Effective Route Scheme of Multicast Probing to Locate High-loss Links in OpenFlow Networks

doi: https://doi.org/10.2197/ipsjjip.29.115

Journal of Information Processing Vol.29 115–123 (Feb. 2021)

[DOI: 10.2197/ipsjjip.29.115]

Regular Paper

Effective Route Scheme of Multicast Probing to Locate High-loss Links in OpenFlow Networks

Nguyen Minh Tri[1,a)] Masahiro Shibata[1,b)] Masato Tsuru[1,c)]

Received: May 14, 2020, Accepted: November 5, 2020

Abstract: The prevalence of cloud computing and contents delivery networking has led to demand for OpenFlow-based centrally-managed networks with dynamic and flexible traffic engineering. Maintaining a high level of network service quality requires detecting and locating high-loss links. Therefore, in this paper, a measurement framework is proposed to promptly locate all high-loss links with a minimized load on both data-plane and control-plane incurred by the measurement, which assumes only standard OpenFlow functions. It combines an active measurement by probing multicast packets along a designed route and a passive measurement by collecting flow-stats of the probing flow at selected switch ports in an appropriate sequential order to access switches. In particular, by designing the measurement route based on the backbone-and-branch tree (BBT) route scheme, the measurement accuracy and the measurement overhead (the number of accesses to switches until locating all high-loss links) can be balanced. The numerical simulation demonstrates the effectiveness of our proposal.

Keywords: active measurement, multicast measurement, flow statistics, OpenFlow network

1. Introduction

Software Defined Networking (SDN) technology in general and OpenFlow in particular have attracted attention over the years [1] and are becoming widespread as a replacement solution for traditional network not only in data centers but also in enterprise networks and wide area networks or so-called SD-WAN. By decoupling the control plane and data plane, SDN becomes more flexible and simpler to design, configure, operate, and monitor. The deployment of services and features in the network is executed by programming on controllers. On a data plane, switches forward packets based on rules from controllers. In particular, the ongoing prevalence of cloud computing and contents delivery networking requires flexible traffic engineering on a network connecting globally-distributed datacenters, which is often centrally managed by OpenFlow [2], [3].

Monitoring is an essential task in network management and operations. Operators need to know the network status/performance information in a real-time manner to make decisions about trouble-shooting, dynamic routing, load balancing, Service Level Agreement management and so on. In general, there are two kinds of measurement approaches: passive and active. The passive approach is used to monitor link traffic state by collecting the statistical information (e.g., flow-stats) from switches (through the OpenFlow monitoring messages or SNMP) or by monitoring the OpenFlow-standard operating messages themselves. There is a trade-off between the measurement accuracy and the measurement overhead in the control network. Polling at a high frequency can increase the timeliness and accuracy but also imposes a greater load on switches and the control network. Studies have been made on how to reduce this load. In OpenTM [4], each flow is measured by a periodical query to one switch. However, the switch decision can affect accuracy. In FlowSense [5], by only using FlowRemoved and PacketIn messages of OpenFlow standard, the network utilization status can be calculated with no additional cost, but it cannot trace quickly changed links. In PayLess [6], a dynamic algorithm to balance the request frequency and the accuracy was introduced. Similarly, in OpenNetMon [7], an adaptive polling rate to access edge switches was designed to reduce network and switch CPU overhead while optimizing the accuracy in throughput, delay and packet loss measurements.

On the other hand, the active approach sends and receives probe packets to measure the packet loss, delay, the round-trip-time (RTT), and so on. With the development of the edge-cloud computing for emerging IoT technologies, the development of reliable networks among a large number of heterogeneous sites is required over wider geographic locations. In such networks, a "link" between two nodes is not always physical but sometimes virtual (e.g., tunneling) one that traverses inaccessible intermediate switches prohibited from being monitored by passive measurements. Therefore an active measurement by probing packets is essential for monitoring entire network information. Furthermore, to realize a highly flexible and dynamic traffic engineering in OpenFlow networks, the status/performance of all links should always be monitored and performance-deteriorated links should be located in a real-time manner. However, probing at a high sending rate for a long duration can impose a greater load on the switches and the data network. Therefore, studies have been made on how to reduce the load while still retaining reliability and precision. Adding an active measurement function by which

[1] Kyushu Institute of Technology, Iizuka, Fukuoka 820–8502, Japan
[a)] tri.nguyen-minh414@mail.kyutech.jp
[b)] shibata@cse.kyutech.ac.jp
[c)] tsuru@cse.kyutech.ac.jp

probe packets will be sent and received on some or all switches that are globally controlled by a manager is a straightforward approach and was implemented in some dedicated switches such as Cisco's Service Assurance Agent (SAA)/Internetwork Performance Monitor (IPM). However this approach requires a special function beyond OpenFlow standard on each switches. Authors in Ref. [8] proposed an infrastructure to monitor RTT that focuses on reducing the flow entries and the number of probe packets. In Ref. [9], a measurement scheme that can cover all links in both directions while minimizing flow entries on switches is presented. Focusing on datacenter networks, in Ref. [10], a controller designs the probing routes and the multiple probing servers send probe packets along the designed routes, which are bounced by some switches back to the servers. A processor then collects the resulting data by accessing the servers. Such arbitrary (effective) probing routes are realized by IP-in-IP technique. In Ref. [11], a real-time failure location is achieved by devising the design of effective probe matrix and using source routing of probe packets. However, since they all use unicast probing in an end-to-end (among servers or beacons) manner, some links may suffer from many overlapped probing paths traversing them.

Boolean network-tomographic approaches have been studied that only monitor performance-level correlations among measurement paths to deduce the location of bad internal links. Seminal works such as Ref. [12] have attracted much attention and been followed by a number of studies because of their practicality (e.g., Ref. [13]). However, network-tomographic approaches always work with a considerable inference errors. The impact of the capability of routing of probe packets has also been studied in localizing failed nodes based on Boolean network tomography [14].

In this paper, based on and motivated by those existing work, we present a network-assisted measurement framework for OpenFlow networks to monitor all links in both directions to promptly and efficiently locate high-loss (performance-deteriorated) links that aims to minimize the load on both the data-plane and control-plane incurred by measurement. In contrast to existing works, our framework combines an active measurement by probing multicast packets from a measurement host to appropriate switch ports and a passive measurement by collecting flow-stats of the probing flow at selected switch ports in an appropriate sequential order to access switches that is determined dynamically. The former reduces unnecessary loads on the data plane incurred by probe packets and avoids the concentration at a link near the measurement host, while the latter reduces unnecessary loads on the control plane incurred by switch accesses until all high-loss links are located. Note that a simple Boolean network-tomographic inference of highly lossy range (a sequence of links) is used to dynamically determine a sequential access order to collect flow-stats. In contrast to typical network-tomographic approaches, a tomographic approach in the proposed framework is not used for finally identifying of the lossy links; it is used as a hint for narrowing the search space and for optimizing the search order.

This paper is an enhanced version of our preliminary conference paper [15] but includes a completely new route scheme that significantly outperforms the route scheme based on the shortest path tree in Ref. [15], in terms of fewer number of accesses to switches to reduce the unnecessary load on the control-plane. In another extension by our group [16], a link weight is introduced in the shortest path tree-based route scheme by using the results of past measurements to find which links are likely to be lossy, in order to place loss-prone links near the ends (leaves) of a route tree. However, since it was also based on the shortest path tree, it should be improved by our new scheme in this paper.

The basic system model is presented in the next section. The probe packet route algorithm is presented in Section 3. How to dynamically determine a sequential access order to collect flow-stats to efficiently locate/identify the high-loss links is shown in Section 4. Section 5 discusses the design of the route scheme. Section 6 provides the experimental results through simulation. Concluding remarks are given in the last section.

2. System Overview

The proposed method is based on the framework that we previously proposed to monitor and locate high-loss links using multicast probing on OpenFlow networks [15]. It assumes the standard functions of OpenFlow-based networks comprising OpenFlow controller (OFC) and OpenFlow switches (OFS) to leverage per-flow flexible routing/multicasting and per-flow monitoring of network statistics in a centralized manner; and is implemented on the OFC. The process begins when the measurement host (MH) sends a measurement request to the OFC, as in **Fig. 1**. Then, the OFC obtains network topology, calculates probe packet routes, and installs them into OFSs. **Figure 2** shows an example of a route scheme that is described later in Section 3.1. The switch port connected to the MH is the root port; the leaf port is a switch port which discards probe packets. A route of the probe packets (the measurement flow) from the root port to a leaf port is referred to as a terminal path. The number of links from the root port to the leaf port is the path length.

Following the above events, a series of probe packets is launched by a single MH. Here, each probe packet (or a copy) passes through each link once and only once (in each direction of a full-duplex link separately) and is discarded at a leaf port on the last OFS along the terminal path. The number of probe packets

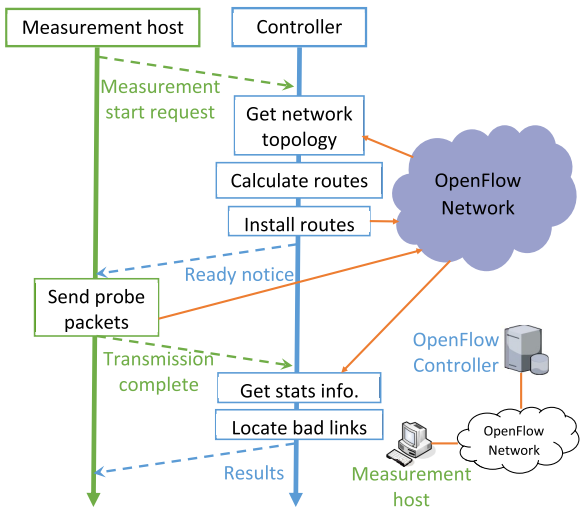

Fig. 1 Measurement process to locate bad links [15].

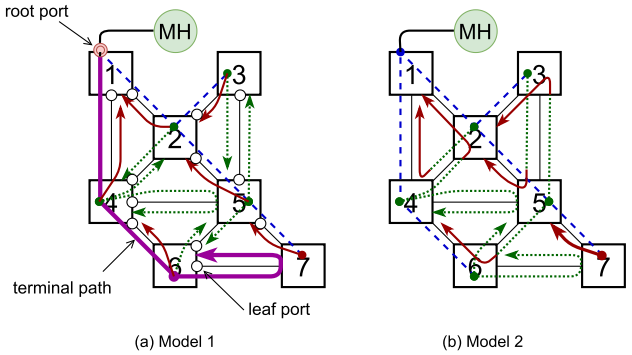

Fig. 2 Route scheme in Ref. [15].

arriving at an individual input port on each OFS is recorded and collected by the OFC if required. Then, the packet loss rate on a link (or a sequence of links) between two switch ports is calculated by finding the difference between the numbers of arriving probe packets at those two ports.

Two important features that are strongly correlated are (i) flexible design of multicast measurement on a route tree with an MH location (the root of the tree) to cover all links in the active probing and (ii) dynamic optimization of the sequential access order to switches for collecting the flow-stats of the measurement flow at some switch ports that are passively monitored to locate high-loss links. In Ref. [15], a shorted-path tree based multicast route scheme is used where each probe packet traverses each link only once to minimize unnecessary loads on the data-plane in OFS incurred by probe packets. This route scheme can avoid concentration of probe packets at links near the MH, especially in large networks. Different possible multicast measurement routes (including a single unicursal unicast measurement route over all links as an extreme case) can have the same benefit relative to data-plane load, however measurement robustness and accuracy are strongly affected by the measurement route. For example, when a large number of probe packets are lost on a given link, all succeeding downstream links on that terminal path might not be monitored accurately due to a reduced number of probe packets passing through those links. Thus, a very long single unicursal route should not be used.

To reduce the time required and an unnecessary load on the control plane in the OFC and OFSs for locating all high-loss links, a sequential access order for the necessary flow-stats on the required OFSs is determined dynamically. However, minimizing the length of each terminal path creates many short terminal paths especially in large-scale networks and results in more accesses to OFSs from the OFC. These negative results occur because the OFC needs to at least collect the flow-stats at the root port and at every leaf ports to get the loss rates of all terminal paths.

Therefore, in this paper we propose a new route scheme that can achieve a balance between measurement accuracy and measurement overhead (the number of accesses). Our proposal can keep the terminal paths short enough to avoid errors in loss rate estimation, while keeping the number of terminal paths small enough to significantly reduce the necessary number of accesses to switches.

3. Route Scheme Design

3.1 Two Baseline Route Schemes

The measurement route traversed by probe packets has a strong impact on search performance or namely how it can locate all high-loss links with a small load. To minimize the load on the data-plane, probe packets have to travel every link only once. Please note that each link is assumed to be in the full-duplex mode and the probe packets should traverse in each direction on every link. To fulfill this requirement, a simplest measurement route is a unicast trail. Since a link is full-duplex, each undirected link between OFSs is considered as two oppositely directed edges and OFSs are considered as vertices. In such a directed graph, from any vertex, the Eulerian cycle (circuit) algorithm can find a trail that passes every edge exactly once because each vertex has an even number of edges (it has an even degree). We call this trail a unicursal path. Its length is double the number of links. In this case, the route scheme has only one terminal path with a maximum length.

However, as discussed above, a long terminal path is not a good choice. In Ref. [15], we proposed a route scheme with minimum length paths, called Model 1 as Fig. 2 (a), and a variant from it, Model 2, with longer paths, as Fig. 2 (b). Both models use Dijkstra's shortest path algorithm first to build a tree on an undirected graph from the MH, which corresponds to a downstream part of the entire multicast route as shown by the blue dashed lines. Then, in Model 1, terminal paths are constructed to keep each of them as short as possible by adding unused links and reverse links to different terminal paths, as shown by the green dotted lines and the red lines, respectively. On the other hand, Model 2 tries to reduce the number of terminal paths by combining unused links and reverse links to increase the length of each terminal path. For example, at the Node 3 of Fig. 2, in Model 1, there are three different paths: two unused paths in the green dotted lines and one reverse path in the red line. Whereas, in Model 2, the reverse path from Node 3 to Node 2 is combined with the unused path from Node 5, resulting in two different paths at Node 3 in total. In general, with short terminal paths, both models operate robustly and accurately. However, Model 2 with less number of terminal paths has better performance compared with Model 1.

Both unicursal and our previously proposed route schemes have shortcomings because of their extremes. The unicursal route with maximum path length requires a smaller number of accesses to OFSs to locate all high-loss links but needs a larger number of probe packets to operate accurately. Our previous shortest path tree-based route scheme with shorter-length paths requires a smaller number of probe packets to maintain accuracy under loss conditions but needs a larger number of accesses to OFSs to locate all high-loss links. In both extremes, the number of and the lengths of terminal paths cannot be intentionally controlled. In the next subsection, we propose a new route scheme that can balance the number of terminal paths and their lengths and which is evaluated through simulation later in Section 6.

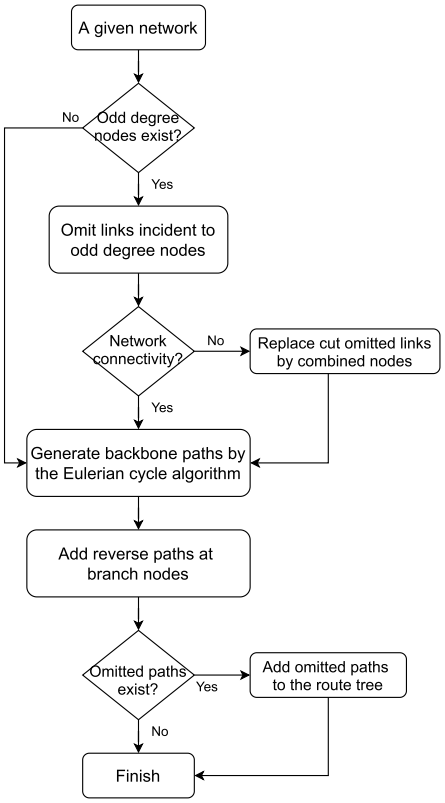

Fig. 3 Proposed route tree design flowchart.

3.2 The Proposed Backbone-and-branch Tree Route Scheme (BBT)

The newly proposed route scheme is called the backbone-and-branch tree route scheme (BBT) and its flowchart is illustrated in **Fig. 3**. In BBT scheme, the Eulerian cycle algorithm is applied to the original undirected graph (network). Since a Eulerian cycle only exists in the graph such that every vertex has an even degree, first we need to process all odd degree vertices (nodes) by temporarily omitting some links in general. Then we generate the backbone paths by using the Eulerian cycle algorithm to efficiently cover links as many as possible and also by considering how to avoid excessively long terminal paths. After generating the backbones, the reverse direction segments of route (toward the measurement node) on the backbone paths are added to the route tree at some branch nodes. Note that a segment is a sequence of adjacent directed links to form a part of route. Finally, we integrate paths of omitted links (called omitted paths) into the route tree.

3.2.1 Omit Links Incident to Odd Degree Nodes

The general idea of processing odd degree nodes is to omit the links between their couples. In this way, the degrees of these nodes become zero or even. Note that the number of odd degree nodes is always even. Omitting links is based on the following criteria with the priority of: (1) try to keep the network connected and (2) minimize the number of omitted links.

If the network obtained by omitting links incident to nodes with odd degrees is still connected, a simple backbone path can be generated. Otherwise, if several omitted links become a cut set in the network (the network becomes disconnected if this path is removed), the backbone path is fragmented and there is an impact on the performance. In this case, the cut omitted path is replaced by a combined node in the network, see **Fig. 4** (a). The omitted path between Node A and Node B is combined into node C, the dotted node in Fig. 4 (b). From that point, the network is still connected, and all nodes have an even degree. Figure 4 (c) is the backbone path from the combined network. However, the final backbone path in Fig. 4 (d) is fragmented. So, maintaining the network connection is the highest priority when omitting odd degree nodes.

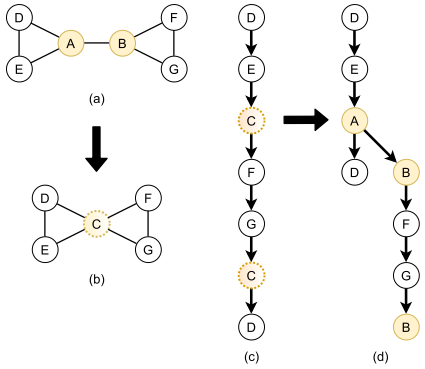

Fig. 4 Example of a cut path.

To minimize the omitted links, it takes many computations for searching all combinations of each odd degree couple. Therefore, we propose an approach to reduce the computation and the time by omitting in the following sequence. Firstly, we omit links of the unique path between a 1-degree node and another odd degree node. Secondly, the link between two neighbor odd degree nodes is omitted. Finally, from the remaining odd degree nodes, we calculate the shortest paths from each node to the others and then select the optimal combination. The selected combination is the case that the number of the omitted links is minimized and the network is still connected. Each series of omitted links is called an omitted group. Note that if two omitted groups have a common node, they are combined into one omitted group.

3.2.2 Generate the Route Tree

After the omitting step, each node in the network has an even degree. We can generate the backbone paths of probe packets. Then, for each segment of the backbone path, its reverse path is added as the branches of the route tree. A reverse path or reverse segment is the reverse direction segment of the route toward the measurement node on the backbone path. This process is illustrated in **Fig. 5**. Figure 5 (c) is an artificial Eulerian cycle-based network topology without odd degree nodes. The numbers on links in the figure indicate the link distance between two nodes. In other words, there are other unnamed nodes/links between those two nodes that are counted in the length of paths and segments, while they are not explicitly illustrated in the figure. From node A as the measurement node, the node that connects to the measurement host, the backbone path of the route tree is generated based on the Eulerian cycle algorithm. Here, there are two options: 1 backbone path case (BBT T1) or 2 backbone paths case (BBT T2). BBT T1 has a full Eulerian cycle as the backbone path, the bold line as in Fig. 5 (a). The path starts from node A, travels through nodes B, C, D, F, E, B, D, and ends at node A (also the start node). BBT T2 has two backbone paths that are

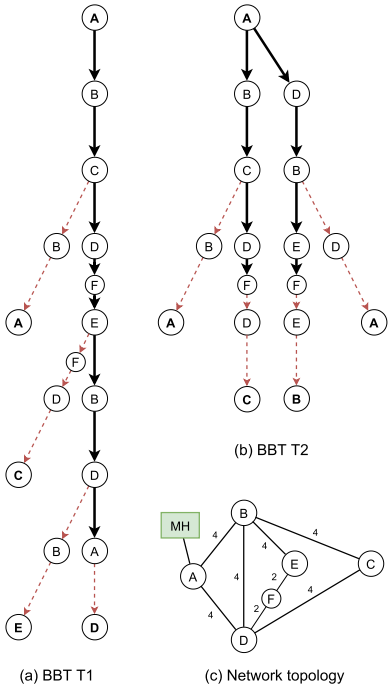

Fig. 5 Illustration of the backbone paths and the reverse branches.

two reversed halves of the Eulerian cycle, see Fig. 5 (b). These are the A-B-C-D-F path and the A-D-B-E-F path, which meet at node F in the middle of the Eulerian cycle. This option can halve the length of the backbone path.

Note that, the number of backbone paths directly affects the number of and the lengths of terminal paths of measurement route, essentially related with the trade-off between the tolerance to heavy loss of measurement packets and the efficiency in lossy link detection by measurement packets. We should properly choose the number of backbone paths depending on the expected loss rates on links, that is the degree of lossiness of links. In the proposed scheme, the number of backbone paths is limited by the degree of measurement node. If the degree of measurement node is two, we can construct two backbone paths at maximum. This is the situation utilized as an example in this paper. If the degree of measurement node is more than two, although it was not quantitatively evaluated, we can construct more than two backbone paths. For example, in Fig. 5 (c), if the measurement node is Node B then three backbone paths can be constructed. These are the B-A-D, B-C-D, and B-D-F-E paths. However, it is desirable to more flexibly decide the number of backbone paths in order to control the above trade-off in response to the network topology and link conditions. A possible option to increase such flexibility is to consider multiple MHs in either a physical or a virtual manner. For example, an appropriate set of nodes can be selected as virtual measurement nodes from which a backbone path will start. In this setting, the probe packets should be forwarded from a real MH to each of virtual measurement nodes along some nodes/links. However, this remains a topic for future work.

To build reverse paths, first, we divide a backbone path into multiple backbone segments. The length of each backbone segment decides the number of backbone segments (it is also the

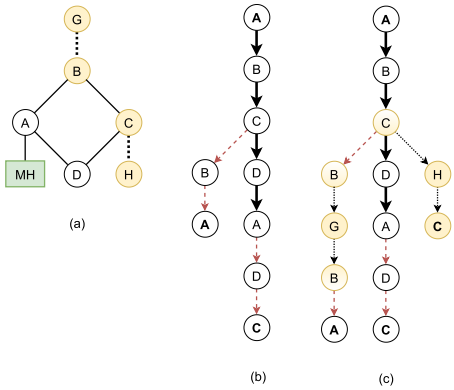

Fig. 6 Example of integrating omitted paths.

number of terminal paths). As we will discuss later in Section 5, it is desirable that the length of segment is similar across all segments. The segmentation should take this point into consideration. Then, at the end node of each segment called branch node, the reverse path is added, like dashed lines in Fig. 5. Each reverse path has the same length but opposite direction of its backbone segment. For example, in Fig. 5 (a), there are four segments: A-B-C, C-D-F-E, E-B-D, and D-A on the backbone. So, 4 reverse paths (segments: C-B-A, E-F-D-C, D-B-E, and A-D) are added to the route tree and generate 4 terminal paths.

After adding reverse paths, the omitted paths are integrated into the route tree. From the common node of the generated route tree and an omitted group, a unicursal path (a single route visits all links of the omitted group in both directions) is generated. This path is integrated into the reverse path or added to the backbone path as an individual branch path. For an example in **Fig. 6**, the route tree before integrating omitted paths is in Fig. 6 (b). In the omitted group B-G, the common node is Node B, in the middle of the reverse path. So, the omitted path B-G-B is integrated into the reverse path C-B-A and this reverse path becomes the integrated reverse path C-B-G-B-A, see the left branch in Fig. 6 (c). To prevent a path from becoming too long, the omitted path can be added to the backbone path as an individual branch path if the common node is the branch node, see the right branch path C-H-C in Fig. 6 (c). In the complete route scheme, all paths from branch nodes (including reverse paths, integrated reverse paths, and individual branch paths) are called branch paths or branch segments. If there is only one branch path from the final branch node (the end node of the final backbone segment), this branch path is included in the final backbone segment.

Note that the Eulerian cycle can start from any node. If a given measurement node is omitted in the "omit links incident to odd degree nodes" step, the nearest remaining node is considered as the new measurement node. A segment of this omitted path from the original MH to the new MH is integrated into the backbone path, and the rest is the branch path.

4. Sequence Access Order

After receiving the probing complete notification from the MH, the OFC begins the next process to locate high-loss links by querying and collecting flow statistics information from selected OFSs in an appropriate access order. Note that link is considered

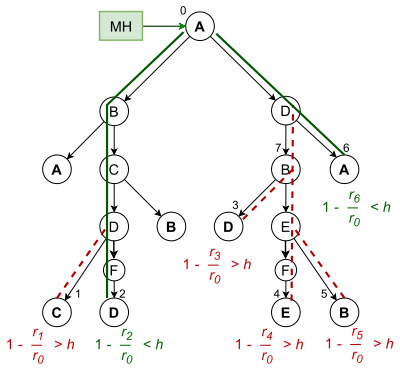

Fig. 7 Example of the access order.

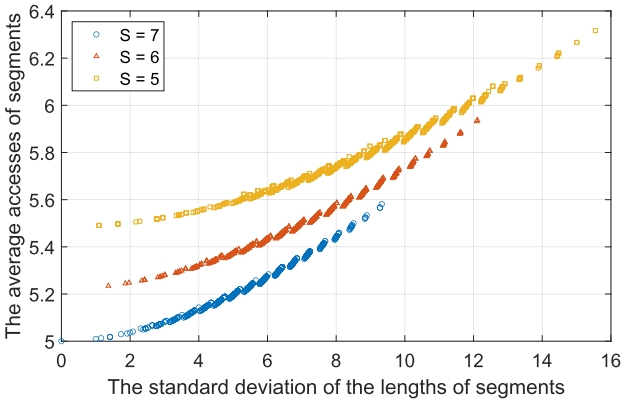

Fig. 8 Average accesses of segments.

as high-loss if and only if its loss rate exceeds threshold value h. Here, h is a design parameter that represents the target link quality to be maintained which depends on the target applications. The packet loss rate (PLR) of a range (a sequence of links) from ports i to j, $PLR = 1 - \frac{r_j}{r_i}$, where r_i and r_j are the numbers of probe packets arriving at switch ports i and j, respectively.

First, the OFC calculates the PLR of each terminal path with the information at the root port and leaf ports. If the PLR of a terminal path is less than h, this terminal path must not include a high-loss link. If the PLR of a terminal path exceeds h, this terminal path is likely to include one or more high-loss links. Then, by considering the correlation among terminal paths in terms of the degree of packet loss, we can narrow the search scope (the expected locations of high-loss links). If a terminal path is high-loss and there are no other high-loss terminal paths, the high-loss links are located within a range between the leaf port and the nearest branch port on the considered high-loss terminal path. The dashed line on the left part in **Fig. 7** shows an example of this case. The binary search algorithm is used to locate all loss links in a range.

If there are multiple terminal paths whose PLR values exceed threshold h, the port most commonly shared by those paths is queried first to collect the number of probe packets that have arrived, which produces separated subtrees, and the same procedure is performed recursively. An example of this case is illustrated in the right part of Fig. 7, the next requested port is the port 7 of node B. The actual packet loss rate of each high-loss link is measured exactly based on the difference between the numbers of arriving probe packets at the link's upper and lower ports.

5. Discussion

In this section, we will discuss issues when designing the route scheme that affect the measurement performance. These are the length of segments and related problems such as the difference in length of segments, impacts of omitted paths and cut paths on the route scheme.

Assuming that there is only a high-loss link in a given segment, the average number of required accesses of a segment is as follows. If the length s of the segment is 2^k, $k = 1, 2, 3, ...$

$$F_{seg} = 2 + k. \tag{1}$$

Otherwise,

$$F_{seg} \approx 2 + \log_2 s. \tag{2}$$

The OFC requests the first and the last ports of this segment. The binary search algorithm is then used to locate the high-loss link.

Let S be the number of segments. We number (as index) all the segments from 1 to S so that the index of the first common segment is 1 and the indexes of the branch segments are $S - B + 1, \ldots, S - 1, S$, respectively. B is also the number of terminal paths. Let s_i be the length of the i^{th} segment, and n be the number of links. If the single high-loss link is randomly located at any link with the same probability $1/n$, the average number of accesses to locate the high-loss link is

$$F_{T1} \approx 1 + B + \frac{s_1}{n}(1 + \log_2 s_1) + \sum_{i=2}^{S-B} \frac{s_i}{n}(2 + \log_2 s_i) + \sum_{i=S-B+1}^{S} \frac{s_i}{n}(1 + \log_2 s_i). \tag{3}$$

The number of accesses is the sum of accesses of one root port, B leaf ports, the average accesses of the segment which has the high-loss link. Since the OFC requested the root port and all leaf ports when searching on the first common segment and branch segments, only one port is needed to request.

Since each probe packet traverses each link only once, the total length of segments is the number of links. The length of each segment impacts on the number of accesses. **Figure 8** shows the relationship between the total of the average accesses of segments and the standard deviation of the lengths of segments. The number n of links is 56, and the numbers of segments are 5, 6, and 7. The optimal values happen at small the standard deviations. That means the length of each segment should preferably be similar among all segments. Note that omitted paths are added to branch segments, so minimizing omitted paths length keeps the difference between segments small. Additionally, because of using binary search to locate high-loss links, the segment length with the 2^k form is a better choice.

If the omitted path is a cut path, besides generating new branch path (causing more accesses), the difference in length among segments also becomes large and therefore the number of accesses becomes large. Thus, keeping the network connected in the backbone is important for providing a good performance.

On the other hand, the length of the segment also decides the number of terminal paths. The shorter segment length is, the more

Table 1 Number of terminal paths and the path length of route schemes on the ideal topology.

	Paths	Average	Min	Max
Unicursal	1	56	56	56
T1_seg8	4	26	16	32
T2_seg8	4	18	16	20
T1_seg4	7	20	8	32
T2_seg4	8	13	8	16

Paths: Number of terminal paths
Average: The average length of terminal paths.
Min: The minimum length. Max: The maximum length.

terminal paths are generated, see **Table 1**. In the table, T1_seg.x is the BBT T1 route scheme with the segment length is x, and T2_seg.x is similar to T1_seg.x. An acceptable (maximum) length of terminal paths is determined by the degree of lossiness of links on the network. In addition, if the length of a segment is long and/or the loss rates of lightly lossy links on the segment are not extremely low, the measured loss rate of the segment may exceed a threshold of high-loss link, which results in more number of accesses due to a decision error when narrowing high-loss ranges and finally locating high-loss links. Since the length of a segment is preferably similar among all segments, the length of a segment on the backbone path is a parameter of our route design. However, an appropriate length of each segment depends on the lossiness of the links, which may be predicted (estimated) based on the past information by constantly monitoring the network. However, this issue remains a topic for our future work. In general, for massive loss networks, the length of each segment should be short and vice versa. For the evaluation in Section 6, we examine route schemes in different loss environments.

6. Evaluation

To evaluate the performance of route schemes, we consider the number of required accesses of the OFC to OFSs. First, we use an artificial ideal topology as shown in Fig. 5c, to compare the performance of different route schemes and their segment length. In this topology, the link distance between two nodes is 4 which means there are 4 links and 3 hidden nodes between them, excluding the D-F link, and the E-F link is 2. We assume high-loss links have loss rates from 0.15 to 0.2 and the threshold is 0.1. Other links are considered in three cases: (1) No light-loss, (2) has small light-loss rates with the value from a range of $[0, 0.02]$, and (3) has moderate light-loss rate with the value from a range of $[0, 0.04]$. The number of probe packets is $100,000$. The samples are $100,000$. The information of terminal paths of route schemes is in Table 1.

Figure 9 shows that in small loss or no loss environments, the route scheme with less number of terminal paths has better performance. In the massive loss environment, the performance of route schemes is similar, see the third column group in Fig. 9. In this case, although there is no high-loss link, the accumulated loss rate of links in terminal paths can exceed the threshold, resulting in more accesses to locate actual high-loss links.

The results in **Fig. 10** show the relationship between the accuracy and the number of probe packets in the mass loss network with 2 high-loss links with a loss rate that is randomly selected from a range of $[0.5, 0.7]$, other links have moderate light-loss

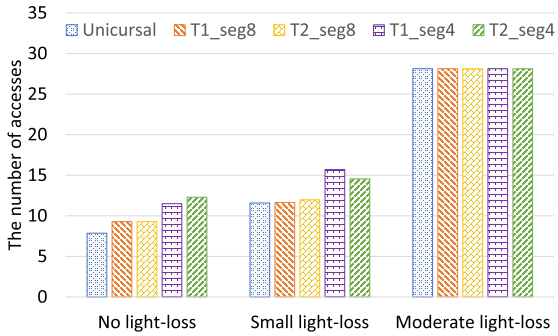

Fig. 9 Number of accesses to locate 1 high-loss link on the ideal topology.

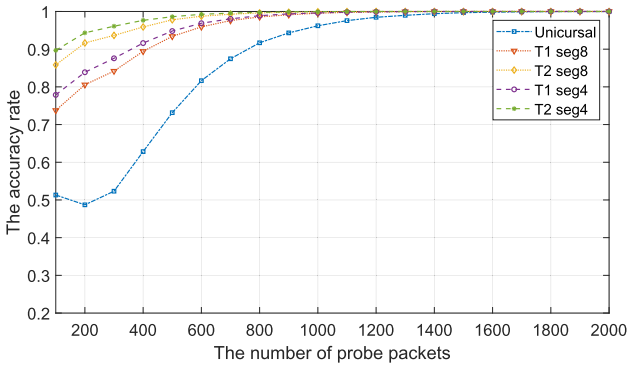

Fig. 10 Accuracy rate of the measurement in the moderate light-loss environment.

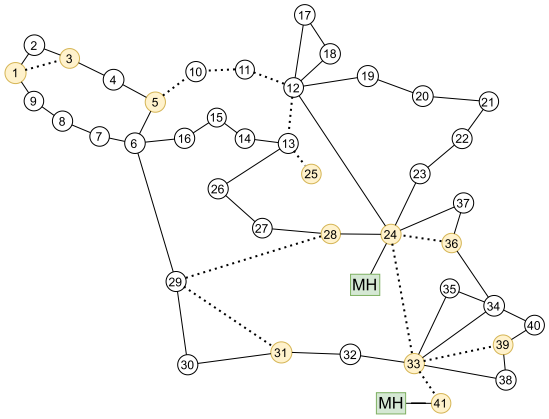

Fig. 11 Renater network topology [17].

rates. We see that the route scheme with longer path length needs more probe packets to operate accurately. In comparison with the unicursal route length, the longest path of our proposal is about 50% in the BBT T1 and 25% in the BBT T2. Besides, by reducing the segment length, we can generate more terminal paths with shorter lengths and increase the accuracy.

To evaluate the performance in the large-scale network, we consider a network topology based on the real network illustrated in **Fig. 11**. In this simulation, the number of high losses is changed from 1 to 4, with the loss rate is from 0.15 to 0.2. Other links have a random light loss value from a range of [0, 0.02]. The number of probe packets is 100,000. The threshold is 0.1. The BBT T2 is used with 2 backbone paths, and the segment length is 8. Dotted lines show omitted paths. Two situations for the MH location, at Node 24 and Node 41, are considered.

Figure 12 shows the number of required accesses on the Renater topology with different methods and MH locations. Our

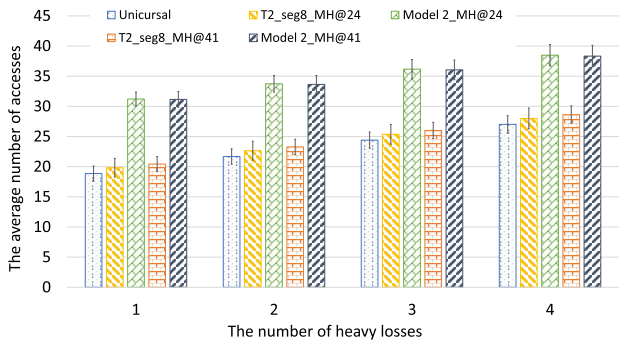

Fig. 12 Number of required accesses to locate high-loss links on the Renater topology in a small light-loss environment.

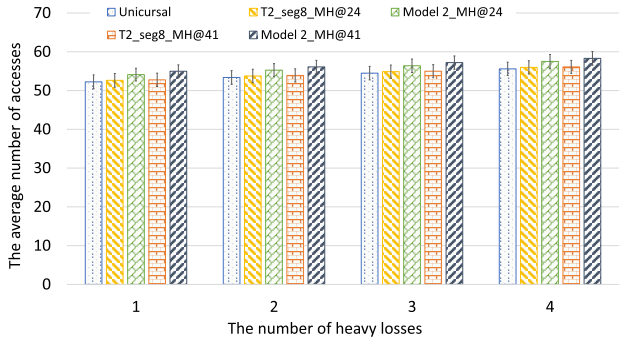

Fig. 13 Number of required accesses to locate high-loss links on the Renater topology in a moderate light-loss environment.

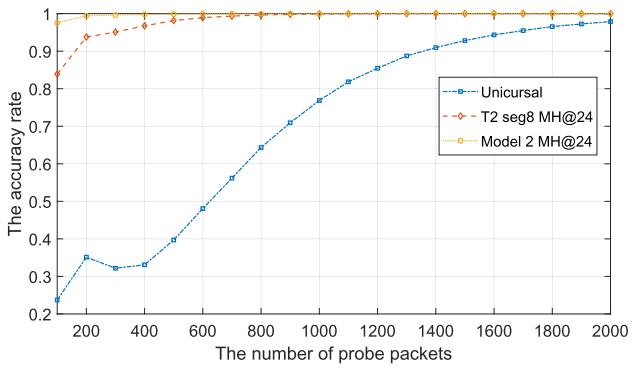

Fig. 14 Accuracy rate on the Renater topology in a moderate light-loss environment.

proposal has performance close to the unicursal route and improves significantly compared to the heuristic variant model (Model 2 in Ref. [15]) in the small light-loss environment. In the moderate light-loss environment, the number of accesses of our proposal is also smaller, see **Fig. 13**. In this environment, the number of probe packets affects the accuracy remarkably. **Figure 14** shows the relationship between the number of probe packets and the accuracy of the measurement. Our proposal also has an accuracy rate close to the previous method.

Note that, in an ideal topology without an odd degree node, our proposed route scheme is independent of the location of the MH. However, in a general network, the location of the MH has a certain influence especially at the omitted nodes. This shows in the performance of two cases where the MH connects to the omitted node, Node 41, and the remaining node, Node 24. The case with the MH at the omitted node is worse. This is because the omitted path included the MH will be fragment into 2 parts. One part is integrated into the backbone path and the other is the branch path. So the route scheme in this case has more segments, and the standard deviation of the lengths of segments is also larger than the case with the MH at remaining Node 24, see **Table 2**.

Table 2 Information of terminal paths and segments of route schemes on the Renater topology.

	Paths	Average	Min	Max	S	stdev
Unicursal	1	108	108	108	1	0
T2_seg8_MH@24	8	19.625	8	28	13	2.51
Model 2_MH@24	26	5.5	3	12	–	–
T2_seg8_MH@41	8	21.125	8	29	14	3.14
Model 2_MH@41	25	7	5	14	–	–

S: Number of segments
stdev: Standard deviation of the lengths of segments

In conclusion, the simulation results show the effectiveness of the proposed BBT route scheme in our framework for an ideal network as well as a real network. It can keep the terminal paths short enough to avoid errors in loss rate estimation compared to the unicursal route, while keeping the number of terminal paths small enough to reduce the number of accesses to switches significantly low compared to the previously proposed shortest-path tree based route.

7. Concluding Remarks

We propose a practical framework for monitoring both directions of all full-duplex links of an entire OpenFlow-based network to promptly locate high-loss links with a minimized load on both data-plane and control-plane incurred by the measurement. The framework introduces a combination of an active measurement by probing multicast packets along a designed route and a passive measurement by collecting flow-stats of the probing flow at selected switch ports in an appropriate sequential order to access switches.

In particular, we have proposed the BBT route scheme. Through simulation-based evaluations on a real-world large network topology, the BBT route scheme was shown to balance between the measurement accuracy and the measurement overhead (the number of accesses to switches until locating all high-loss links). Note that our framework was implemented in the Ryu OpenFlow framework and tested on a Mininet environment and is scheduled for testing on a nation-wide OpenFlow testbed.

As a topic for future work we will strive to adaptively optimize schemes for the multicast probe packet route for flowing on every link and the access order to switches for collecting flow-stats by reflecting the past measurement results in continuous monitoring scenarios, e.g., by leveraging machine learning techniques.

Acknowledgments These research results have been achieved by the "Resilient Edge Cloud Designed Network (19304)," NICT, and by JSPS KAKENHI JP20K11770, Japan.

References

[1] Feamster, N., Rexford, J. and Zegura, E.: The Road to SDN: An Intellectual History of Programmable Networks, *ACM SIGCOMM Computer Communication Review*, Vol.44, No.2, pp.87–98 (2014).
[2] Jain, S., Kumar, A., Mandal, S., et al.: B4: Experience with a Globally-deployed Software Defined WAN, *Proc. ACM SIGCOMM'13*, pp.3–14 (2013).
[3] Hong, C.-Y., Kandula, S., Mahajan, R., et al.: Achieving High Utilization with Software-driven WAN, *Proc. ACM SIGCOMM'13*, pp.15–26

(2013).
[4] Tootoonchian, A., Ghobadi, M. and Ganjali, Y.: OpenTM: Traffic Matrix Estimator for OpenFlow Networks, *Proc. PAM 2010*, LNCS Vol.6032 (2010).
[5] Yu, C., Lumezanu, C., Zhang, Y., et al.: FlowSense: Monitoring network utilization with zero measurement cost, *LNCS*, Vol.7799, pp.31–41 (2013).
[6] Chowdhury, S.R., Bari, M.F., Ahmed, R. and Boutaba, R.: PayLess: A low cost network monitoring framework for Software Defined Networks, *Proc. 2014 IEEE NOMS*, 9 pages (2014).
[7] van Adrichem, N.L.M., Doerr, C. and Kuipers, F.A.: OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks, *Proc. 2014 IEEE NOMS*, 8 pages (2014).
[8] Atary, A. and Bremler-Barr, A.: Efficient round-trip time monitoring in OpenFlow networks, *Proc. IEEE INFOCOM*, pp.1–9 (2016).
[9] Shibuya, M., Tachibana, A. and Hasegawa, T.: Efficient active measurement for monitoring link-by-link performance in OpenFlow networks, *IEICE Trans. Commun.*, Vol.E99B, No.5, pp.1032–1040 (2016).
[10] Tan, C., Jin, Z., Guo, C., et al.: Netbouncer: active device and link failure localization in data center networks, *Proc. 16th USENIX Conference on NSDI*, pp.599–613 (2019).
[11] Peng, Y., Yang, J., Wu, C., et al.: deTector: A Topology-aware Monitoring System for Data Center Networks, *Proc. 2017 USENIX Annual Technical Conference*, pp.55–68 (2017).
[12] Duffield, N.: Network Tomography of Binary Network Performance Characteristics, *IEEE Trans. Information Theory*, Vol.52, No.12, pp.5373–5388 (2006).
[13] Tachibana, A., Ano, S., Hasegawa, T.:, et al.: Locating Congested Segments over the Internet Based on Multiple End-to-End Path Measurements, *IEICE Trans. Commun.*, Vol.E89-B, No.4, pp.1099–1109 (2006).
[14] Ma, L., He, T., Swami, A., et al.: Network Capability in Localizing Node Failures via End-to-End Path Measurements, *IEEE/ACM Trans. on Networking*, Vol.25, No.1, pp.434–450 (2017).
[15] Tri, N.M. and Tsuru, M.: Locating deteriorated links by network-assisted multicast proving on OpenFlow networks, *Proc. 24th IEEE Symposium on Computers and Communications* (*ISCC19*), pp.701–706 (2019).
[16] Goto, S., Shibata, M. and Tsuru, M.: Dynamic optimization of multicast active probing path to locate lossy links for OpenFlow networks, *Proc. 2020 International Conference on Information Networking* (*ICOIN*), pp.628–633 (2020).
[17] The Internet Topology Zoo, available from ⟨http://www.topology-zoo.org/⟩ (accessed 2020-05-14).

Nguyen Minh Tri was born in 1987. He received his M.S. degree from University of Science, Vietnam in 2014. He is currently a Ph.D student at Kyushu Institute of Technology, Japan. His research interest is network management and communication systems.

Masahiro Shibata was born in 1989. He received BE, ME, and DE degrees in Computer Science from Osaka University, in 2012, 2014, and 2017, respectively. Since 2017, he has been an Assistant Professor at Kyushu Institute of Technology. His research interests include network management distributed algorithms. He is a member of JPSJ and IEICE.

Masato Tsuru received B.E. and M.E. degrees from Kyoto University, Japan in 1983 and 1985, respectively, and then received his D.E. degree from Kyushu Institute of Technology, Japan in 2002. He worked at Oki Electric Industry Co., Ltd., Nagasaki University, and Japan Telecom Information Service Co., Ltd. In 2003, he moved to the Department of Computer Science and Electronics, Kyushu Institute of Technology as an Associate Professor, and has been a Professor there since April 2006. His research interests include performance measurement, modeling, and management of computer communication networks. He is a member of the ACM, IEEE, IEICE, and IPSJ.