Automated troubleshooting toolset for SDNs

Knop Thibaut, Rochet Florentin
EPL, UCL
Louvain-la-Neuve, Belgique
{thibaut.knop,florentin.rochet}@student.uclouvain.be

***Abstract*—TODO**

***Index Terms*—TODO**

I. Introduction

For many years, the debugging as always been one of the major concerns in network maintenance. In order to localize problems into the network, operators usually use a narrow toolset, composed of traditional tools as ping, trace route and SNMP agents [**?**].
Hopefully, this difficult and time-consuming [**?**] process could change, given the deployment of Software-Defined Networks. In a nutshell, SDNs are based on a separation between control- and data planes, which offer the opportunity of programmable networks [**?**]. Those SDNs are composed of a network of switches managed by a logically-centralized controller, whose role is to (un)install rules into the flow table of the switches, to read traffic statistics and respond to the network activity.
However, and because SDNs allows different operators, developers to dynamically program the same network, the complexity of software will increase [**?**] and potentially the numbers of bugs. To minimise the trade-off between introducing new functionality and increase the number of bugs in the network, there is a serious need for a complete and effective automated testing toolset, allowing the admins to focus on fixing the issues instead of localizing them. In traditional network architecture, it is almost impossible to create such an automated test suite, due to the complexity of *knowing the operator's intent* and *checking network behavior against intent* [**?**] (for more information about how traditional networks could be extended to support automated troubleshooting, please refer to [**?**]).
As we will explain, we use the different layers of the SDN stack to review the available tools for automated troubleshooting. Note that the methodology and the structure of this paper is influenced from the one used in [**?**]. It indeed seems the better way to articulate and present the different tools destined to localize the problems and their cause in an automated way. The paper is structured as follow : first we recall the different layers of the SDN stack, and how it can be leveraged to provide automated troubleshooting for SDNs, then we present different existing tools by positioning them regards to the SDN layering, and we eventually conclude.

II. SDN layering, the key used to a better troubleshooting

Finding and solving network bugs are not the aim of SDN, but we can use it to re-think the way we troubleshoot networks.
The SDN architecture is decomposed into layers, those layers can be represented in a two dimensionnal array. As you can see on Figure **??**, we have the two main layers called *State layers* and *Code layers*. The state layers hold a representation of the network's configuration for each parts of the network architecture. The code layers implement logic to maintain the mapping between two state layers. Each states layers should verify the equivalence properties, which means that each of them should correctly mapping every other state layer. The idea is that, for each policy, if the state layers are correctly mapped among each other, then the policy is set and acts like it should.
Thus, thanks to the SDN stack, we are able to first build a tool to check consistency between state layers in order to identify on which part of the network architecture a bug is happening.
When the layer is identified, an other tool take over to localize the issue inside the code layer. We will see in (TODO: indicates next section) which kind of tools could be used to handle that.

- explication graphique
- plus en dÃľtail finding the code layer and finding within the code layer. (summary)

III. Network Troubleshooting - Tools

A. tool1 - 1 subsection par tool

- NDB - prÃľsenter un outil pour finding the code layer
- Veriflow
- Soft
- Nice
- FlowChecker ?

Expliquer ici par rapport au layer

voir page 2 a Nice way (challenge pour tester l'open flow) et les solutions qui existent)

IV. Conclusion

TODO

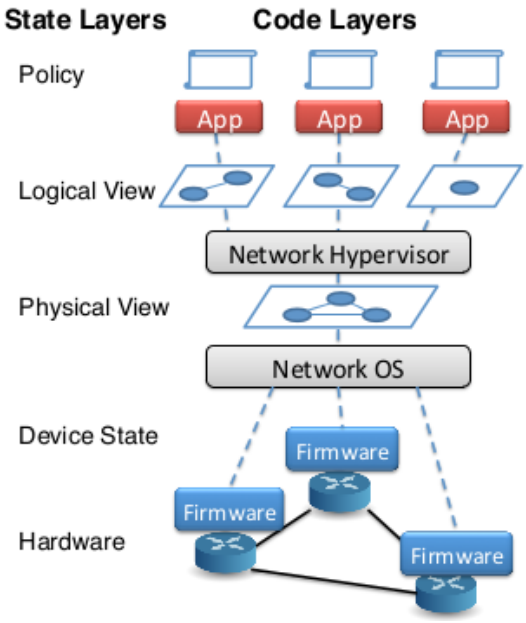

Fig. 1. SDN architecture