Methods for localizing network link failures

Akbari Indra Basuki and Fernando Kuipers
Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
I.B.Akbari@tudelft.nl, F.A.Kuipers@tudelft.nl

***Abstract*—Most of today's networks use Link Aggregated Group (LAG) to increase the bandwidth. A single physical link failure under LAG will reduce network capacity significantly. Unfortunately, there are no standard protocols to localize link failure, especially to localize link failure at the physical layer. Some protocols, such as BFD and LACP, only work locally and do not distribute a notification about the link failure. SNMP, OSPF, and IS-IS only work at the data-link layer. In this paper, we will evaluate and compare the methods to localize link failure in the presence of LAG. The methods can be classified into two mechanisms: correlated-paths probing and per-link monitoring. We first implement per-link monitoring methods using Software Defined Networking (SDN) and compare them with existing correlated-paths probing based on their scalability, latency, accuracy, flexibility, and applicability. Our results show that there is no overall winner, but gives insight into which approach is best suited for which objectives.**

***Index Terms*—Localizing link failure, per-path monitoring, per-link monitoring, SDN.**

I. Introduction

Today's networks consist of high-speed lines having tens to hundreds of Gb/s capacity per link. One single link failure could therefore significantly reduce network capacity. In order to timely react to link failures or reduced performance that is indicative of imminent link failure, it is important to be able to quickly and accurately pinpoint the location of a link failure. Packet rerouting, by forwarding the packet through an alternative path, is the first action to be taken to minimize packet loss. The next step is to re-evaluate the routing inside the network, to make sure that the reduced network capacity is exploited in the best way possible. And, finally, efforts should be undertaken to repair the failed network parts as soon as possible. For all steps, it is crucial to quickly know what has failed where in the network.

Localizing Link Failure (LLF) is far from trivial, since there are no standard protocols that facilitate this. SNMP [18], OSPF [14], and IS-IS [20], for example, are only able to detect link failures at the data-link-layer level. Localizing link failure at the physical layer is important, considering that today's networks often use Link Aggregation Group (LAG, Fig. 1) to combine several physical links into a single data-link-layer link. This allows to reach Tb/s link capacity by combining several 100 Gb/s fiber optic links. Unfortunately, today's network protocols are unable to localize link failures occurring inside aggregated links (unless such aggregated links would fail completely). Other protocols, such as BFD [8] and LACP [2], only handle link failures locally and do not report failures to a central controller or set of monitoring nodes. More advanced link-failure monitoring approaches are therefore needed.

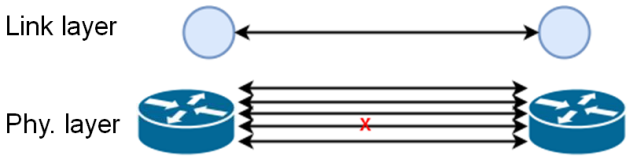

Fig. 1. A link failure within an aggregated link.

Two kinds of monitoring methods can be considered to localize a link failure: (1) correlated-paths probing and (2) per-link monitoring.

Existing solutions to localize link failure usually adopt correlated-paths monitoring by sending probe packets over explicit paths, via existing network protocols like RSVP-TE and MPLS.

Typically, there are four steps in correlated-paths-based link failure localization: (1) selecting a (or some) monitoring point(s), (2) computing monitoring paths through the network and back to the monitoring point(s), (3) sending the probe packets over the monitoring paths, and (4) correlating the results on which probe packets arrived over which paths to determine the location of link failures. While the approach may work, it suffers from several drawbacks:

- Scalability problems: Typical path computations, such as for P-cycles [6], are hard to solve in large networks.
- Lack of multi-link failure support: Most of the correlated-paths-based LLF solutions only work for single link failures, because localizing multiple link failures requires intensive computation in determining monitoring paths [6].
- Inflexible to network changes: The monitoring paths and look-up-tables for the correlation process have to be recomputed in case the network topology or monitoring nodes change.

In per-link monitoring, one probe is assigned to test one specific link in the network. Per-link-based LLF does not have the above-mentioned issues and may be a good alternative, because (a) the probe-packet size is typically very small and would likely consume only a small portion of the total network capacity and (b) it supports multi-link failure detection. The drawback of using per-link based LLF methods is that a fairly high amount of probe packets may be needed. To mitigate this problem somewhat, per-link-based LLF methods could deploy

multiple monitoring nodes, so that the localization process could be split into k non-overlapping groups.

There are two different types of per-link-based LLF, namely proactive and reactive. Proactive localization uses a heartbeat protocol, such as BFD, to test the link periodically. In a reactive mechanism, the data-plane is waiting for a signal from a monitoring node to test the link and to report it back to the monitoring node.

In this paper, we present and compare several methods to localize link failure at the physical layer. In particular, our main contributions are as follows:

- We propose several per-link-based LLF methods, since no such methods currently exist for LAG-based networks.
- We compare, via simulations, the per-link methods to correlated-paths methods, based on five Key Performance Indicators (KPI's).

The following KPI's will be evaluated: (1) scalability, (2) detection time, (3) localization accuracy, (4) flexibility, and (5) applicability in the network. Scalability relates to algorithmic complexity, the state that needs to be maintained by each network device, communication overhead, and the extent to which the approach is centralized. Detection time reflects the time it takes to localize a link failure. There are two aspects in localization accuracy: (a) the ability to localize link failure at the physical layer and (b) the ability to localize multiple links that fail at the same time. Flexibility refers to how quickly the method can adjust to network changes. Network applicability pertains to the ease/difficulty of implementing the method.

The remainder of this paper is structured as follows: Section II proposes several SDN-based link-failure localization approaches. Section III provides our performance evaluation. Section IV concludes the paper.

II. System Design

In this section, we describe the design of four different kinds of per-link-based LLF methods. We use Software-Defined Networking (SDN), since SCMon [7] showed that an SDN-based solution can greatly improve LLF performance. We implement both proactive and reactive per-link-based LLF methods for two types of data-planes (see Table I): (1) a dumb data-plane that only follows the instructions of the controller and (2) a smart/stateful data-plane that has the ability to keep state about the network.

TABLE I
Per-link-based Localizing Link Failure methods.

	Proactive	Reactive
Dumb data-plane	Combining OpenFlow with BFD	Per-link Segment Routing (OpenFlow-based)
Stateful data-plane	Combining BFD with Stateful flooding	Stateful flooding with adjacent link testing

A. OpenFlow + BFD

BFD is used to test link connectivity. There are therefore two important events that BFD can detect: an existing link goes down, or a link comes up. However, this information is not passed on to a monitoring node and only used by the nodes connected to the link. Therefore, in OF+BFD, OpenFlow is used to install flow rules that forward information on a change in link state (it goes down or comes up) to one or more monitoring nodes. This could be done via the shortest paths.

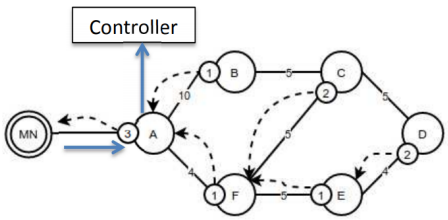

Fig. 2. Any common host can become a monitoring node by sending a request to the controller.

A variant of OF+BFD, which is the one we have implemented, is that all BFD packets are sent to the controller. The controller can then infer from unreceived packets that a link has gone down. This approach creates more probe traffic, but it avoids to keep state at the nodes on whether a link is up or down.

B. Per-link segment routing

In a reactive mechanism, a probe packet is sent to a specified node to test a link and the result is reported back to a monitoring node. However, installing a specific flow rule for every probe packet is not efficient. Segment routing could be used to mitigate this problem by abstracting every node and link in the network using a segment ID [19], [4]. The forwarding path computation and flow rule updates for the segment routing implementation are handled by the controller.

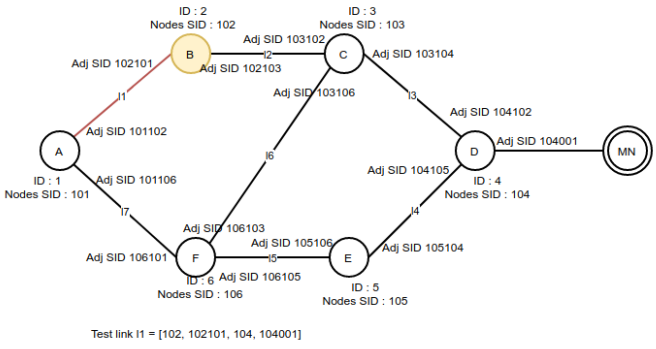

Fig. 3. An example segment ID sequence to test link L1.

The segment routing implementation must implement node Segment IDs (node SIDs) and adjacency Segment IDs (Adj SIDs) to support LLF. Every host, in principle, could test every link in the network by inserting a sequence of 4 SIDs into the header of a probe packet. The sequence of 4 SIDs is as follow: [Node SID of the target node ($N0$), Adj SID of the link (L_Adj) to be probed, Node SID of the node (Na) adjacent

to the monitoring point, and Adj SID of adjacent link between the monitoring point and the adjacent node (MN_Adj)]. $N0$ could be any of the two end-points of the link to be probed. We need Na and (MN_Adj) in order to be able to send the probe back to the monitoring point. Fig 3 provides an example of this configuration.

Per-link segment routing LLF does not require SDN and could rely on IS-IS or OSPF instead. To compute the monitoring paths, a translation algorithm is used to convert topological link information into a correct sequence of SIDs and set subtraction is used to localize multi-link failures (Algorithm 1).

The drawback of this method is that it relies on external processes to route the probe packets. If these routes are affected by a link failure, it could take some time to converge to a new routing state, which could delay the process of localizing the failure.

Algorithm 1 PerLinkSegmentRouting

procedure INIT($G, Na, inPort$)
  $PP \leftarrow null$
  for each Node N in G **do**
    $N.segment\_routing \leftarrow enable$
  for each Link L in G **do**
    $pp.id = L$
    $N0\_sid = Node\_SID[L[0]]$
    $N1\_sid = Node\_SID[L[1]]$
    $L\_Adj = Concat(N0, N1)$
    $Na\_sid = Node\_SID[Na]$
    $MN\_Adj = Concat(Na, inPort)$
    $pp.label = [N0\_sid, L\_Adj, Na\_sid, MN\_Adj]$
    $PP \leftarrow pp$
  Return PP
procedure LOCALIZE(G, PP)
  $AL \leftarrow null$
  $RL \leftarrow null$
  $LF \leftarrow null$
  for each Link L in G **do**
    $AL \leftarrow L$
  for each probe_packet pp in PP **do**
    $send(pp)$
  while not timeout **do**
    if receive(pp) then $RL \leftarrow pp.id$
  $LF = AL - RL$
  Return LF

C. Stateful flooding

A stateful data-plane is defined as a data-plane equipped with a state machine to work smarter than non-stateful data-planes. The state machine could be implemented in various ways, e.g. by keeping a certain value as the state for a specific flow table [9], [10], or by being able to install or delete a new flow rule based on an incoming packet (instead of per controller instruction), such as in OpenVSwitch [16].

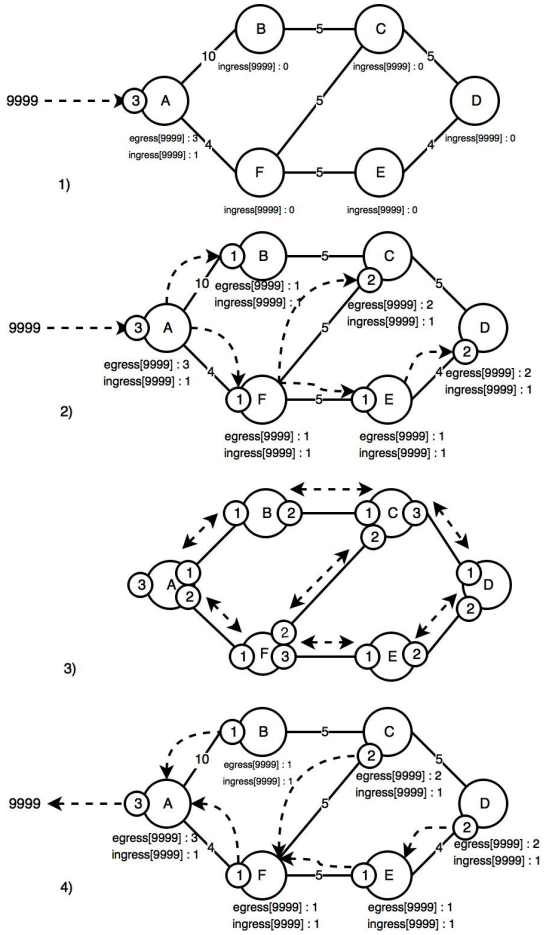

Fig. 4. Localization steps using stateful flooding: 1) initial phase, 2) discovery phase, 3) Testing phase, 4) Reporting phase.

Stateful flooding works by flooding the network using a single probe packet to install a new state into the data-plane of every node. The state is used to construct reporting paths to monitoring nodes without intervention of the controller. For every unique probe packet that is received by the data-plane, there are two states that must be installed: a visited state in the ingress table and a shortest-path state in the egress table. The visited state prevents the probe packet from looping by dropping the probe packet at the second encounter. The shortest-path state keeps the interface number that received the probe packet for the first time. This number will later be used as a destination port to forward the probe packet back to the monitoring node (Fig. 4).

In a proactive mechanism, the probe packet acts as a beacon to install forwarding paths from every node's data-plane to the monitoring node. The incoming BFD packet will use the installed forwarding path to report link status to the monitoring node.

In a reactive mechanism, the probe packet not only acts as a beacon to install the reporting paths, but also acts as a trigger for every node to start the link testing. The data-plane of the receiving node will make a copy of the probe packet, by using

Algorithm 2 Stateful Flooding

```
1: procedure INIT STATE(N)
2:     Ingress state ← null
3:     Egress state ← null
4: procedure ADVERTISE(N, Probe, inPort)
5:     if State for Probe.id is not set then
6:         Ingress state← id
7:         Egress state← inPort
8:         Reflood Probe()
9:         Send link_test()
10: procedure SEND LINK_TEST(N, Probe)
11:     for each Interface If in N, If ≠ inPort do
12:         Probe.label ← N.id + Probe.id
13:         Probe.mode ← link_test
14:         send Probe to If
15: procedure RECEIVE LINK_TEST(N, Probe, inPort)
16:     Probe.label ← N.id + inPort
17:     Probe.mode ← report
18:     send Probe to inPort
19: procedure RECEIVE REPORT(N, Probe)
20:     forward Probe to Egress table
21:     send probe to Interface[Egress State]
22: procedure LOCALIZE(G, Probe)
23:     AL, RL, LF ← null
24:     for each Link L in G do
25:         AL ← L
26:     send a probe packet
27:     while not timeout do
28:         if receive Probe, Probe.mode = report then
29:             RL ← Probe.label
30:     LF = AL − RL example
31:     Return LF
32: procedure INIT BFD(G)
33:     for each Node N in G do
34:         for each Interface If in N do
35:             If.bfd ← enable
36: procedure RECEIVE BFD(N)
37:     Probe.label ← N.id + inPort
38:     forward Probe to Egress table
39:     send probe to Interface[Egress State]
```

a group bucket, and will modify the probe packet mode to a $link\_test$ packet (MPLS TC = 1). It then floods the testing packet to all adjacent nodes and waits for their responses. The adjacent nodes must embed their ID and incoming port into the testing packet before sending it back to the sender. After returning, the testing packet mode will be changed into a report mode (MPLS TC = 2) and then be forwarded to the egress table. By using the state value stored in the egress table, the probe packet can be forwarded from one node to another until it arrives back at the monitoring node (Algorithm 2).

The advantage of stateful flooding is that it quickly reacts to link failures by avoiding them during state installation. The obvious disadvantage is that it needs to maintain state, the amount of which is limited by the available memory. To solve this problem, our implementation limits the number of requesters to a specified number m. This means that at any time there are at most m monitoring nodes. Each monitoring node has its ID in the range of 10000 to $10000+m$, stored as an MPLS label. For $m = 10$, the maximum number of state that must be kept in the data-plane of every node is $2{*}m = 20$.

III. EVALUATION

Our evaluation focuses on the following KPI's: Scalability, Detection time, Accuracy, Flexibility, and Applicability. We conduct both a theoretical evaluation as well as an empirical evaluation. The empirical evaluation is based on experiments using Mininet [15], the Ryu SDN controller, OpenVswitch [16], and ofsoftswitch13 [9]. The topologies used are Surfnet [3] and those of the Rocketfuel project [21].

The LLF methods used in our evaluation are limited to those that are able to localize link failure in aggregated links (LAG). We consider three correlated-paths LLF methods: TABM [12], SCMon [7], and a novel method proposed by us, called SPTree, which is introduced in the Appendix. We will compare them to our proposed per-link LLF methods: Combining OpenFlow with BFD (OF+BFD), Per-Link Segment Routing (PLSR), Stateful Flooding (SF), and combining Stateful Flooding with BFD (SF+BFD).

A. Method scalability

The scalability of LLF methods is determined by four major factors: algorithmic complexity, amount of state that must be kept, the number of probe packets needed, and the measure of centralization (or single point of failure). Table II provides a high-level overview.

TABLE II
SCALABILITY COMPARISON OF LLF METHODS.

LLF methods	Algorithm [Complexity]	States	Probe packets	SPoF
TABM	Set cover problem [NP Complete]	High	Low	No
SCMon	All-pairs shortest Path [$O(EV + V^2 logV)$]	None	Low, depends on k	No
SPTree	Single source shortest path [$O(E + V logV)$]	None	$E - V + 1$	No
OF+BFD	Single source shortest path [$O(E + V logV)$]	None	$2 * E/m$	Yes
PLSR	Simple translation [$O(V)$]	None	$2 * E/m$	Yes /No
SF	No computation	$2 * m$	$2 * E/m$	No
SF+BFD	No computation	$2 * m$	$2 * E/m$	No

TABM is the least scalable LLF method considering the high computation demands to compute monitoring paths and high amount of state required (equal to the number of monitoring paths). SCMon scales best, since it has a simple algorithm (all-pairs shortest-path), no state to be maintained, and a small number of probe packets. SPTree has similar scalability as SCMon, except for the amount of probe packets used.

All of our per-link methods require a fairly high number of probe packets. However, since they support multiple monitoring nodes, this can be mitigated.

In stateful flooding, the amount of state is related to the number of monitoring nodes and not to the network size.

A Single Point of Failure (SPoF) may thwart the scalability and result in complete failure of the localization process. When the central node or link to this node is lost or broken, the entire localization process will not work. OF+BFD and PLSR use a centralized controller [4], [19] and thus have a SPoF, unless multiple controllers are used. PLSR. which is based on IGP routing, does not have a SPoF.

B. Localization time

Localization time consists of two factors: localization latency and localization frequency. Localization latency determines how fast the localization process finishes. Localization frequency means how often the localization can be repeated to yield a stable and correct result.

In single link failure localization, localization latency and frequency have the same value. The next round of localization can only start if a specific time window (localization frequency) has passed.

In multi-link failure localization, localization latency and frequency may differ, considering that the localization process depends on an external path computation that may not update the paths quickly after a link failure has occurred. Since the correlated-paths LLF methods are unable to perform multi-link failure localization for LAG networks, we consider only per-link LLF when evaluating localization frequency.

There are three factors that determine localization latency: (1) correlation methods, (2) length of monitoring paths, and (3) links delays. There are two possible correlation methods to localize link failure, namely probabilistic and deterministic methods. Probabilistic methods are used in absence of the data-plane ability to steer packets through a certain interface. The selected interface is chosen using a hash-table to randomly select the interface. As a result, we need to gather some adequate amount of data and do some probabilistic analysis to conclude if the link is down. In deterministic methods, we send the probe packets through a certain interface and hence we can exactly determine the location of a link failure by analyzing the path taken by the probe packet. TABM [12] uses a probabilistic method and the other methods are deterministic.

In this paper, considering that the compared methods use different kinds of software switch and have different mechanisms in forwarding the probe packet, we need to characterize the inner latency (IL) of the data-plane in forwarding a probe packet.

Fig. 5 show the inner latency of the methods for some linear topologies using zero link delay configuration.

Internal latency of LLF methods highly depends on the type of data-plane. The OpenVswitch data-plane has a very fast flow rule processing. LLF methods that use the OpenVSwitch data-plane (OF+BFD, SF+BFD, SPTree, SCMon) have a nearly flat latency. The internal latency is not affected by the length of hops. On the other hand, the internal latency of PLSR and SF methods, which are implemented using ofsoftswitch13 [9], display a linear increase according to the length of the path. In PLSR, stateful flow rules are always fixed and are only reinstalled when there is a topology change. In SF, the stateful flow rules are installed at each probing session. As the result, SF has a longer internal latency compared to PLSR.

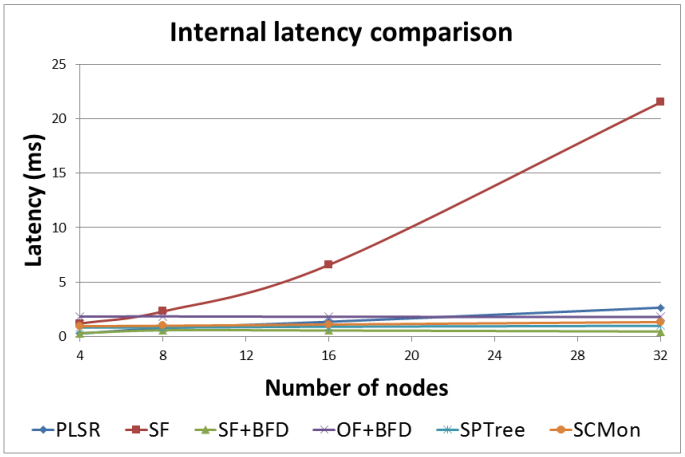

Fig. 5. Internal latency comparison of LLF methods.

Since the measurements were done with mininet, our results should be seen as an indication of the latency. Experiments on a real testbed would be needed to gain a better insight.

Table III shows the comparison of localization frequency for each per-link LLF method.

TABLE III
LOCALIZATION FREQUENCY OF MULTI-LINK FAILURE IN PER-LINK LLF METHODS.

LLF Methods	Localization frequency
OF+BFD	5 seconds (OpenvSwitch [16]
PLSR	5 seconds (Controller based [16]), 40 seconds (IGP routing convergence time [20], [14])
SF	$<$ 50 ms
SF+BFD	$<$ 50 ms

C. Flexibility

In correlated-paths localization, computing monitoring paths, such as P-cycles, may be hard, which means that adjusting to topology changes would be hard as well. Per-link localization methods rely on simple shortest paths computations. And in stateful flooding no computation is needed, since it is automatically constructed. Table IV shows the flexibility parameter for each per-link LLF method.

In OF+BFD, network changes have to be handled by the controller. In segment routing methods, the shortest paths are calculated by the IGP routers. However IGP routing needs a convergence time t to adapt to changes. Stateful flooding finds shortest paths by keeping state and flooding. Hence, monitoring nodes do not require any information about topology

TABLE IV
FLEXIBILITY PARAMETER IN PER-LINK LOCALIZATION METHODS.

Parameter	BFD+OF	Per-link SR	SF(+BFD)
Change of network topology	shortest path computation	shortest path computation (IGP Router)	No computation
Information update of topology change	manually to controller	manually to monitoring nodes	No need
Change monitoring node location	shortest path computation	No computation	@ 1 seconds
Addition of a new monitoring node	shortest path computation	No computation	No computation

changes. However, the location of monitoring nodes cannot be moved before state timeout, which is at least 1 second.

D. Applicability

The applicability aspect indicates to what extent the LLF methods can be implemented in various types of networks. We consider four different types:

- All-optical network
- Ethernet network
- SDN-enabled network
- SDN-enabled network with programmable data-plane

An all-optical network has limited packet switching flexibility. Packet forwarding usually is achieved by constructing a tunnel using RSVP-TE or MPLS LDP protocols. This property limits the way to localize link failures. If a probe packet must be sent through a certain path, a tunnel has to be constructed for that path. An Ethernet network has better packet switching and forwarding flexibility than an all-optical network, such as IGP routing. Even though SDN can be considered a promising way of networking, most existing network devices do not support SDN protocols such as OpenFlow. Segment routing based LLF methods can be implemented by extending IGP routing protocols, therefore it has better applicability. It is easier and cheaper to upgrade the firmware of devices rather than to replace them with new ones that support OpenFlow. Stateful flooding requires a very specific data-plane to keep or memorize the states, which limits its usage in today's networks.

IV. CONCLUSION

A summary of our KPI analysis is presented in Table V.

TABLE V
KPI ANALYSIS SUMMARY OF LLF METHODS

Parameter	TABM	SCMon	SPTree	OFBFD	PLSR	SF(+BFD)
Scalability	++	++	+	++	+++	+++
Latency	+	+++	++++	++++	++++	++++
Accuracy	++	++	++	++++	++++	++++
Flexibility			+	++	+++	++++
Applicability	++++	+++	+++	++	+++	+

Per-link LLF methods have the upper hand in terms of localization latency. They have better latency than most other LLF methods, except SPTree, which uses a shortest path tree. By using a shortest paths tree, the longest time to forward packets is the longest propagation time from monitoring node to the furthest nodes in the network, which is equal to the diameter of the network, which is likely shorter than the cycles used in correlated-paths methods.

Per-link LLF methods have better accuracy, considering that they are able to localize multi-link failures at the physical layer (LAG implementation). Even though most of them are unable to localize multiple link failures in a short time, they are still useful to localize permanent link failures, such as fiber cuts or broken interfaces. Stateful flooding is the only method that is able to localize multi-link failures in a very short time.

Correlated-paths LLF methods have better applicability, since they can be implemented in any type of network. Per-link LLF methods require specific network protocols, such as segment routing, OpenFlow, or stateful data processing.

Our KPI analysis shows that there is no overall winner. Each method has its own strengths and weaknesses.

REFERENCES

[1] Core-netwerk, regioringen en stadsringen. https://www.surf.nl/themas/netwerk/core-netwerk-regioringen-en-stadsringen/index.html. Accessed: 2017-02-08.

[2] Ieee 802.3ad link bundling.

[3] The internet topology zoo. http://www.topology-zoo.org/. Accessed: 2017-02-22.

[4] Onf's spring-open project.

[5] Open-nfp.

[6] Rachna Asthana, Yatindra Nath Singh, and Wayne D Grover. p-cycles: An overview. *IEEE communications surveys & tutorials*, 12(1), 2010.

[7] François Aubry, David Lebrun, Stefano Vissicchio, Minh Thanh Khong, Yves Deville, and Olivier Bonaventure. Scmon: Leveraging segment routing to improve network monitoring. In *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pages 1–9. IEEE, 2016.

[8] Manav Bhatia, Mach Chen, Marc Binderberger, Sami Boutros, and Jeffrey Haas. Bidirectional forwarding detection (bfd) on link aggregation group (lag) interfaces. 2014.

[9] Giuseppe Bianchi, Marco Bonola, Antonio Capone, and Carmelo Cascone. Openstate: programming platform-independent stateful openflow applications inside the switch. *ACM SIGCOMM Computer Communication Review*, 44(2):44–51, 2014.

[10] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

[11] P4 Language Consortium et al. Behavioral model (bmv2).

[12] Nicolas Guilbaud and R Cartlidge. Localizing packet loss in a large and complex network. In *Proc. NANOG*, volume 57, 2013.

[13] Timothy Adam Hoff. Extending open vswitch to facilitate creation of stateful sdn applications.

[14] Kireeti Kompella and Yakov Rekhter. Ospf extensions in support of generalized multi-protocol label switching (gmpls). 2005.

[15] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM, 2010.

[16] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan J. Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Jonathan Stringer, Pravin Shelar, Keith Amidon, and Martín Casado. The design and implementation of open vswitch. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, pages 117–130, Berkeley, CA, USA, 2015. USENIX Association.

[17] Stefano Previdi, Martin Horneffer, Stephane Litkowski, Clarence Filsfils, Bruno Decraene, and Rob Shakir. Source packet routing in networking (spring) problem statement and requirements. 2016.

[18] JD Case M Fedor ML Schoffstall and C Davin. Rfc 1157: Simple network management protocol (snmp). *IETF, April*, 1990.
[19] A Sgambelluri, F Paolucci, A Giorgetti, F Cugini, and P Castoldi. Sdn and pce implementations for segment routing. In *Networks and Optical Communications-(NOC), 2015 20th European Conference on*, pages 1–4. IEEE, 2015.
[20] Henk Smit and Tony Li. Is-is extensions for traffic engineering. 2008.
[21] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.
[22] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.
[23] Han Wang, Robert Soulé, Huynh Tu Dang, Ki Suh Lee, Vishal Shrivastav, Nate Foster, and Hakim Weatherspoon. P4fpga: A rapid prototyping framework for p4. In *Proceedings of the Symposium on SDN Research*, pages 122–135. ACM, 2017.
[24] Bin Wu, Pin-Han Ho, and Kwan L Yeung. Monitoring trail: On fast link failure localization in all-optical wdm mesh networks. *Journal of Lightwave Technology*, 27(18):4175–4185, 2009.

APPENDIX

A. Shortest-path tree (SPTree) LLF

SPTree consists of three phases: path mapping, probing, and correlation. Path mapping will only be computed once the topology is changing. The probing phase is when all probe packets are sent concurrently or almost at the same time. The correlation phase is when all received probe packets are analyzed or correlated to determine the exact location of a link failure. At every monitoring interval, probe packets are sent through their respective precomputed paths. In the next interval, the correlation process for the previous interval is executed and probe packets for the current interval are sent out. Therefore, the link failure localization response is equal to the probing interval. A more detailed explanation of each phase will be described in the following.

1) Path mapping
In this phase, path mapping is computed using single-source Dijkstra's shortest paths algorithm. The purpose is to find the main paths and shortcut links. The main paths are the shortest paths to all nodes. Shortcut links are the remaining unchosen links (not present in the shortest paths tree). In Fig. 6, there are 3 main paths: (3, 2, 1), (3, 6, 7), and (3, 4, 5) and two shortcut links: (1-6) and (6-5).

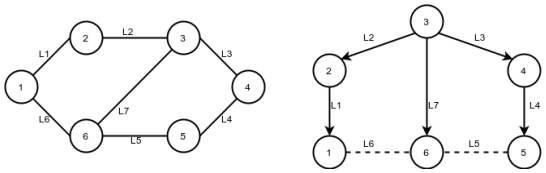

Fig. 6. Simple network and its shortest paths representation.

The shortest path tree is only able to determine link failure if all nodes in the network are connected to at least three adjacent nodes. In case a node has one adjacent link (node 7) or two adjacent links (node 1, 2, 4, and 5), an additional probe packet is needed to exactly pinpoint the failure at those links.

To create stack information for each probe packet, we construct the SID stack as follows. If a link, L1, connects two nodes (A and B), so that $L1 = (A, B)$, the structure of the segment stack is: [node SID of A, adjacency SID of (A,B), node SID of MP]. For every recorded additional node (AN), create an additional probe packet and construct its stack by combining node SID and monitoring point SID: [node SID of AN, node SID of MP].

Algorithm 3 Path mapping Algorithm(G)

1: All links $(AE) \leftarrow G.links()$
2: Node degree $(ND) \leftarrow ComputeDegree(G.nodes())$
3: Monitoring Point $(MP) \leftarrow ND[0]$
4: Additional Nodes $(AN) \leftarrow \forall n \in ND$, $degree == 1$ or $degree == 2$
5: Shortest path $(SP) \leftarrow DijkstraShortestPath(G)$
6: Probe packet $(PP) \leftarrow null$
7: $\forall link \in AE, link \notin SP$:
   a) stack $\leftarrow null$
   b) stack $\leftarrow Nodes\_SID(link[0])$
   c) stack $\leftarrow Adj\_SID(link[0], link[1])$
   d) stack $\leftarrow Nodes\_SID(MP)$
   e) $PP \leftarrow stack$
8: $\forall nodes \in G.nodes()$:
   a) $PP \leftarrow$ [Nodes_SID of node, Nodes_SID of MP]
9: return PP

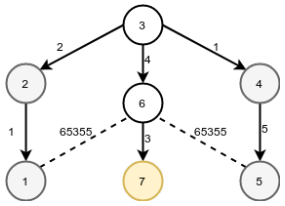

Fig. 7. Simple IGP weighting using main path and adjacency links dichotomy.

In our solution, link cost in the main path can be set to any number, as long as the total sum of the costs for every branch to the leaf node is lower than the cost of adjacent links. For example, in Fig. 7, we set link cost for every adjacent link with maximum number 65355. By using this configuration, the main path, or the shortest path between one node to others, is always consistent.

2) Probe forwarding
Probe packet forwarding is executed at a very short interval (50 ms) to enable dynamic link failure monitoring. At every iteration, each probe packet which is already preset with its respective stack information, is sent by the monitoring point. This monitoring point broadcasts probe packets depending on the top most label stack. The other nodes will forward the packet based on the segment ID stored in the MPLS stack. The

last destination of all probe packets is the monitoring point. We use a single monitoring point to simplify the correlation procedure.

3) Correlation Procedure
At the initialization phase, we compute the set of links used by every probe packet and then use them to assign every link a unique combination of probe packets. Every probe packet which uses link A as its probing path will be used by link A as its set of probe packets. Localization is performed by comparing the unreceived probe packets with the set of probe packets owned by every link. If all probe packets in the set are unreceived, the link can be assumed to be failed.

Algorithm 4 Initialization Algorithm (PP)

1: All links $(AE) \leftarrow G.links()$
2: $\forall probe \in PP$:
   a) "set of links" $\leftarrow null$
   b) $\forall label \in probe\_stacks$:
      i) "set of links" $\leftarrow SetofLinkAtSegment(label)$
   c) matrix value at row[probe] $\leftarrow$ "set of links"
3: $\forall links \in AE$:
   a) set of probe packets of link $\leftarrow SumOfColumn(link)$

Algorithm 5 Localization Algorithm(unreceived PP)

1: uPP $\leftarrow unreceived\_PP$
2: List of link failure (LLF) $\leftarrow null$
3: $\forall link \in AE$:
   a) if all probe packets $\in uPP$:
      i) LLF $\leftarrow link$
4: return LLF