whether the baseline traceroute can be used as the reference path when it does not reach the destination. If a baseline path S stops at hop S_x, we try to guess if S_x is a firewall using some heuristics. S_x must meet the following four requirements before we consider it a firewall:

1. From MonD's passive data, we know the client is able to send and receive TCP packets with the local node. Therefore, the path is working when the baseline traceroute is being calculated.
2. S does not return an ICMP destination unreachable message, which usually indicates that the traceroute encounters routing problems at S_x [13].
3. S_x and the destination are in the same AS. We assume that a firewall and its protected clients should belong to the same organization.
4. S_x is within n hops (close) to the destination.

The first three requirements are easy to verify, so we focus on the last requirement. Let $RevHop(h)$ be the number of hops from hop h to the local node on the reverse path. We first want to check if $0 < RevHop(dst) - RevHop(S_x) \leq n$. From MonD, we know $RevTTL(dst)$, the TTL of a packet when it arrives at the local node from the destination. If

Type	Number	Percentage
Path Change	120,283	48%
Forward Outage	23,921	10%
Other Outage	62,107	24%
Temporary	44,022	18%
Total	250,333	100%

Table 7: Non-loop anomalies breakdown

the TTL is initialized to $InitTTL(dst)$ by the destination, we have $InitTTL(dst) - RevTTL(dst) = RevHop(dst)$ because the TTL is decremented at each hop along the reverse path. The issue is how to determine $InitTTL(dst)$. The initial TTL values differ by OS, but are generally one of the following: 32 (Win 95/98/ME), 64 (Linux, Tru64), 128 (Win NT/2000/XP), or 255 (Solaris). Because most Internet paths have less than 32 hops, we can simply try these 4 possible initial TTL values and see which one, when subtracted by $RevTTL(dst)$, gives a $RevHop(dst)$ that is less than 32 hops. We will use that as $InitTTL(dst)$ to calculate $RevHop(dst)$. Similarly, from the traceroute, we can also calculate $RevHop(S_x)$ using $RevTTL(S_x)$.

Although inter-AS routing can be asymmetric, intra-AS paths are usually symmetric [19]. Since S_x and the destination are in the same AS, their forward hop count difference should be the same as their reverse hop count difference, which we are able compute as described above.

Choosing an appropriate n for all settings is difficult, as there may be one or more hops between a firewall and its protected hosts. We conservatively choose $n = 1$, which means we consider S as a valid reference path only when S_x is one hop away from the destination. This will minimize the possibility that a real path outage is interpreted as a traceroute being blocked by a firewall. However, it leads to bias against large organizations, where end hosts are multiple hops behind a firewall. In such cases, we cannot determine if the anomalies are due to path outage or blocking at a firewall. Therefore, we do not further analyze these anomalies.

7 Classifying Non-loop Anomalies

In this section, we discuss classifying anomalies by comparing the reference path R with the local path L. There are three possibilities when we compare L and R:

1. L reaches the last hop of R. In this case, L must differ from R in some intermediate hops, or else we would not have confirmed it as an anomaly. This case corresponds to a *path change*, which will be discussed in Section 7.1.

2. If L stops at some intermediate hop of R, it could be due to path outage on the forward path or reverse path failure. We will describe how to distinguish between them in Section 7.2.

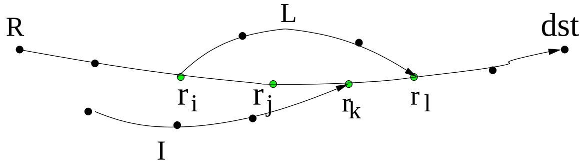

Figure 3: Confining the scope of path change

3. If L diverges from R after some hops and stops before merging into R, we consider it as a path outage although it is accompanied by a route change. We will also discuss this case in Section 7.2.

We observe a total of 250333 non-loop anomalies, with their breakdown shown in Table 7. About half of them are path changes, and 10% are forward path outages. For the 24% that are classified as other outages, we cannot infer whether they are on the forward or reverse paths. The remaining 18% are temporary anomalies. In these cases, the first local traceroute does not match the reference path, but the second local traceroute matches. In these cases, the recovery has taken place before we can gather the results of the remote probes, making characterization impossible. While it is possible that some remote probes may see the anomaly, the rapidly-changing state is sure to cause inconsistencies if we were to analyze them. To be conservative, we do not perform any further analysis of these anomalies, and focus only on path changes and forward outages. These temporary anomalies are different from the ultrashort anomalies in that the ultrashort anomalies were already in the repair process during the first probe. So, while we choose not to analyze temporary anomalies further, we can at least inarguably confirm their existence, which is not the case with the ultrashort anomalies.

7.1 Path Changes

We first consider path changes, in which the local path L diverges from the reference path R after some hops, then merges back into R and successfully terminates at the last hop of R. This kind of anomaly is shown in Figure 3.

7.1.1 Scope and End-to-End Effects

As discussed in Section 2, it is usually very difficult to locate the origin of path anomalies purely from end-to-end measurement [8]. However, even if the precise origin cannot be determined, we may be able to narrow the *scope* of the anomaly. We define the scope of a path change as the number of hops on R which possibly change their next hop value. Flows through these routers may all have their paths changed. In Figure 3, L

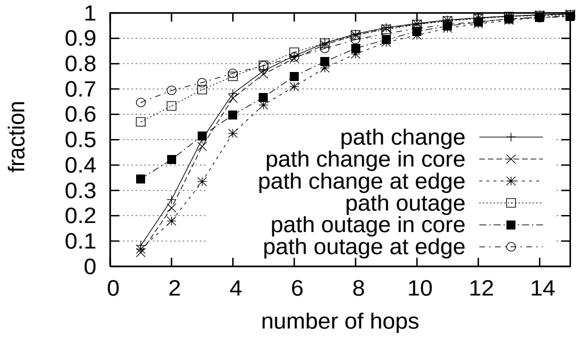

Figure 4: Scope of path changes and forward outages in number of hops

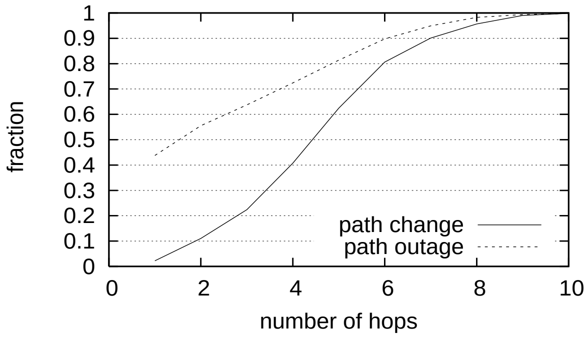

Figure 5: Distance of path changes and forward outages to the end hosts in number of hops

	Change	Fwd Outage
Total	120283	12740
No Ref Path	N/A	11181
1 AS	24418 (20%)	6534 (51%)
2 ASes	43909 (37%)	3413 (27%)
3 ASes	29426 (25%)	1321 (10%)
4 ASes	12603 (10%)	856 (7%)
5 ASes	6322 (5%)	411 (3%)
6 ASes	3605 (3%)	205 (2%)
Guessed Last Hop	N/A	1055
Scope Changed	4292 (4%)	1225 (10%)
Tier-1 AS	12374 (6%)	2746 (15%)
Tier-2 AS	43104 (23%)	3255 (18%)
Tier-3 AS	88959 (47%)	4638 (26%)
Tier-4 AS	8015 (4%)	3501 (19%)
Tier-5 AS	38313 (20%)	3838 (21%)
Total	190765	17978

Table 8: Summary of path change and forward outage. Some counts exceed 100% due to multiple classification.

diverges from R at r_i and merges into R at r_l. All the hops before r_i or after r_l (including r_l) follow the same next hop towards the destination. So the hops which are possibly influenced by the path change and have different next hops are r_i, r_j and r_k.

In some cases, we may be able to use remote traceroutes to narrow the scope even further. For example, in Figure 3, if I, a traceroute from another ProbeD merges into R at r_k, a hop that is before r_l, we can eliminate r_k from the scope of the path change anomaly, since we know r_k has the same next hop value as it did in the reference path. We call I the *intercept path*. This method may still overestimate the scope of path change: it is possible, for example, that r_j's next hop value is unaffected, but we cannot know this unless some traceroute merges into R at r_j.

Performing traceroute from multiple geographically diverse vantage points increases our chances of finding an intercept path. Our ability of narrowing the scope is affected by the location of the anomaly. If it is closer to the destination, we have a better chance of obtaining intercept paths by launching many forward traceroutes, and thus successfully reducing the scope of the anomaly. In contrast, if the anomaly is nearer our source, the chance of another traceroute merging into the reference path early is low.

Figure 4 shows the CDF of path change scope, measured in hop count. We can confine the scope of 68% of the path changes to within 4 hops. We do not count fluttering as path changes, since these would appear as a large number of path change anomalies, each with a very small scope. We also examine how many ASes the path change scope spans, shown in Table 8. Again, we can confine the scope of 57% of the path changes to within two ASes and 82% of them to within three ASes.

To gain some insight into the location of the anomalies, we also study whether the path change occurs near end hosts or in the middle of the path [7]. We measure the *distance* of a path change to the end host by averaging the distances of all the routers within the path change scope. The *distance* of a router is defined as the minimum number of hops to either the source or the destination. Figure 5 plots the CDF of path change distances. As we can see, 60% of the path changes occur within 5 hops to the end hosts.

We can also use AS tiers to characterize network locations, so we map the routers within anomaly scopes to ASes and AS tiers. The breakdown of possibly affected routers by their AS tiers is shown in Table 8. The routers in Tier-3 are most likely to be affected by path changes, since they account for nearly half of the total. By comparison, the routers in tier-1 ASes are rather stable, though presumably they are traversed repeatedly by many paths.

In Figure 4, we see that path changes in the core net-

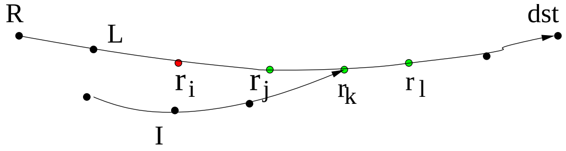

Figure 6: Confining the scope of forward outage

work have narrower scope than those in the edge. This is probably because the paths in the core network are likely to be traversed by traceroutes from many vantage points to reach the destination. In contrast, if a route change occurs near a local node, we have less chance of finding an intercept path that happens to merge into the reference path early. As a result, the anomaly scope in these cases is more loosely confined.

Since path change is a dynamic process and anomaly scopes may evolve over time, a measured anomaly scope should be viewed as a snapshot of which routers are affected when the traceroutes reach them. In Table 8, we show how many path changes have changed scope between the first and second sets of forward probes. We find that only 4% of them have changed during that period (mostly within one minute). In addition, 66% of the scope changes are due to local path changes instead of intercept path changes.

We now examine the effect of path changes on end-to-end performance. The effect of path changes on RTTs is relatively mild, as can be seen in Figures 8. The RTTs measured during path changes are only slightly worse than the RTTs measured in baseline traceroutes. But the loss rates during path changes can be very high. Nearly 45% of the path changes cause loss rates higher than 30%, which can significantly degrade TCP throughput.

7.2 Path Outage

We now focus on path outages and describe how to distinguish between forward and reverse path outages. In Figure 6, the local path L stops at r_i, which is an intermediate hop on the reference path R. At first glance, one might conclude that a failure has occurred after r_i on the forward path, which prevents the packets from going through; but other possibilities also exist—because Internet paths could be asymmetric, a failure on the reverse path may produce the same results. For example, if a shared link on the reverse paths from all the hops beyond r_i to the source has failed, none of the ICMP packets from those hops can return. Consequently, we will not see the hops after r_i.

If we have control of the destination, we can simply distinguish between forward and reverse path outages using ping [7]. However, since our clients are outside of PlanetLab and not under our control, we cannot perform pings in both directions, and must use other information to disambiguate forward path outages from reverse path failures. Specifically, we can infer that the outage is on the forward path using the following rules:

Route change	Unreachable	Fwd Timeout
12822 (54%)	2751 (11%)	8348 (35%)

Table 9: Breakdown of reasons for inferring forward outage

- There is a route change on the forward path in addition to the outage.
- The local traceroute returns an ICMP destination unreachable message.
- The anomaly is reported as *timeouts on forward path*. As described in Section 3.4, MonD will report this type of anomaly when it infers ACK losses on the forward path from the local node to the client.

Table 9 shows the number of forward path outages inferred from each rule. As we can see, all three rules are useful in identifying forward outages. More than half of the outages are accompanied by route changes, as the failure information is propagated and some routers try to bypass the failure using other routes. Forward timeouts help us infer one third of the forward outages. This demonstrates the benefit of combining passive monitoring with active probing, since we would not have been able to disambiguate them otherwise.

7.2.1 Scope

To characterize the scope of path outages, we use a technique similar to the one we used to characterize the scope of path change. We define a path outage scope as the number of hops in a path that cannot forward packets to their next hop towards the destination. In Figure 6, R is the reference path and L is the local path. L stops at r_i, which is an intermediate hop of R. Hence, all the hops after r_i (including r_i) are possibly influenced by the outage and may not be able to forward packets to the next hops towards the destination. However, when we can find another intercept path, we can narrow the scope. For example, if I merges into R at r_k and reaches the destination, then only r_i and r_j can possibly be influenced by the outage. Again, this method might overestimate the scope of a path outage, for the same reasons described earlier on estimating a path change scope.

Note that unlike in previous work [6, 7], we use a set of routers to quantify the effect of path outages instead of just using the last hop where the traceroute stops. Since outage information is usually propagated to many routers, using only one router does not give us a sense of how many routers may have been affected by the outage.

In some cases, we may not have a complete baseline path which reaches the destination or the penultimate router. In these cases, we can not estimate the scope

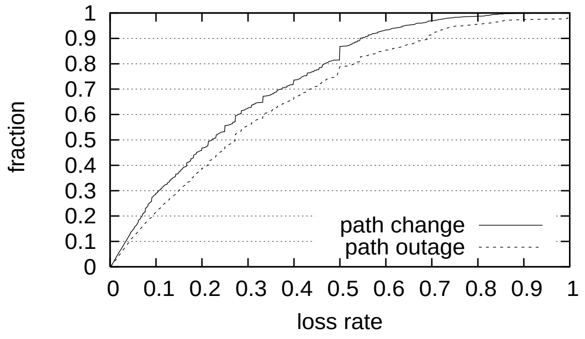

Figure 7: CDF of loss rates proceding path changes and forward outages

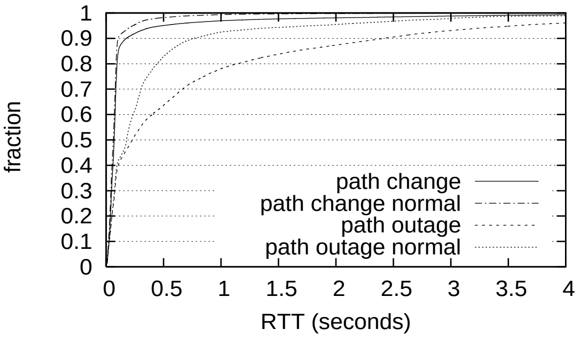

Figure 8: CDF of RTTs proceding path changes and forward outages *vs.* under normal conditions

of the forward outage because we do not know the exact number of hops between the last hop of the baseline path and the destination. We only know that the anomaly occurs somewhere on the forward path. Among all the outages, about 47% have no complete reference path. In the following, we use only those with complete reference paths in the scope analysis.

In Figure 4, we plot the CDF of the number of hops in the forward outage scope. Compared with path change, we can confine the outage scope more tightly. Nearly 60% of the outages can be confined to within 1 hop and 75% of them can be confined to 4 hops.

We suspect that such tight confinement is due to last hop failures. In Figure 5, we plot the distances of forward outages to the end hosts. The distance of an outage is defined as the average distance of the routers within the outage scope to the end hosts, similar to the definition used for path change in Section 7.1. As we can see, 44% of the outages do occur at the last hop, allowing us to confine their scopes to 1 hop. This observation explains why the outages in the edge network are confined more tightly than those in core networks, as shown in Figure 4.

Excluding last hop failures, we can only confine 14% of the outages to one hop, a result that is slightly better than that for path changes. In general, the scopes of path outages tend to be smaller than those of path changes. Compared with path changes in Figure 5, path outages tend to occur much closer to the end hosts. More than 70% of the outages occur within 4 hops to the end hosts.

Table 8 gives the number of ASes that the outages span. Compared with path changes, we can confine a much higher percentage of outages (78%) within two ASes. If we examine the AS tiers where the affected routers are located, outages are spread out more evenly across tiers than path changes are. Paths in tier-1 ASes are the most stable and those in tier-3 ASes are most unstable. If we look at both Table 5 and Table 8, we note that paths in tier-3 ASes are most likely to be affected by all types of anomalies. They account for 40% of temporary loops, 46% of persistent loops, 47% of path changes and 26% of forward outages. In contrast, paths in tier-1 ASes are most stable.

Finally, in Table 8, we find that the scopes of 10% of the forward outages might have changed between the first and second set of forward probes, mostly due to local path changes. Another 8% of the forward outages have reference paths that do not terminate at the destinations. These last hops are considered firewalls based on the heuristic described in Section 6.

7.2.2 End-to-End Effect

We also study how path outages influence end-to-end performance. Not surprisingly, forward outages can be preceded by very high loss rates, which are slightly worse than those generated by path changes. The comparisons are shown in Figure 7. Similarly, outages tend to be preceded by much worse RTTs than path changes, as shown in Figure 8: 23% of the outages experience RTTs that are over one second, while only 7% do when there is no outage. The RTT variances can also be very high: 17% of them exceed 0.5 seconds.

8 Discussion

8.1 Bypassing Anomalies

In addition to characterizing anomalies, one of the goals of PlanetSeer is to provide benefits to the hosts running it. One possible approach is using the wide-area service nodes as an overlay, to bypass path failures. Existing systems, such as RON [1], bypass path failures by indirectly routing through intermediate nodes before reaching the destinations. Their published results show that around 50% of the failures on a 31-node testbed can be bypassed [7]. PlanetSeer differs in size and scope, since we are interested in serving thousands of clients that are not participants in the overlay, and we have a much higher AS-level coverage.

Determining how many failures can be bypassed in our model is more complicated, since we have no con-

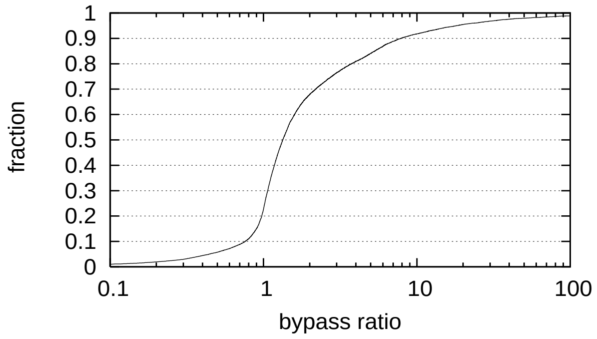

Figure 9: CDF of latency ratio of overlay paths to direct paths

trol over the clients. Clients that are behind firewalls and filter pings and traceroutes may be reachable from other overlay nodes, but we may not be able to confirm this scenario. As a result, we focus only on those destinations that are reachable in the baseline probes, since we can confirm their reachability during normal operation.

For this group of clients, we have a total of **62815** reachability failures, due to anomalies like path outages or loops. Of these failures, we find that some nodes in PlanetSeer are able to reach the destinations in **27263** cases, indicating that one-hop indirection is effective in finding a bypass path for **43%** of the failures.

In addition to improving the reachability of clients using overlay paths, the other issue is their relative performance during failures. We calculate a *bypass ratio* as the ratio between the minimum RTT of any of the bypass paths and the RTT of the baseline path. These results are shown in Figure 9, and we see that the results are moderately promising – 68% of the bypass paths suffer less than a factor of two in increased latency. In fact, 23% of the new paths actually see a latency *improvement*, suggesting that the overlay could be used for improving route performance in addition to failure resiliency. However, some paths see much worse latency degradation, with the worst 5% seeing more than a factor of 18 worse latency. While these paths may bypass the anomaly, the performance degradation will be very noticeable, perhaps to the point of unusability.

8.2 Reducing Measurement Overhead

While PlanetSeer's combination of passive monitoring and distributed active probing is very effective at finding anomalies, particularly the short-lived ones, the probing traffic can be aggressive, and can come as a surprise to low-traffic sites that suddenly see a burst of traceroutes coming from around the world. Therefore, we are interested in reducing the measurement overhead while not losing the accuracy and flexibility of our approach. For example, we can use a single traceroute to confirm loops, and then decide if we want distributed traceroutes to test for the presence of correlated loops. Similarly, for path changes and outages, we can reduce the number of distributed traceroutes if we are willing to tolerate some inaccuracy in characterizing their scope. In Figure 10, we plot the CDF of the number of the probes from other vantages points we have to examine before we find the intercept traceroutes that can successfully narrow the scopes of the anomalies. Using only 15 vantage points, we achieve the same results as when using all 30 vantage points in 80% of the cases. We are interested in studying this issue further, so that we can determine which vantage points we need to achieve good results.

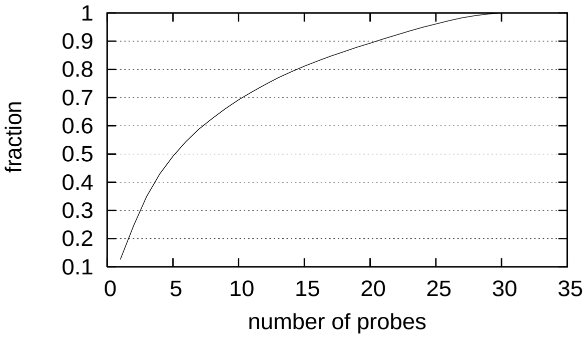

Figure 10: CDF of number of path examined before finding the intercept path

9 Related Work

There is extensive existing work on studying Internet path failure. Labovitz and Ahuja [15] studied inter-domain routing failures using BGP data collected from several ISPs and 5 network exchange points. They 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. They also studied the intra-domain routing failures of a medium-sized regional ISP by examining the data from the trouble ticket tracking system managed by the Network Operation Center (NOC) of the ISP, together with the OSPF routing messages. Based on this data, they characterize the origins of failures into hardware, software and operational problems.

Iannaccone *et al.* investigated the failures in Sprint's IP backbones using the IS-IS routing updates collected from three vantage points [12]. They examined the frequency and duration of failures inferred from routing updates and concluded that most failures are short-lived (within 10 minutes). They also studied the interval between failures. Again, their focus is on the temporal properties of failure.

Mahajan, Wetherall and Anderson [16] studied BGP misconfigurations using BGP updates from Route-

Views [18], which has 23 vantage points across different ISPs. They found BGP misconfigurations were relatively common and classified them into origin and export misconfigurations. Note that BGP misconfigurations may or may not be visible to end users. They observed that only 1 in 25 misconfigurations affect end-to-end connectivity.

More recently, Feldmann *et al.* have presented a methodology to locate the origin of routing instabilities along three dimensions in BGP: time, prefix, and view [9]. Their basic assumption is that an AS path change is caused by some instability either on the previous best path or the new best path. Caesar *et al.* [4] and Chang *et al.* [5] also propose similar approaches to analyze routing changes, although their algorithms are different in details.

All of the above failure studies are based on either inter-domain (BGP) or intra-domain (IS-IS, OSPF) routing messages, from which failures are inferred. Some of them require access to the ISP's internal data, such as trouble tickets. Our study complements this work by studying failures from an end-to-end perspective, and quantifies how failures affect end-to-end performance, such as loss rate and RTT.

There also has been much work on studying Internet failures through end-to-end measurement, and these have greatly influenced our approach. Paxson [19] studied the end-to-end routing behavior by running repeated traceroutes between 37 Internet hosts. His study shows that 49% of the Internet paths are asymmetric and visit at least one different city. 91% of the paths persist more than several hours. He used traceroutes to identify various routing pathologies, such as loops, fluttering, path changes and outages. However these traceroutes do not distinguish between forward and reverse failures.

Chandra *et al.* [6] studied the effect of network failures on end-to-end services using the traceroute datasets [19, 21]. They also used the HTTP traces collected from 11 proxies. They model the failures using their location and duration and evaluate different techniques for masking failures. However, the HTTP and traceroute datasets are independent. In comparison, we combine the passive monitoring data and active probing data, which allows us to detect failures in realtime and correlate the end-to-end effect with different types of failures. They also classify the failures into near-source, near-destination and in-middle by matching /24s IP prefixes with end host IPs. In contrast, we study the location of failures using both IP-to-AS mapping [17] and 5-tier AS hierarchies [23]. This allows us to quantify the failure locations more accurately and at a finer granularity.

Feamster *et al.* measured the Internet failures among 31 hosts using periodic pings combined with traceroutes [7]. They ping the path between a pair of nodes every 30 seconds, with consecutive ping losses triggering traceroutes. They consider the location of a failure to be the last reachable hop in traceroute and used the number of hops to closest end host to quantify the depth of the failure. They characterize failures as inter-AS and intra-AS and use one-way ping to distinguish between forward and reverse failures. They also examine the correlation between path failures with BGP updates.

Our work is partly motivated by these approaches, but we cannot use their methodology directly because of environmental differences. With the large number of clients that connect to our proxies, we can not afford to ping each of them frequently. Failure detection and confirmation are more challenging in our case, since many clients may not respond to pings (behind firewalls) or even are offline (such as dialup users). We infer anomalies by monitoring the status of active flows, which allows us to study anomalies on a much more diverse set of paths. We also combine the baseline traceroutes with passive monitoring to distinguish between forward and reverse failures and classify forward anomalies into several categories. Since where the failure appears may be different from where the failure occurs, we quantify the scope of failures by correlating the traceroutes from multiple vantage points, instead of using one hop (last reachable hop) as the failure location. Finally, we study how different types of anomalies affect end-to-end performance.

10 Conclusion

This paper introduces what we believe to be an important new type of diagnostic tool, one that passively monitors network communication watching for anomalies, and then engages widely-distributed probing machinery when suspicious events occur. Although much work can still be done to improve the tool—e.g., reducing the active probes required, possibly by integrating static topology information and BGP updates—the observations we have been able to make in a short time are dramatic.

- Passive monitoring allows us to detect more anomalies in less time: we have confirmed nearly 272,000 anomalies in three months. This is roughly 3,000 a day, and is 10 to 100 times more than reported previously. We also see a qualitative change, such as large numbers of ultrashort and temporary anomalies that last less than one minute.

- Due to our wide coverage, we see new failure distribution and location properties. Failures are heavily skewed, rather than pervasively distributed: Tier 3 seems to be the most problematic, accounting for almost half of the loops, path changes, and path outages that we see. Tier 1 ASes are generally the most stable.

- We provide some new measurements about routing loop behavior. Temporary loops have much longer lengths than persistent loops. 97% of the persistent loops consist of only 2 routers, but only 50% of temporary loops do. Many temporary loops span 4 routers. This makes sense since the more routers are involved in a loop, the less stable it is. Persistent loops are either resolved in a relative short time (54% last less than 30 minute) or continue for an extended period of time (23% last more than 7.5 hours). Our results confirm Paxson's findings that routing loops are correlated.

- Path changes exhibit different characteristics than outages. Outages appear closer to the edge of the network: 63% of outages occur within 3 hops to end hosts while the figure is 20% for path changes. Path changes tend to have wider impact: 57% of path changes can be confined to two ASes and 50% of them can be confined to within three hops, while the respective figures are 78% and 70% for outages. Path changes have a much milder effect on RTTs than outages while they both can incur high loss rates.

- Our measurements suggest less opportunity for indirection-based resiliency than previous studies: alternative routes are available only 43% of the time, and a significant fraction of them suffer from high latency inflation. These results stem from most outages occurring nearer the edge of the network than the core; redundancy is less available, and less practical when it is available.

We have shown that PlanetSeer provides an effective means to detect large numbers of anomalies with broad coverage, especially in the case of wide-area services that cannot rely on cooperation from one endpoint. In addition to the detection rate, the short delay between emergence and detection allows us to capture anomaly behavior more effectively, and our distributed framework provides improved characterization.

Acknowledgments

This research was supported in part by NSF grant CNS-0335214. We would like to thank KyoungSoo Park for his effort in keeping CoDeeN operational during this work. We thank our shepherd, Miguel Castro, for his guidance and helpful feedback, and we thank our anonymous reviewers for their valuable comments on improving this paper.

References

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. *ACM SOSP*, Oct. 2001.

[2] S. Banerjee, T. G. Griffin, and M. Pias. The interdomain connectivity of PlanetLab nodes. In *Passive and Active Measurement Workshop*, April 2004.

[3] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Mawrzoniak. Operating system support for planetary-scale network services. In *USENIX/ACM NSDI*, Mar. 2004.

[4] M. Caesar, L. Subramanian, and R. H. Katz. Route cause analysis of Internet routing dynamics. In *Tech Report UCB/CSD-04-1302*, 2003.

[5] D.-F. Chang, R. Govindan, and J. Heidemann. The temporal and toplogical characteristics of BGP path changes. In *IEEE ICNP*, Nov. 2003.

[6] M. Dahlin, B. Chandra, L. Gao, and A. Nayate. End-to-end WAN service availability. *ACM/IEEE Trans. Netw.*, Apr 2003.

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

[8] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: methodology and experience. *ACM SIGCOMM*, Aug. 2000.

[9] A. Feldmann, O. Maennel, Z. Mao, A. Berger, and B. Maggs. Locating Internet routing instabilities. In *ACM SIGCOMM*, Aug 2004.

[10] U. Hengartner, S. Moon, R. Mortier, and C. Diot. Detection and analysis of routing loops in packet traces. In *ACM IMW*, 2002.

[11] IANA. Special-use IPv4 addresses. RFC 3330.

[12] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *ACM IMW*, Nov 2002.

[13] J. Postel. Internet control message protocol. RFC 792.

[14] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. In *ACM SIGCOMM*, Sep 2000.

[15] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental study of Internet stability and wide-area backbone failures. Technical Report CSE-TR-382-98, University of Michigan, 1998.

[16] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *ACM SIGCOMM*, 2002.

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

[18] U. of Oregon RouteViews Project. http://www.routeviews.org.

[19] V. Paxson. End-to-end routing behavior in the Internet. In *ACM SIGCOMM*, Aug 1996.

[20] V. Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Trans. Netw.*, 7(3), 1999.

[21] S. Savage, A. Collins, and E. Hoffman. The end-to-end effects of Internet path selection. *ACM SIGCOMM*, Aug. 1999.

[22] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A facility for distributed Internet measurement. *USITS*, March 2003.

[23] L. Subrmanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. *IEEE INFOCOM*, June 2002.

[24] Traceroute.Org. http://www.traceroute.org.

[25] V. Paxson and M. Allman. Computing TCP's retransmission timer. RFC 2988.

[26] L. Wang, V. Pai, and L. Peterson. The effectiveness of request redirection on CDN robustness. In *OSDI*, Dec. 2002.

[27] Y. Zhang, N. Duffield, V. Paxson, and S. Shenkar. On the constancy of Internet path properties. *ACM IMW*, Nov. 2001.