On the inadequacy of link connectivity monitoring

Mohit Saxena and Ramana Rao Kompella
Computer Science Department
Purdue University
Email: {msaxena,kompella}@cs.purdue.edu

***Abstract*—Internet backbone networks are constantly evolving along several dimensions such as technology, protocols, and features. The rapid rate of this evolution often places a tremendous amount of burden on network operators and router vendors to cope with both unanticipated as well as complicated failure scenarios. While routing protocols are designed to detect these failures and respond by switching to alternate paths, we argue that current routing protocols (*e.g.*, OSPF) do not completely verify all data plane paths properly. Thus, there potentially exist failure scenarios that are not detected by them and thereby recover from. To alleviate this problem, we show how a simple extension to OSPF, wherein each router injects probes to a 2-hop neighbor, can easily verify all the data plane forwarding paths within the adjacent router. While we point out the extension in the context of OSPF, we believe our approach is generally applicable to other control plane protocols as well. Using Rocketfuel topologies, we quantify the resulting overhead of injecting these 2-hop probes for routers in real ISPs.**

I. Introduction

Today's operational backbone networks are extremely large and complicated. A typical backbone consists of about a thousand routers supported by access and core transport networks with several hundred thousand network elements. The rapid pace of deployment of these devices together with the constant evolution in technology (e.g., ultra long-haul optics), protocols (e.g., MPLS) and applications (e.g., VoIP) often results in a wide variety of complex faults and impairments.

With the increasing reliance on electronic commerce and convergence of several other communication services to IP-based networks, network outages are becoming increasingly costly, in terms of lost revenue for companies relying on the infrastructure and for those that are managing it. Not surprisingly, a great deal of effort is often expended by ISPs to rapidly detect, diagnose and recover from faults in the backbone networks.

A certain amount of fault tolerance is built into the network architecture. This fault tolerance is often achieved via over-provisioning, whereby the network is engineered to have enough spare capacity to carry fail-over traffic in case certain primary links fail. This basic fault tolerance is quite effective at masquerading many of the common failure modes, such as fiber cuts or optical layer failures, observed in the network. This effectiveness, however, hinges mainly on the ability of the routers to effectively detect the presence of a failure and subsequently re-route the traffic via other functional paths.

Routers, therefore, periodically generate OSPF (or IS-IS) "Hello" messages to their neighbors both to advertise their presence as well as check connectivity between the two OSPF instances (on the routers). The lack of receipt of a certain number of consecutive messages indicates loss in connectivity. This triggers link state advertisements propagated throughout the network, and routers avoid sending traffic through this failed link by computing alternate paths.

While the basic mechanism using these Hello messages works well, and has indeed served its purpose quite well so far, with the advent of newer (possibly distributed) router architectures, we argue that this mechanism is not fundamentally capable of detecting all the faults it has been originally designed for. In particular, these Hello probes originate at a router's CPU and terminate at the adjacent router's CPU; the path traversed by these probes may be significantly different from the regular data-plane forwarding paths within routers. Thus, there potentially exist failure scenarios where these probes do not automatically detect forwarding problems within the router and thus respond to.

Lacking intrinsic capability to detect many such faults, ISPs resort to fault detection and diagnosis outside of the network with the help of measurement servers connected directly to provider edge routers. Each server typically injects active probes to all the other servers to check for liveness and other performance problems, and take any remedial actions in case any problems exist. Such a feedback loop often provides a second layer of resiliency in the network and is implemented in the management plane.

While this external feedback loop enables operators to manage faults not handled directly by the routers, in many cases, this feedback loop in the management plane is quite slow. For example, using active probes allows detection of failures in the order of a few seconds. Typically, the fault needs to be localized before it can be repaired, which can take a few minutes. Finally, the repair (at least a quick fix) can be performed in a matter of minutes. Thus, the overall duration of the failure, from detection to repair can take a few minutes, which is quite large.

It is, therefore, important to consider mechanisms that can quickly detect data plane failures, and more importantly be coupled with the routing protocols so that they can react to the problems once detected. In this paper, we present a simple mechanism whereby routers can detect all forwarding problems in the network. We start with the OSPF Hello messages as our point of departure. We observe that these messages do not automatically pass through the forwarding paths within routers. We, therefore, devise a new set of probes, similar to the OSPF probes, except between a router and all it's

2-hop neighbors. These probes would essentially pass through an adjacent router just as normal data packets would, and hence verify the corresponding forwarding path within the router. This extension, while simple, can provide an effective feedback mechanism to all routers similar to the normal OSPF Hello messages for link problems.

Thus the main contributions of this paper are as follows:

- Using the typical architecture of a router, we show how today's OSPF Hello probes are insufficient to cover all the forwarding paths in the network.
- We propose novel 2-hop probes as a scalable mechanism to allow routers to verify forwarding plane liveness within routers.
- Using Rocketfuel topologies, we evaluate the additional overhead caused by these new 2-hop probes.

The rest of the paper is organized as follows: First, we discuss how current routing protocols detect link-level failures and why we believe these mechanisms are incomplete in Section II. We then discuss our approach of using 2-hop segments as a way to alleviate this problem in Section III. Finally, we show the overhead of using 2-hop segments on the router architecture using Rocketfuel topologies in Section IV.

II. Native fault detection

In this section, we discuss the native fault detection mechanisms employed by classical intra-domain routing protocols such as OSPF, and show why they fail to capture all inherent data-plane failure modes within routers. In particular, using a typical router architecture, we expose the particular forwarding paths not completely verified by the OSPF active probes.

A. Link connectivity probes

Simply speaking, a network can be viewed as a graph with a bunch of nodes connected via edges. In such a simplistic representation, failures can occur in two flavors—link and node failures, both of which can be easily detected using simple probes between every pair of adjacent nodes. Indeed, popular link-state routing protocols, such as OSPF [10] and IS-IS [2] employ such a simple mechanism using periodic Hello messages to verify both these types of failures.

When a router is switched on, it initializes the routing protocol data structures and waits for indications from the lower layer protocols to determine which of its interfaces are functional. OSPF Hello probes are then sent out on each of the functioning interfaces to discover and maintain neighboring relationships. For each neighbor and interface, separate data structures and state machines are maintained and are regularly updated. Hello probes are sent periodically every *HelloInterval* seconds on all interfaces [10]. For physical networks having multicast or broadcast capability, they are periodically sent to the IP multicast address *AllSPFRouters*.

Similarly, when a router receives a Hello packet on an interface, a *HelloReceived* event is triggered. Each Hello packet also contains a list of neighbors of the router which originated the packet. This list of neighbors in the Hello packet is examined and if the router itself appears in this list, then it is assumed that the Hello packets have been received on both ends recently (enters a *2-WayReceived state*). A single-shot *InactivityTimer* is also restarted now for this neighbor, whose duration is *RouterDeadInterval* seconds. Firing of this timer in future will indicate that no Hello Packet has been seen from this neighbor recently. In this way, each router periodically keeps track of live OSPF instances running on its neighbors.

Apart from the firing of the InactivityTimer, a router can also detect loss of communication with a neighbor in three other ways. If a Hello packet is received in which the router itself is not mentioned indicates that the communication is no longer bidirectional now. While such an event may be triggered due to administrative changes to the status of a link, still, the routing protocol eliminates that link from its topology similar to when a link failure occurs.

Any change in the neighbor list of any interface at a router causes an update to its link state database. Other routers in the network are notified of this change using the Link State Update packets which implement the flooding of Link State Advertisements (LSAs). It is important to note that both the Hello packets and Link State Update packets travel only one hop from their originator. This is ensured by setting their *IP TTL* field to 1 in the beginning. An important difference between these two packets is that Hello packets are sent to all the neighbors associated with an interface, but the Link State Updates are sent to only those neighbors which are *adjacent* to this router [10]. Adjacencies are created with selected neighboring routers to exchange routing information.

Next, we explain the architecture of a typical router in order to describe the path traversed by these Hello probes within the router.

B. Typical router architecture

We show the architecture of a typical router in Figure 1. Routers have several hardware and software components typically classified into one of two planes based on their functionalities. Forwarding plane (or data plane) refers to the part of the router architecture which is responsible for packet forwarding, *i.e.*, the act of receiving packets on one interface and sending them on the same or other interfaces through the internal switching fabric (as shown in Figure 1). Control plane, on the other hand, comprises of the routing protocols that gather and maintain network topology information and configures the forwarding plane appropriately.

In order to achieve high performance and throughput, specialized hardware is used for forwarding plane operations, so that they do not contend for resources such as processing power, shared with the control plane. While forwarding can be done using a central forwarding engine, for scalability reasons, many modern routers employ a distributed architecture, with the forwarding operations spread across different line cards [12]. Packets received on an interface are handled by a forwarding engine local to that line card.

The path from one ingress line card to egress line card is shown in Figure 1. When a packet is received on the ingress, the packets are first recovered from the optical medium on

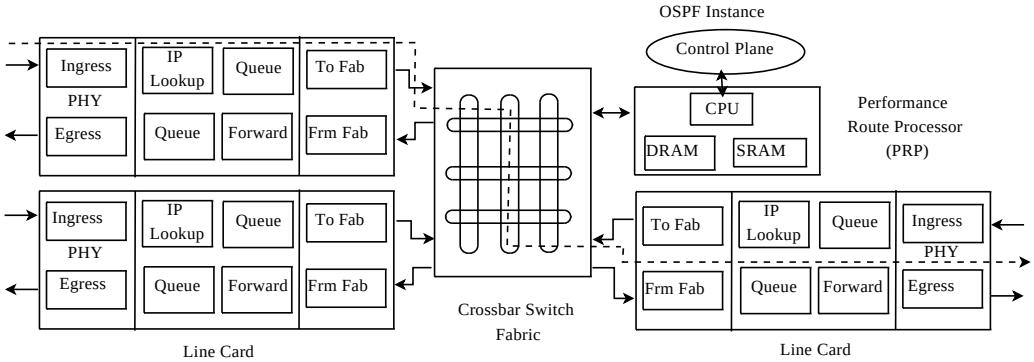


Fig. 1. Typical router architecture.

the PHY interfaces. A specialized module on the line card computes a forwarding decision for the packet to locate which egress interface the packet needs to be switched to. Typical routers today are input-queued and they contain virtual-output queues (VOQs) at each line card [9]. Many routers use a crossbar switch fabric, in which the cross-points are configured using a switch scheduler based on the current queue occupancies of all the VOQs.

The exact architecture of a router may differ in terms of the sequence of functions packets are exposed to, and the complexity of each function in turn. For example, an optional traffic shaper module may be present to shape the traffic to certain specifications (*e.g.*, conform to a specific delay). There can also be a QoS scheduler devise that can prioritize among several priority classes such as DiffServ [1]. Researchers have proposed several mechanisms to scale the switch fabric even further, either with the help of optics [7] or with several stages of switching [6]. Thus, in essence, there are several data paths within a router that potentially traverse different set of functions within the router. Thus, depending on the particular type of packets, therefore, there are several data paths within the router.

In many routers, the route processor itself is a stand-alone line card connected via the switch fabric (as shown in Figure 1). So, all the control plane messages destined to the router IP address, will be directed by the forwarding engine towards the route processor. Thus, just like data packets, they might visit several series of functions, before these packets end up at the route processor, where appropriate control plane actions can be performed. Similarly, when a route processor transmits a packet (*e.g.*, the OSPF Hello messages to the neighbor), they cross the switch fabric to the corresponding egress line card, where they might be further subject to specific functions before they are forwarded on the medium.

C. Faults that evade detection

Given the router architecture described above, we argue that the current fault detection mechanisms employed by today's routers, *i.e.*, using the OSPF Hello messages, *do not* capture all data-plane failure modes. For example, consider the path of a typical incoming OSPF Hello message through the router. The message enters the ingress line card, it's egress line card (which is the card containing the route processor) is looked up in the forwarding table, and is put in the queue corresponding to the route processor and finally is switched to that route processor. Similarly, an outgoing OSPF Hello message generated by the router, passes through the switch fabric to the corresponding egress line card, where it is transmitted to the neighbor over the link.

Clearly, with the combination of incoming and outgoing Hello messages, all the links are properly checked for connectivity. However, there are $O(p^2)$ combinations within the switch fabric, where p is the number of ports, out of which only $2 \times p$ paths are verified by these messages, p from all ingress ports to the route processor and another p from the route processor to all the egress ports.

While this example only shows that some of the crossbar switch fabric paths not checked by the OSPF Hello messages, there could be several other forwarding paths not verified by these Hello messages, such as those that involve specific function applied to specific types of traffic. We argue that while the exact nature of the problems might differ based on the particular router architecture, we believe that these probes fundamentally lack the ability to capture all the forwarding paths within routers, even at the IP layer alone and not considering other forwarding planes such as MPLS.

D. End-to-end probing

While the faults mentioned above evade detection using the OSPF Hello messages, still, they can be easily identified using end-to-end probes, which ISPs fortunately use anyway to measure SLA violations and in general, health of the network. Special measurement servers connected to provider edge routers inject probes periodically to other such measurement servers. When a sufficient number of probe packets are not acknowledged by the receiver, the measurement servers detect a connectivity loss and generate alarms for an operator to look at. Thus, while some set of paths may not be verified by the Hello messages, still, the end-to-end probes can detect and appropriately alarm the operator.

Upon alarms, the operator is faced with the problem of locating root cause, before any repair actions can be per-

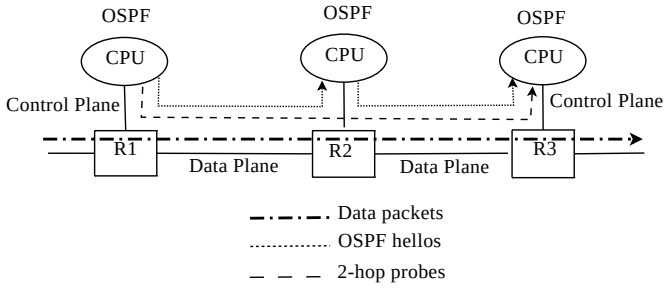


Fig. 2. 2-hop probes issued from router R1 to R3.

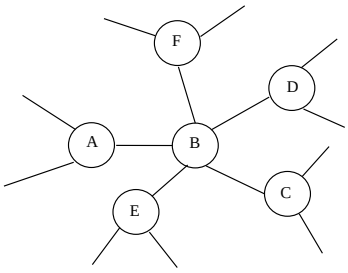


Fig. 3. Example

formed. One way to determine the root cause is to feed these connectivity losses into a localization engine (*e.g.,* [8]), that can then spatially correlate probe data according to the underlying topology to identify a small set of likely locations of the failure. These locations can then be examined manually by the operator for the failure. Thus, these problems can be easily taken care of.

We argue, however, that there are two main problems associated with these end-to-end probes. First, this feedback loop that is implemented in the management plane is quite slow, with the result that disruption can seriously affect traffic for extended periods of time. Second, end-to-end probes scale as $O(n^2)$, which can translate to a lot of traffic. To control the rate at which traffic is injected, the probes are only issued at low frequencies (*e.g.,* 1 per minute [8]), which subsequently increases the detection time. In general, therefore, to overcome these limitations, we need mechanisms that can be closely coupled with routing protocols while covering all the data plane forwarding paths within routers.

III. Our approach

In this section, we describe a simple mechanism that allows routers to verify all types of forwarding paths within routers. The basic idea is to allow routers to inject two-hop probes in addition to the traditional one-hop probes, *i.e.,* OSPF Hellos. These two-hop probes are forwarded through the adjacent router just as a regular data packet would, and thus, they would be able to verify connectivity across all forwarding paths within the adjacent router. We examine the mechanisms required to incorporate this basic idea into the existing routing protocols. While we use OSPF as a canonical example protocol in the following discussion, we note that our mechanisms can be used to verify connectivity of any forwarding path within routers by injecting probes at the appropriate layer (*e.g.,* MPLS) and that belong to the appropriate class (*e.g.,* Diffserv).

A. Basic Idea

Our goal is to verify the connectivity of all the forwarding paths within a router. One way to achieve this goal is to allow each ingress line card to send keep-alive messages to every other line card within the router. While this is indeed possible, this would require hardware modifications[1] to the line cards to generate keep-alive messages to other line cards within the router.

Alternatively, we propose the following simple extension to existing OSPF Hello mechanism to achieve our goal. Each router generates a new set of probes in the control plane, similar to OSPF probes, except between a router and all its 2-hop neighbors as shown in Figure 2. Each router can easily identify its two hop neighbors in link-state protocols (such as OSPF), as each router has typically the state of all the links in the network. These 2-hop probes would essentially pass through the adjacent router's data plane and would terminate at the 2-hop neighbor as shown in Figure 2.

As 2-hop probes are originated by the router CPU, there is no need for any additional hardware primitives to accomplish this, and thus can be easily implemented in today's routers. Note, however, that these 2-hop probes are *not* meant to be a replacement to the normal OSPF Hello messages and are instead, in fact, complementary to them. The normal OSPF Hello messages are still required even in the presence of the 2-hop probes, because, OSPF still operates at the granularity of a link. In case of a link failure, OSPF still needs to recompute routes that avoid the failure, just as today. For this to happen, two adjacent OSPF instances still need to communicate amongst themselves. The 2-hop probes only help detect forwarding-plane problems that are not completely captured by the current OSPF Hellos, which involve those that traverse the entire data plane from ingress to egress just as a normal data packet.

Similar to how routers already generate link-state advertisements (LSAs) during link failures (observed either using Hello messages or through lower layer alarms), routers also generate path state advertisements (PSAs) whenever these probes are not received by a router from a 2-hop neighbor within a certain amount of time. By flooding these PSAs to other routers, the router which is involved in the particular 2-hop probe will stop forwarding any packets towards the router where the failure occurred.

In Figure 3, we show an example, where router A sends 2-hop probes to its 2-hop neighbors, routers C, D, E and F

[1]In some cases, there is an additional CPU per line card, in which case, software modifications should be enough. However, these CPUs tend to be of lesser power than that of the general purpose route processor, and thus may still face practical deployment issues.

via router B. Suppose the forwarding path between A and C, *i.e.*, $A-B-C$, has failed, this would be detected by the router C which in turn propagates a corresponding PSA to indicate this failure. The router A notices the problem and avoids the router B altogether for any of its shortest paths. Note that while potentially the path $A-B-D$ is active, still, the router A conservatively decides not to use the router B for forwarding. This is because, it would require a way to selectively reroute the packets that go only via $B-C$ while allowing $B-D$ to continue as usual. While selective rerouting is not impossible, it could be difficult as forwarding tables today only output the next hop and not the next 2-hop. A different router, say E, however, can still use the paths through router B, as a PSA corresponding to $A-B-D$ does not automatically mean that $E-B-D$ is inactive.

One problem with PSAs is that any link failure can also cause potentially several of the 2-hop probes to be lost, thus generating a flurry of PSAs. One way to get rid of this problem is to make use of the fact that the entire network need not be aware about the PSAs, unlike LSAs which need to be flooded throughout the network. Typically, the sender of a given 2-hop probe as well as the router which has the particular problem need to be aware of this, which can be easily incorporated by allowing the receiver to just uni-cast the loss of the 2-hop probe to the upstream routers by the router that detects the failure. Thus, while many PSAs that depend on LSA are potentially effected, these messages will only have local scope. We later quantify the number of PSAs that need to be generated for specific ISP topologies in Section IV.

B. Other implementation details

The 2-hop probes need to be restricted to travel only two hops from the source, which can easily be ensured by setting the IP TTL value to 2 for these probes. Interface state machines can be similar to that of the Hellos, except that the router maintains 2-hop neighbors instead of the immediate neighbors. Neighbor data structures such as HelloInterval periodic timers and RouterDeadInterval can also be set accordingly for the 2-hop neighbors. As we show later in Section IV, the number of 2-hop probes are greater than the number of Hello probes. Thus, we can reduce the resulting overhead of these 2-hop probes on the router CPU by increasing HelloInterval timers. Thus, while there are large number of these 2-hop probes, they are launched less frequently as compared to the regular OSPF hello messages, so that the messaging overhead is manageable.

One issue we have skirted so far, concerns the verification of data plane paths within border routers. The border routers could be between two OSPF areas within the same AS or they could also belong to two different ISPs. Unfortunately, verifying all the data paths within border routers using our mechanism requires support from external routers, which might be difficult. Of course, the border routers can still generate 2-hop probes to its neighbors within the AS. The only issue is that no other router will be able to generate 2-hop probes to verify data plane forwarding within the border router. The other issue we have not discussed at length is incremental deployment. Unlike OSPF or other routing protocols, we can deploy the 2-hop probes incrementally. Even if all the routers are not compliant with the 2-hop probes, we can still get marginal benefit only along the 2-hop paths where such problems are continuously probed.

C. Comparison

The main advantage of the 2-hop probes in contrast with end-to-end probes is that, these 2-hop probes are issued in the network much closer to the place where the failure actually happens, and thus, they allow routers to take corresponding recovery actions based on this feedback.

We summarize our discussion on fault detection using different probe-based schemes in Table I for the different types of failures—link failures and path failures (*i.e.*, those only detected by the 2-hop probes). Clearly, end-to-end probes can detect both these faults, but localization is imprecise due to the fact that the inference problem using end-to-end probes is inherently under-constrained [3], [5]. Thus, it cannot easily be integrated into the control plane. On the other hand, OSPF Hello probes are able to detect link-level failures precisely but data plane faults go undetected which are detected using the 2-hop probes.

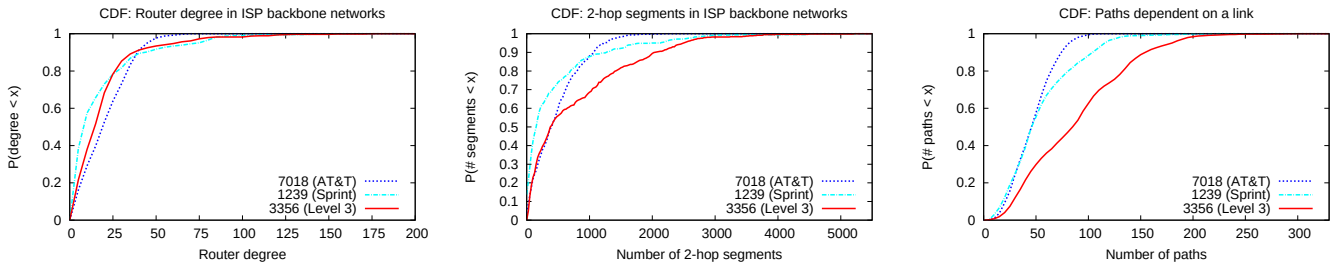
We also show the overall complexity of these various mechanisms. Probe traffic overheads for end-to-end probing is O(n^2), whereas that for OSPF Hello probes is O(m), where m represents the number of network links. 2-hop probes have a total probe traffic complexity of O($n \cdot d^2$), where *d* represents the average degree of a router and *n* being the number of backbone routers. We quantify these overheads associated with OSPF and 2-hop probing schemes in real ISP backbone networks in the next section.

IV. Evaluation

In this section, we evaluate the overhead associated with using OSPF Hello probe and 2-hop probe based approaches for fault detection. We use Rocketfuel topologies and maps for ISP backbone networks [11].

The Rocketfuel data provides the degree d_i of each router *i* and the list of its immediate neighbors *N(i)*. We note that d_i is always greater than or equal to the cardinality of *N(i)*, due to the existence of multiple links between two routers or due to aliasing (multiple IP interfaces of a router). As Rocketfuel maps do not provide us the exact number of links between a router *i* and its immediate neighbor *j*, we approximate it as $d_i/|N(i)|$. We can of course choose other ways to approximate this as well, but the overall results are not going to be that different. Thus, the total number of 2-hop segments originating from a router *i* can be then computed as $\Sigma_{j \in N(i)}\{(d_i/|N(i)|) \cdot d_j\}$.

Another important metric of interest is the number of path state advertisements or PSAs, which need to be generated for a single link failure. Thus, when a single link failure occurs, the number of 2-hop probes that would be affected because of this failure can be calculated as follows. Let d_i and d_j be the degrees of two routers i and j. If the link (i, j) fails, then,



(a) (b) (c)

Fig. 4. Distribution in ISP backbone networks.

	Types of fault		Features		
Scheme	Link Connectivity	Path Connectivity	Detection type	Reaction Time	Probe overhead
End-to-end	Imprecise	Imprecise	External	High	O(n^2)
OSPF Hello	Precise	Undetected	Internal	Low	O(m)
2-hop	Precise	Precise	Internal	Low	O($n \cdot d^2$)

TABLE I
FAULT DETECTION SCHEMES: A COMPARISON.

the 2-hop probes originating from all other neighbors of i to j will be affected. Similarly, all the 2-hop probes originating from i to all other neighbors of j will also be affected. Thus the total number of PSAs that need to be generated is $d_i + d_j$.

Figure 4(a) shows the distribution of the degree of the routers for three different ISPs. The median degree of a router, which corresponds to the number of Hello messages generated by the router, for all three ISPs varies between 8-20. Figure 4(b) shows the distribution of 2-hop segments for routers in three ISP backbone networks. As we observe in Figure 4(b), the median number of 2-hop probes per router is 150-400. If we assume that these probes are sent once every 20 seconds, probing rate is roughly 7-20 pps. As each of this probes is very small in size (only a few kilobytes), the bandwidth overhead will be almost negligible for routers operating at Gbps speeds today. The number of interrupts triggered per second is also fairly small, and we have the flexibility to fine-tune it by configuring the probing rate according to the needs of the router and the network.

Figure 4(c) shows the number of forwarding paths dependent on a link between adjacent router pairs in the backbone networks. As discussed earlier, this quantifies the number of path state advertisements needed to be generated for one link failure. As we also observe in Figure 4(c), the median number of such advertisements varies between 46-85 for the three ISPs. Though these advertisements are relatively large in size as compared to the 2-hop probes, their associated overhead is still very low when compared to their value.

V. CONCLUSIONS

In this paper, we showed that today's routing protocols such as OSPF that employ simple Hello packets to detect faults, do not completely cover all the forwarding paths within routers. As routers become more complicated with newer types of architectures and protocols, it is imperative that routing protocols be retrofitted with mechanisms that can deterministically monitor all forwarding path failures. While end-to-end probes in the management plane offer a line of defense, it is still important to incorporate these mechanisms into existing routing protocols, as the external feedback loop can be quite slow. Our 2-hop probes provides a simple and effective mechanism to incorporate such data-plane checks into routing protocols with small amount of overhead.

VI. ACKNOWLEDGEMENTS

We thank Cisco systems for their support. We would like to thank anonymous reviewers for their comments.

REFERENCES

[1] S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Dec. 1998.
[2] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. RFC 1195, Dec. 1990.
[3] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *ACM SIGCOMM*, 2004.
[4] B. Davie and Y. Rekhter. *MPLS: technology and applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.
[5] N. Duffied. Simple network performance tomography. In *USENIX/ACM Internet Measurement Conference*, 2003.
[6] S. Iyer and N. W. McKeown. Analysis of the parallel packet switch architecture. *IEEE/ACM Trans. Netw.*, 11(2):314–324, 2003.
[7] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown. Scaling internet routers using optics. In *ACM SIGCOMM*, pages 189–200, 2003.
[8] R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Detection and localization of network black holes. In *IEEE Infocom*, May 2007.
[9] N. McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM Trans. Netw.*, 7(2):188–201, 1999.
[10] J. Moy. Ospf version 2. RFC 2328, IETF, Apr. 1998.
[11] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP Topologies with Rocketfuel. In *Proceedings of the ACM SIGCOMM Conference*, Aug. 2002.
[12] G. Varghese. *Network Algorithmics*. Morgan-Kaufmann, 2005.