Understanding Bugs in Software Defined Networking

Paper #0, 2 pages

ABSTRACT

Software-Defined Networking (SDN) has enabled innovative solutions in the domain of networks by decoupling the control plane and data plane. Bugs form an inevitable part of any software development cycle and in turn causes a divergent behaviour which could lead to fail-stop errors and byzantine faults. Developing a taxonomy of bugs seems to be a promising way to lay the the foundation required for combating against these critical bugs as well as open new research directions for the development of bug-tolerant system. In this paper we analyze 150 critical bugs in three most widely-used SDN controllers i.e FAUCET , ONOS , CORD. We evaluate the root causes , bug types , trigger conditions , location of the fault followed by their outcome and fixes. To the best of our knowledge this study is the first of its kind in the area of SDN and concludes a detailed taxonomy for SDN bugs.

1. INTRODUCTION

Software Defined Networking (SDN) has enabled a paradigm shift from the naive legacy networks to flexible programmable networks . It has been adopted by major Telco companies such as Vodafone , AT&T , Orange labs as well as major internet content delivery service providers such as Facebook , Google . Adoption of SDN by all the above major companies has enabled them to simplify Provisioning and Management of underneath networks , better usage of all the network resources available for disposal , as well as lower CAPEX and OPEX . But the adoption pace seems much slower than the expected , this behaviour has been expected due to no of factors such as Network scalability , Immaturity of supporting technology but the most important factor that requires immediate action is Network reliability. Reliability and availability are the most vital performance indicators for the networks as they ensure uninterrupted service is delivered by high performance networks.

Bugs are indigenous to almost all software systems and hinder the availability and reliability of the system to a great extent , following this SDN is no exception. Bugs often lead to fail stop errors , endless staling as well as byzantine faults i.e these are the faults that lead the system to behave incorrectly at the same time violating the protocol specifications.

Controller	Goal	Implementation
FAUCET	Compact Easy to debug No inter-dependency No external database	multi-table packet processing 1000's LOC
ONOS	Modular code Configuration flexibility Isolation of subsystems Protocol Agnosticism	tiers of functionality services, managers provider, store 1,000,000's LOC
Open CORD	CORD Vision	CORD distribution

Table 1: Target SDN controllers with their vision and implementation

Every Software Defined network depend on the appropriate behaviour of three components i.e SDN application , SDN controller , Network elements.

TO DO : more details on previous works and introduction to be added

2. METHODOLOGY

2.1 Target Systems

To develop a detailed Taxonomy of bugs in SDN we have focused on three most widely used controllers i.e open CORD [1], FAUCET [2] , ONOS [3] . As highlighted in Table 1 all these controllers showcase different architectures to support different visions by using very diverse code bases. In this section we briefly introduce all the three controller and their components.

Open CORD(Central Office Re-architected as a Datacenter) [1] uses a hybrid approach involving SDN , NFV , Cloud to leverage innovation in Teleco Central Office (CO) and replace purpose built hardware to build cost-effective agile networks. CORD utilizes four opensource projects Openstack , ONOS, Docker, XOS.

FAUCET [2] boast a very compact code-base with the aim to migrate network functionalities such as routing protocols

, neighbour discovery to vendor-independent firmwares. It's latest release utilizes OpenFlow 1.3 specifications at the southbound interface . It is composed of two controller components namely GAUGE and FAUCET itself. FAUCET manages all the forwarding decisions as well as switch states whereas on the other hand GAUGE maintains a consistent openflow connection with the switch to monitor the port and flow states followed by sending it out to influxDB or Prometheus for later reference.

ONOS (Open Network Operating System) [3] was developed with the mission to cut out the proprietary and closed nature of current networking system and develop an open source network operating system that would enable innovative solutions for the evolvement of SDN. ONOS utilizes the concept of services and subsystem for implementing the control plane where the collections of all the sub-components that form the service are called the "subsystem". Provider , manager and store constitutes as the basic functioning units of each subsystem.

2.2 Gathering Bugs

Among the three targeted systems, ONOS and CORD employ a well-organized way of issue tracking with JIRA and they utilize gerrit for rolling out fixes collaboratively whereas the FAUCET community uses github itself for bug reporting followed by merging a fix to the master. We start our study from scratch due to the lack of existing database and analyze the issues or bugs that were critical to the developer community. FAUCET, ONOS , CORD community has reported 149 ,172 ,195 such critical bugs that needed immediate action at this moment.

Unfortunately , we couldn't cover all the issues and fixes as each bug requires an intensive analysis of code for understanding the fix , so instead we pick 50 bugs at random from each controller and perform a complete manual static analysis to categorize all the bugs in a meaningful taxonomy.

TO DO :A table showing categories and sub categories

3. ROOT CAUSE

TO DO :A table showing root causes and their distribution among the three controllers

3.1 Load

3.2 Concurrency

3.3 Memory

3.4 Missing Case & Human

4. LOCATION

TO DO : A horizontal bar plot with percentage distribution for each location across three controllers

4.1 SDN Controller

4.2 Background services

5. OUTCOME

TO DO : Heatmap between root causes and outcome combined for three controllers

5.1 Slow Performance

5.2 Crash

5.3 Error Message

5.4 Wrong Behaviour

6. FIX

TO DO : Heatmap between root causes and fix separate for three controllers

6.1 Fix configuration

6.2 add logic

6.3 add compatibility & Package upgrades

6.4 add synchronization

6.5 workaround

6.6 Restart & Rollback upgrades

7. IMPLICATIONS

8. RELATED WORKS

9. CONCLUSION

Acknowledgments

10. REFERENCES

[1] Cord retrieved from. https://opencord.org/. Accessed: 10-11-2018.

[2] Faucet retrieved from. https://faucet.nz. Accessed: 10-11-2018.

[3] Onos retrieved from. https://onosproject.org/. Accessed: 10-11-2018.