Monitoring-Flow Based Network Verification and Failure Localization in Software Defined Networks

Bence Ladóczki*, János Tapolcai, Alija Pašić*
*MTA-BME Future Internet Research Group and MTA-BME Information Systems Research Group,
Budapest University of Technology and Economics (BME), Hungary, E-mail: {ladoczki, tapolcai, pasic}@tmit.bme.hu

***Abstract*—In this paper, we present the deployment of a monitoring-flow based network verification and failure localization approach for SDN networks. It not only minimizes the number of static forwarding rules but also significantly reduces the control plane load, i.e., reduces the total number of messages needed for network verification and failure localization. Our flexible hybrid implementation consists of MikroTik RB2011iLS-IN and Open vSwitch (Mininet) switches, enabling the user to test network verification and failure localization in a more complex manner even if the number of physical devices is limited.**

***Index Terms*—failure detection, failure localization, Chinese Postman Problem, Software Defined Networks, OpenFlow**

I. Introduction

Network verification and failure localization are essential for providing reliable services in SDN networks. However, today's OpenFlow based SDN architectures do not support protocols for neighborhood and topology discovery. Other solutions require control signaling between the switches [1]. Note that some controller frameworks utilize existing control plane solutions like Link Layer Discovery Protocol (LLDP) [2], however, continuous monitoring via LLDP is not a feasible approach [1] since this leads to an excessive number of messages which must be handled by slow software agents resulting in congestion.

In [1] an innovative solution was presented, where controllers deploy forwarding rules and utilize control flows (i.e., monitoring-flows) dedicated to topology verification and failure localization. The key idea that controllers periodically send probing messages in the network along the deployed routes, to check the status of the links it traverses. In the current work, we present an implementation of this method using OpenFlow, Mininet and MikroTik RB2011iLS-IN switches. In this paper, we overview the main challenges faced during the deployment.

II. Architecture

Figure 1(A) shows the architecture of the monitoring flow-based network verification and failure localization demo, while Figure 1(B) shows the flowchart of the POX controller application. The state machine represents the network verification process, where the execution is sequential after connecting the switches to the application.

We differentiate between two modes/states of our software:

1) *Network verification:* After the initialization of the controller application, it imports the detected network topology, computes a route (i.e., monitoring-flow) and identifies the required forwarding rules that will be deployed in the switches afterward [1]. The heart of the optimization is the Chinese Postman Problem (CPP) solver with a rule optimization module. Finally, the network verification is performed by sending heartbeat-like packages with a given (adjustable) frequency along the pre-configured monitoring-flow. As soon as the monitoring package transgressed the walk, it gets returned to the controller, implying that the network is intact.
2) *Failure localization:* In a less fortunate case, one of the switches fails to forward the packet due to link failure and the application switches its state to *failure localization*. As proposed in the original paper [1], the controller application uses a binary search to locate the single link failure along the route of the monitoring-flow by sending out packets while gradually shrinking the inspected interval. The packets, responsible for the localization are bounced back from the switches specified by their headers. Controller application loads these bounce-back rules during initialization.

Our physical test environment in the demo architecture (Fig. 1(A)) consists of an Apache web server, web clients and commercially available MikroTik RB2011iLS-IN routers compatible with the 1.0 OpenFlow protocol. The routers are programmed by a POX based highly portable Python controller application. Our implementation is capable of connecting physical and virtual networks (through Mininet), enabling the user to test failure detection and localization more complexly. Meaning that the user can initialize virtual SDN switches using Mininet and an arbitrary number of virtual-physical network interface pairs can be defined to create connections in the hybrid network. From SDN routing perspectives, SDN software switches are entirely identical to the Mikrotik routers, and we utilize this transparency in our framework.

We develop a *Graphical User Interface (GUI)* to enhance usability and visualize its operation, which is capable of displaying the network graph, launching the application and warning the user if a single link failure occurs. The GUI runs entirely independently (because of security reasons) from the controller application, they communicate using the web server's endpoint, and the GUI is updated using AJAX.

III. Use Cases for Demonstration

Our implementation can handle arbitrary topologies, and we demonstrate the operation of our software which minimizes the

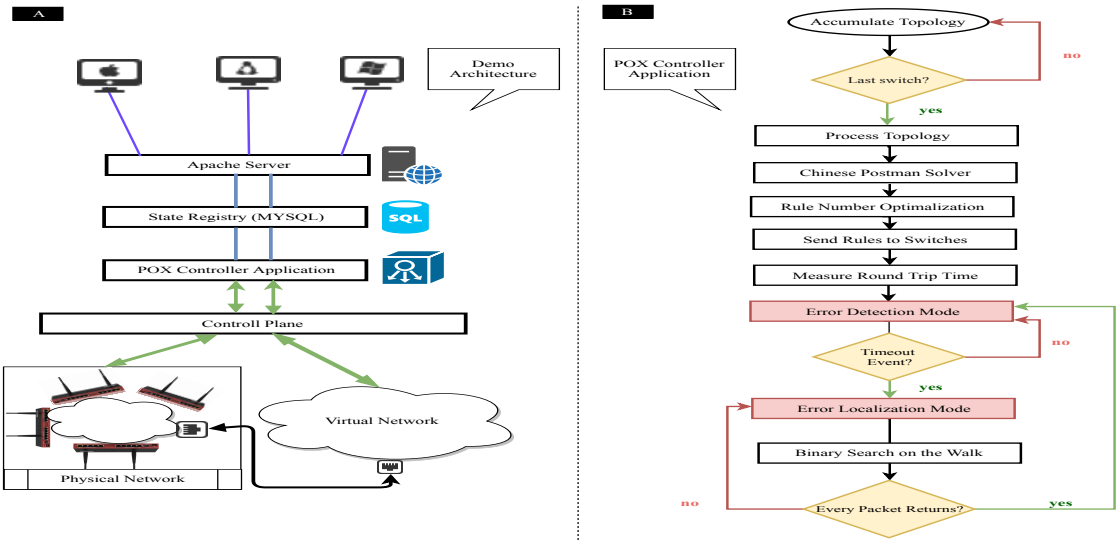

Fig. 1: Architecture of the Optimal Single Link Failure Localization Software Package

number of static forwarding rules needed and also significantly reduces the control plane load on various networks.

Topologies, where the walk transgresses many links more than once, are especially challenging because the implementation of the forwarding rules and the VLAN tagging is not straightforward. The VLAN tags are used for various purposes. First of all, we differentiate forwarding paths from one end to another and the path that transgresses the opposite direction. In addition to this, we use VLAN tags to mark the sequence of the nodes on a given walk. One of many exciting issues originates from the fact that real networks posses delays due to various factors that are omitted from a theoretical work [1]. Our implementation solves this problem by measuring the delays in the actual network and automatically adjusts the thresholds based on the measured delays. This measurement is performed during the initialization phase. The measured round trip time (RTT) and standard deviation (σ) with respect to the number of Open vSwitches (Ring size) is presented in Table I. RTT grows with the number of switches, however the

Ring size	20	70	100	150	200
RTT	11.0 ms	27.4 ms	27.7 ms	43.4 ms	48.3 ms
σ	0.3 ms	7.0 ms	7.7 ms	4.3 ms	13.0 ms

TABLE I: Round trip times (RTT) and their standard deviation in the virtual network with respect to the number of switches

dependency doesn't seem to be rigorously linear. This might be related to resource instabilities in the virtual environment.

During the demo session, we will demonstrate failure detection an localization using various network sizes and point out the importance of the self-calibrating RTT module that adjusts the detection threshold based on the measured round trip time in the network.

IV. Conclusions

We presented an implementation of a new monitoring-flow based network verification and failure localization framework [1] which minimizes the number of static forwarding rules and significantly reduces the control plane load. Our implementation is capable of connecting physical (commercially available MikroTik RB2011iLS-IN routers compatible with the 1.0 OpenFlow protocol) and virtual networks (Mininet - Open vSwitch based), enabling the user to test failure detection and localization in a more complex manner even if the number of physical devices is limited. We note that the scalability of the implementation can be maintained by defining subdomains on the topology and localizing in a parallel manner. The main lesson we have learned during the deployment is that message delay in real networks fluctuates thus to avoid false failure alarms when reducing the failure localization time we need to implement a mechanism that adjust the delay thresholds continuously.

V. Acknowledgements

The research leading to these results was partially supported by the High Speed Networks Laboratory (HSNLab). Project no. 123957, 129589, 124171, 128062 and 124171 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FK_17, KH_18, K_17, K_18 and K_17 funding schemes respectively. The research report in this paper was also supported by the BME-Artificial Intelligence FIKP grant of EMMI (BME FIKP-MI/SC).

References

[1] U. C. Kozat, G. Liang, K. Kokten, and J. Tapolcai, "On optimal topology verification and failure localization for software defined networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2899–2912, 2016.

[2] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in software defined networks," in *Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on*. IEEE, 2014, pp. 1–8.