Detection and Analysis of Routing Loops in Packet Traces

Urs Hengartner, Sue Moon, Richard Mortier, Christophe Diot

Abstract—

Routing loops are caused by inconsistencies in routing state among a set of routers. They occur in perfectly engineered networks, and have a detrimental effect on performance. They impact end-to-end performance through increased packet loss and delay for packets caught in the loop, and through increased link utilization and corresponding delay and jitter for packets that traverse the link but are not caught in the loop.

Using packet traces from a tier-1 ISP backbone, we first explain how routing loops manifest in packet traces. We characterize routing loops in terms of the packet types caught in the loop, the loop sizes, and the loop durations. Finally, we analyze the impact of routing loops on network performance in terms of loss and delay.

I. Introduction

Inconsistencies in routing state cause *routing loops* in the Internet. A routing loop causes packets trapped in the loop to be delayed or discarded, depending on whether or not the packet exits the loop. Internet routing protocols aggregate routable addresses into prefixes, and so many packets can be caught in a loop involving a given prefix, allowing routing loops to impact a substantial quantity of traffic.

We present a new method for detecting routing loops from packet traces, and apply our method to a selection of packet traces taken from Sprint's IP backbone.

Routing loops cause packets caught in the loop to traverse the same link in the network multiple times, and to show up as replicas of the original packet in the trace. We term a set of replicas with a suitably decrementing TTL (Time-To-Live) a *replica stream* and identify replica streams in the traces. As one routing loop may cause many packets to be replicated, we then merge replica streams caused by the same routing loop.

There are two forms of routing loop that occur: *transient* and *persistent*. Transient loops occur as part of the normal operation of the routing protocol due to the different delays in the propagation of information to different parts of the network. Transient loops should be resolved without intervention as the routing protocols converge. Persistent loops arise for a number of reasons, perhaps most commonly router misconfiguration. Eliminating a persistent loop thus requires human intervention. Persistent loops are difficult to analyze for two reasons. First, they are rare. Second, they often occur across multiple ASs (Autonomous Systems) and so require cooperation of many network operation groups to be analyzed [1]. Consequently we do not discuss persistent loops in this work, but focus rather on analysis of transient routing loops.

We propose a loop detection algorithm and use it on packet traces taken from a selection of Internet links to analyze the impact of routing loops on the performance of the network. Losses due to routing loops remain very small, but for brief moments loops can cause the loss rate to increase significantly. Routing loops from which packets do not escape have no effect on the delay performance of the network. However, the delay of packets that do escape a routing loop is increased by 25 to 1300 ms, an increase in similar magnitude to the normal end-to-end delay of the Internet.

The remainder of this paper is organized as follows. In Section II we demonstrate how transient routing loops can arise even when the network is operating normally and in Section III we review related work. In Section IV we present our algorithm to detect loops from packet traces. In Section V we apply the algorithm to packet traces collected from a selection of Internet links. Section VI concludes the paper.

II. Routing Loop Evolution

A router in today's Internet belongs to an AS (Autonomous System), and different ASs typically correspond to different administrative domains. Routing information *within* an AS is distributed via intra-domain routing protocols such as OSPF and IS-IS [2], [3]. Routing information *between* ASs is distributed via inter-domain routing protocols, of which BGP is the only currently deployed example [4]. Finally, BGP is also used to propagate external routing information within an AS. When used for the former purpose, it is referred to as E-BGP, and when used for the latter purpose as I-BGP.

Routing protocols distribute information through the network so that all routers in the network eventually converge to a consistent view of the network. Consequently, routing loops occur either as a result of temporary inconsistency arising during the convergence process, or as a result of more permanent inconsistency due to misconfiguration or route oscillation. As explained earlier, in this paper we focus on the analysis of transient loops, and leave analysis of persistent loops to future work.

A. Transient Loops

A simple example of the evolution of a transient routing loop is shown in Figure 1. Consider a small network of three nodes, R1, R2, and R3. These nodes are connected, with solid lines representing physical links, and dotted lines representing the flow

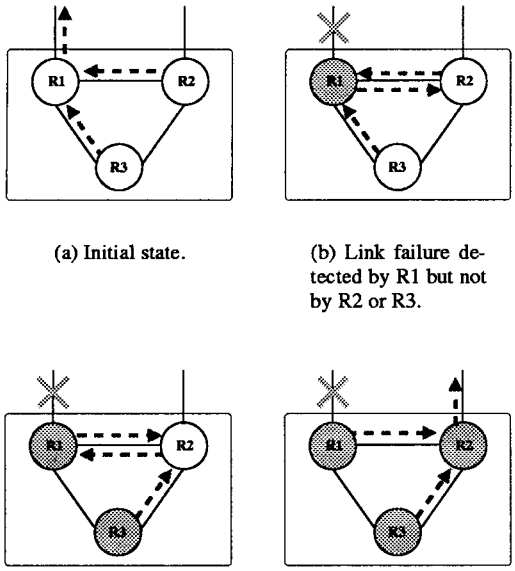
Urs Hengartner is with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Email *uhengart@cs.cmu.edu*

Sue Moon is with Sprint ATL, 1 Adrian Court, Burlingame, CA 94010, USA. Email *sbmoon@sprintlabs.com*

Richard Mortier is with Microsoft Research Ltd., 7, JJ Thomson Avenue, Cambridge UK CB3 0FB. Email *mort@microsoft.com*

Christophe Diot is with Sprint ATL, 1 Adrian Court, Burlingame, CA 94010, USA. Email *cdiot@sprintlabs.com*

IMW'02, Nov. 6-8, 2002, Marseille, France



(a) Initial state.

(b) Link failure detected by R1 but not by R2 or R3.

(c) R1 and R3 agree but R2 is confused.

(d) Convergence achieved.

Fig. 1. Scenario for a transient routing loop.

of traffic between nodes. Nodes with consistent routing state are shown with the same shading. Traffic from all three nodes to other networks initially travels through R1 as depicted in Figure 1(a); R2 has also advertised an alternative route to other networks.

The link connecting R1 to other networks fails and so R1 is the first to detect the failure. Traffic should now flow to other networks via R2, as shown in Figure 1(d). However, until R2 learns of the failure of R1, it continues to direct traffic for other nodes in the network to R1. Since R1 knows that its link has broken and an alternative path is available via R2, it forwards this traffic back to R2 resulting in traffic looping as shown in Figure 1(b).

In Figure 1(c) the updated information has reached R3 before R2. Thus R3 begins sending traffic destined for other networks to R2. As R2 is still unaware that R1 has failed, it continues to send this traffic to R1, which returns it to R2. Finally, in Figure 1(d), information about the link failure reaches R2 and so it stops routing traffic for other networks to R1, and uses its own link instead. Thus the loop has disappeared.

The scenario depicted in Figure 1 is of a routing loop involving just two routers separated by a single hop. However, routing loops involving more than one hop can also occur, since the only requirement for a loop to form is that the routing information in two routers becomes unsynchronized.

Transient loops arise in the Internet since all of BGP, OSPF and IS-IS admit such transient inconsistencies, in addition to converging at different speeds. In the case of BGP, transient loops can arise due to:

- A peer withdrawing one or more prefixes that are also advertised via other peers, requiring its neighbors also to withdraw them and switch to alternative routes.
- A connecting link to a peer going down, potentially causing all BGP sessions carried on that link also to go down. All prefixes advertised on such sessions must then be withdrawn.
- A prefix routed via one router being newly advertised by another where the new route is to be preferred.

In OSPF and IS-IS transient inconsistencies arise between the link-state databases of participating nodes due to delays in forwarding updated link-state information.

B. Loop Resolution

The duration of a transient loop is directly related to the convergence time of the routing protocol in question. It has been shown that BGP convergence is not fast, in certain cases taking on the order of tens of minutes [5]. The convergence times of link-state protocols such as OSPF and IS-IS are affected by a number of factors: the time taken to detect link failure; the time taken to flood new topology information; and the time taken to perform any required shortest path recomputation [6]. More recent findings show that implementation and configuration dependent timer values and FIB (Forwarding Information Base) update times add significantly to the overall convergence time [7].

The time it takes a router to detect link failure depends on the interface type, but is usually on the order of milliseconds for point-to-point links. In any case, it is limited by a timer taking values on the order of tens of seconds. Following failure detection, new information is flooded and shortest paths recomputed where necessary. Several factors affect these times: damping algorithms are used to prevent spurious updates, potentially delaying the propagation of updated information; the diameter of the network affects the time taken to flood new information; and the implementation of the shortest path algorithm affects the time taken to perform the recomputation. The end result is that such link-state protocols typically converge in seconds.

One should note an important difference between transient loops arising from the EGP (Exterior Gateway Protocol) and those arising from the IGP (Internal Gateway Protocol). In the former case, loops are caused by an event external to the AS, whereas in the latter they are caused by changes local to the AS. Thus, it is harder to control the impact on a network of loops due to the EGP since the causes are typically outside the control of the network operator.

III. Related Work

As routing tables have grown exponentially in recent years [1], the performance and behavior of routing protocols have become critical issues in network performance, management and engineering [8]. It is possible to detect certain BGP misconfiguration errors by examination of conflicting short-lived updates of origin AS changes, but such detection is difficult to automate [1].

Paxson has studied routing loops using end-to-end traceroute measurements collected in 1994 and 1995 [9]. Although his work focuses on persistent loops, he detects a few transient loops, and conjectures that such loops are caused by link failure information about single link failures propagating through the network.

[1] See `http://www.telstra.net/ops/bgp/index.html` for example.

Labovitz et al. have studied the impact of delayed routing convergence on connectivity, packet loss, and latency [5]. Measurements were taken by injecting path failures into the network, and using ICMP echo ('ping') packets sent to a set of web sites beginning 10 minutes before the path failure event. They found that convergence times in inter-domain routing are on the order of minutes, and that end-to-end loss and latency increase significantly during convergence. More recent work by Freedman shows that there is a correlation between BGP churn and UDP packet losses [10].

As an alternative to such work, we focus on the detection of routing loops from packet traces captured on backbone links. Loop detection using end-to-end tools such as traceroute is error-prone and cannot help assess the impact on traffic not looped. It is also hard to successfully detect transient loops with such techniques. By using off-line analysis of traces containing the header of every packet traversing a link, we can detect routing loops as manifest by the same packet traversing the link many times in a short period. This gives us more complete information about routing loops, which allows us to go on to quantitatively capture the performance degradation due to these loops.

In the next section we describe how we detect routing loops from packet traces, and go on to analyze their duration, composition in terms of packet types, and length in terms of router hops.

IV. Routing Loop Detection

This section presents our algorithm for detecting loops from packet traces. Results of this algorithm and verification of the results are presented in following sections.

A. Detection Algorithm

Routing loops cause replicas of a packet to cross a fixed point in the loop. A set of replicas is called a *replica stream*, and it corresponds to multiple instantiations of a packet on a single link. We use these replica streams to detect routing loops. The algorithm has three steps:

Step 1. Detect replicas.

Step 2. Validate replica streams (sets of replicas).

Step 3. Merge replica streams into routing loops.

The result of this algorithm is a collection of merged replica streams. Each of these merged sets indicates that a routing loop occurred between the first packet in the set and the final packet in the set. We now discuss each step in the algorithm in detail.

A.1 Detecting Replicas

We detect replicas of the same packet crossing a link by determining whether two packets are replicas of a looping packet.

Specifically, two packets a and b where b is observed after a are considered to be replicas of a single looped packet if their headers are identical except for the TTL and IP header checksum fields; their TTL values differ by at least two; and their payloads are identical. The IP identification field in the IP header serves to distinguish packets that are looped from those that are simply part of the same 'connection.'

The packet traces used in our evaluation contain only the first 40 bytes of an IP packet. For a TCP packet with neither IP nor TCP options, a trace thus includes the IP and TCP or UDP headers, but not the TCP or UDP payloads. Consequently, we assume two packets have identical payloads if they have identical TCP or UDP checksums.

A.2 Validating Replica Streams

For a set of replica packets to be evidence of a routing loop, it must satisfy two conditions.

First, we eliminate those having only two elements. Such affects are often due to the link-layer injecting duplicate packets. For example, the sender may fail to drain the packet in a token ring, or a misconfigured SONET protection layer may transmit packets on both the working and protection links.

Second, we verify that all packets to the same prefix are part of replica streams during the time of the proposed routing loop. We merge replicas of packets with destination addresses with the same 24 bit prefix into single replica streams, as 24 bits is the longest prefix currently honored by tier-1 ISPs.

Since routing loops signify a transition in routing state, the longest prefix match may change before and after the transition. Merging replica streams according to the relevant entries in the routing table does not make sense, unless we can reconstruct the exact sequence of protocol state changes in a router from receiving a route update to the updating of the FIB on the interface from which the packet trace was collected. To do so would require complete routing tables, or better, complete update logs, from the particular routers from which traces were taken; this information is not available for the traces in question.

If a packet with the same destination subnet as a replicated packet does not itself belong to a replica stream, then other replicas observed at that time cannot be due to a routing loop, since the loop should affect all packets to the destination in question.

Replicas not satisfying these conditions are ignored. We term the resulting sets of replica packets *replica streams*; each replica stream originates from a single unique packet.

A.3 Merging Replica Streams

At this point we have identified replica streams that indicate a routing loop could be in progress. Consequently, to gain more insight about the duration of routing loops causing replica streams, we now merge replica streams likely to be due to the same routing loop.

First, we merge replica streams that overlap in time and have identical destination address prefixes. Such replica streams have high probability of being caused by the same routing loop.

However, simply merging replica streams in the same subnet underestimates the length of a routing loop if there are points in the loop where there are no replica packets to detect. Consequently, we also merge replica streams that occur less than one minute apart provided that (as in step 2) the resulting merged replica stream does not overlap with packets to the subnet that are not looped [2]. As we validate each merged replica stream against routing updates for a possible loop, it can be considered to be from a single loop and is referred as such in the remainder of the paper.

[2] We also tried 2 and 5 minute intervals, but found the number of merged replica streams not to be significantly different to using 1 minute intervals.

V. RESULTS

We use a set of packet traces collected from Sprint's tier-1 Internet backbone [11]. They were gathered in parallel over multiple uni-directional OC-12 links on the east coast of the US. All of the links connect two distinct ASs, each under separate administrative control. The first two traces started at 13:00 GMT on November 8th, 2001, and the second two at 20:00 GMT on February 3rd, 2002.

Trace	Length (hours)	Avg BW (Mbps)	Packets Total (10^6)	Looped Packets
Backbone 1	24	1	50	2 419 792
Backbone 2	7.5	243	1 677	1 987 309
Backbone 3	11	2.2	20	337 570
Backbone 4	11	107	1 350	364 230

TABLE I
DETAILS OF TRACES.

Table I provides the length, average bandwidth, number of packets, and number of individual looping packets found in the traces. Backbones 1 and 3 have very low link utilization. The absolute number of looped packets on Backbone 2 is similar to that on Backbone 1, but lower in relative terms since Backbone 2 has a much higher average bandwidth of 243 Mbps giving a total of 1.7 billion packets.

A. General Observations

We consider three metrics in our analysis of the routing loops: the amount by which replicas have their TTL decremented (the *TTL delta*), the number of replicas in a replica stream, and the inter-replica spacing caused by the loop.

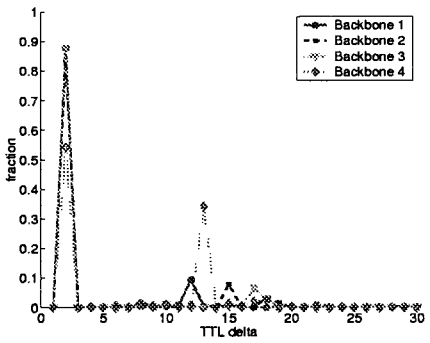


Fig. 2. TTL delta distribution.

The difference in the TTL value of two sequential replicas indicates the number of nodes involved in the routing loop. In Figure 2, we give the distribution of the TTL delta of all replica streams detected in a trace. In Backbones 1, 2, and 3 the majority of replica streams have a TTL delta of 2, and 5 to 10% of have TTL deltas between 12 and 18. Backbone 4 has has approximately 55% and 35% of replica streams with TTL delta of 2 and 13, respectively.

A TTL delta of 2 typically corresponds to routing inconsistencies between two adjacent routers. As routing updates are propagated via flooding in IGP (Interior Gateway Protocols) and to fully meshed peers in I-BGP (Internal-Border Gateway Protocol), two adjacent routers at the boundary of the update propagation are most likely to create a transient loop, and that is why a TTL delta of 2 is most common. As it requires only two routers to become unsynchronized to create a loop, and the updates can propagate at different speeds to different parts of the network, the distance between those two routers can be greater than one hop, creating replica streams with a TTL delta larger than 2.

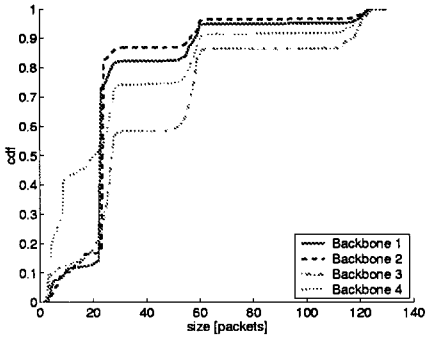


Fig. 3. CDF of the number of replicas in a replica stream.

Figure 3 shows the CDF of the number of replicas in a replica stream. For all of the backbone links there are jumps at 30 and 60 replicas. These jumps are due to the common TTL delta on these links being 2, in conjunction with 64 and 128 being popular initial TTL values for Linux and Windows 2000 respectively. A packet with an initial TTL of 64 will cause approximately 30 replicas to be generated as it traverses a loop with a TTL delta of 2, before it is discarded.

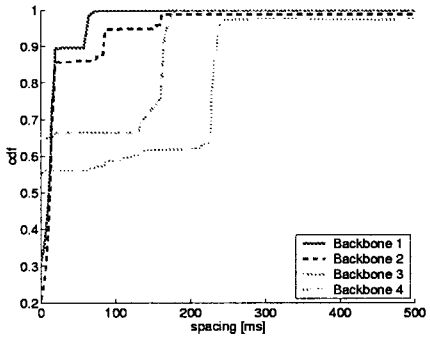


Fig. 4. CDF of inter-replica spacing time.

Finally, in Figure 4 we plot the CDF of inter-replica spacing in time. We use an average of all inter-replica spacing times calculated per replica stream. In Backbones 1 and 2 about 90% of replica streams have inter-replica spacing times less than 80 ms and almost 100% under 150 ms. In Backbones 3 and 4, 65% and 55% of replica streams have less than 10 ms, and almost all less than 150 ms and 220 ms, respectively. The common step shapes detected in all traces are due to the dominant TTL delta values in the traces in conjunction with the common initial TTL values of 64 and 128. The larger the TTL delta is, the more hops a replica traverses, resulting in longer inter-replica spacing time. We identify those replica streams with TTL deltas larger than 10

to have inter-replica spacing times larger than 50 ms.

The inter-replica spacing gives an indication of the time taken to traverse the loop, and hence the minimum delay imposed on packets caught in a loop. The increased packet delay due to a loop can be explained from the initial TTL value and the inter-replica spacing. The higher the initial TTL value, and the longer the inter-replica spacing time are, the longer delay the packet experiences. Consequently, a larger contribution is added to the end-to-end delay if the packet ever escapes the loop.

B. Replica Properties

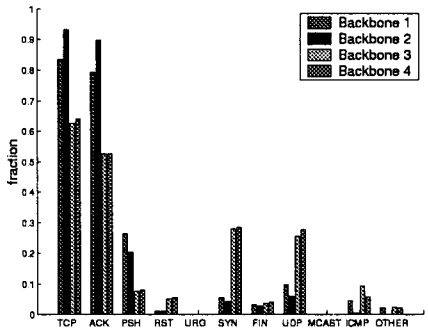


Fig. 5. Traffic type distribution of all traffic.

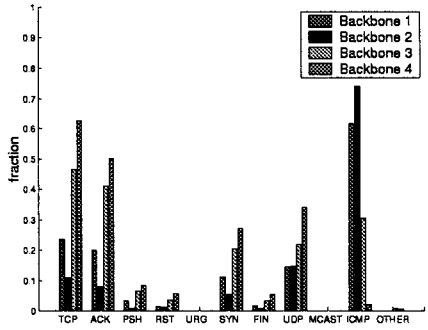


Fig. 6. Traffic type distribution of looped traffic.

We now investigate the properties of the packets caught in the routing loops. Figures 5 and 6 plot the distribution of the various packet types for all traffic on the link, and for all replica streams respectively. Internet traffic consists principally of TCP, UDP, and ICMP packets. Note that a single replica can show up in multiple categories, a TCP SYN-ACK being listed in all of the TCP, SYN, and ACK categories for example. In all traces TCP packets take up more than 80% and UDP packets take up about 5% to 15% of the total packets on the link. TCP SYN and FIN packets take less than 10%.

The proportion of the total looped traffic made up of SYN packets is higher than the proportion of SYNs in the total traffic (looped and not looped) on the link. Since looped packets tend to expire in the loop, the looped SYNs do not successfully create a TCP connection and so they do not cause further TCP traffic to be transmitted. TCP does not transmit data traffic if the connection is not made or if a packet is lost, but UDP has no such restriction. Effectively, if a TCP SYN is lost, that prevents the traffic that would have been associated with that connection being transmitted, whereas if a UDP packet is lost, that loss has no effect on the transmission of other UDP packets.

Backbones 1, 2, and 3 have a high proportion of replica streams involving ICMP packets. Most are 'echo request' and 'time exceeded' packets, but there is also one host that generates ICMP packets seen in Backbones 1 and 2 with multiple 'reserved' type fields. Although this is unusual behavior, we are confident that the corresponding replicas are due to loops as other packets sent to the same destination prefix by other hosts also loop.

In general, we hypothesize that the high proportion of looped ICMP traffic is caused respectively by end hosts pinging and tracerouting when they see packets being lost, and by the routers dropping packets that expire due to loops. Presence of such streams of ICMP traffic might provide a strong indication that a loop is in progress.

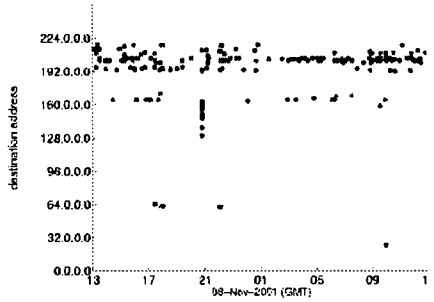


Fig. 7. Destination addresses of replica streams in Backbone 1.

Figure 7 shows the destination addresses of replica streams over time for a subset of the traces. This time-series plot reveals that a wide spectrum of addresses are affected by routing loops during the packet trace collection. Not all lengths of IP prefix are equally affected. There are more looped packets in the Class C IP addresses (192.0.0.0 to 223.255.255.255), either due to this portion of the address space being more highly utilized, or to link-specific traffic dynamics.

C. Duration of Routing Loops

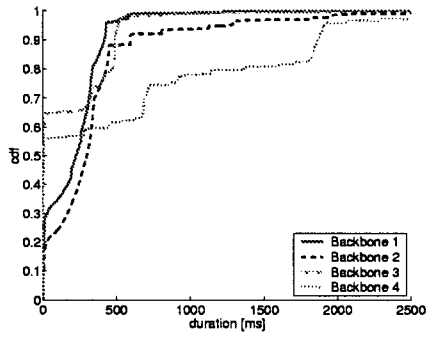


Fig. 8. CDF of replica stream duration.

We now consider the duration in time of replica streams, and finally of detected routing loops.

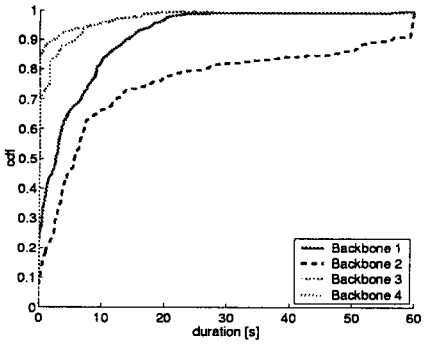


Fig. 9. CDF of routing loop duration.

In Figure 8 we plot the CDF of the duration of replica streams, given by the time difference between the first and last replicas in a replica stream. For Backbones 1 to 3, most replica streams last less than 500 ms, and as in Figure 3, there is a stepwise pattern due to the TTL delta of the loop and the initial TTL of the looped packet. However, this pattern is less distinct than the previously observed step patterns, most probably due to the increased random noise such as queuing delay that affects the lifetime of a packet in a loop. In Backbone 4 there are three noticeable step increases at around 10, 700 and 1800 ms. Recalling from Figure 3 that Backbone 5 has three dominant initial TTL values, the steps are self-explanatory.

Table II presents an overview of the results, giving the number of raw replica streams and the number of routing loops after all merging as described in Section IV-A.3 has taken place. This table shows that the many replica streams that occur in the traces typically merge well, and are caused by comparatively few routing loops.

Trace	Replica Streams	Routing Loops
Backbone 1	79257	852
Backbone 2	70144	413
Backbone 3	7377	1485
Backbone 4	11997	1568

TABLE II
NUMBER OF ROUTING LOOPS.

Figure 9 plots the CDF of routing loop durations after replica streams have been merged following the algorithm described in Section IV. In this figure, we can see that 90% of the loops are short in duration, lasting less than ten seconds in Backbones 3 and 4. The convergence time in routing state involves a change detection (a link failure or a new link), update propagation, and recalculation of the shortest paths. Our finding that the loop duration is mostly under 10 seconds is in agreement with the parallel work that the convergence time after a link failure in the current Internet is between 5 to 10 seconds [7]. In case of Backbones 1 and 2, we observe longer loops. As future work, we will look into the causes behind observed replica streams and merged loops, and adderess routing behaviors that have caused these long-lasting loops.

VI. SUMMARY

We showed how to detect routing loops from packet traces. We described in detail a routing loop detection algorithm, and explained how we verified its results. We applied the algorithm to packet traces from the Sprint IP backbone network, and analyzed replica streams in terms of TTL deltas, numbers of packets, and inter-replica spacing times.

Routing loops of any sort can lead to significant performance degradation. We analysed the impact of routing loops on link utilization, and report that it could contribute up to 90% of packet loss per minute, depending on the trace, eventually impacting the queueing delay of other packets [12]. However, losses due to routing loops remain very small for most of the time. Routing loops also significantly increase the delay experienced by packets that escape. Between 0.6% and 11% of looping packets escape their loop, and incur between 25 ms and 1300 ms extra delay. We note that those packets that escape a loop can be delivered out-of-order.

Analysis of merge routing loops showed that the majority of them last shorter than 10 s, while 30% of loops on a subset of links last longer than 10 s. Although our verification of loops provided plausible mechanisms to correlate replica streams, the routing behaviors behind the loops remain unknown. In further work, we are extending our data collection techniques to include complete BGP and IS-IS routing data. This will enable a more detailed analysis of routing loops through a single link to be performed, and allow us to provide explainations of the causes and effects of routing loops.

REFERENCES

[1] R. Mahajan, D. Wetherall, and T. Anderson. Understanding bgp misconfiguration. In *To appear in Proceedings of SIGCOMM*, Pittsburgh, PA, August 2002.

[2] R. Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*. Addison-Wesley Professional Computing Series, 1999.

[3] J. Moy. *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley, 1998.

[4] J. W. Stewart III. *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley Networking Basics Series, 1999.

[5] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. In *Proceedings of SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[6] C. Alaettinoglu, V. Jacobson, and H. Yu. Towards milli-second IGP convergence. IETF draft, November 2000.

[7] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in a large IP backbone. In *To appear in Proceedings of SIGCOMM IMW 2002*, Marseille, France, 2002.

[8] G. Huston. Analysis of the Internet's BGP routing table. *Internet Protocol Journal*, 4(1), March 2001.

[9] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):610–615, November 1997.

[10] A. Freedman. Edge/core update propagation, churn vs. performance. Presentation at Internet Statistics and Metrics Analysis workshop (ISMA), December 2001.

[11] C. Fraleigh, S. Moon, C. Diot, B. Lyles, and F. Tobagi. Packet-level traffic measurements from a tier-1 IP backbone. *Sprint ATL Technical Report TR01-ATL-110101*, November 2001.

[12] U. Hengartner, S. Moon, R. Mortier, and C. Diot. Detection and analysis of routing loops in packet traces. *Sprint ATL Technical Report TR02-ATL-051001*, May 2002.