Faul Management in Software Defind Network

Hari Achmad Aulia

Abstract- Software-defined networking (SDN) has emerged as a new network paradigm that promises control/data plane separation and centralized network control. While these features simplify network management and enable innovative networking, they give rise to persistent concerns about reliability. The new paradigm suffers from the disadvantage that various network faults may consistently undermine the reliability of such a net-work, and such faults are often new and difficult to resolve with existing solutions. SDN controllers may lose its normal functions when it is attacked by hackers or infected with viruses. In order to provide a fault-tolerant and reliable computing environment in distributed SDNs, we design a fault-tolerant consensus protocol to improve fault tolerance of distributed SDNs. Therefore, comprehensive reviews and constant improvements are required to remain on the leading edge of SDN fault management. In this paper, we present the first comprehensive and systematic survey of SDN faults and related management solutions identified through advancements in both the research community and industry. We apply a systematic classification of SDN faults, compare and analyze existing SDN fault management solutions in the literature, and conduct a gap analysis between solutions developed in an academic research context and practical deployments. The current challenges and emerging trends are also noted as potential future research directions. This paper aims to provide academic researchers and industrial engineers with a comprehensive survey with the hope of advancing SDN and inspiring new solutions.

***Keyword**: software-defined network, fault-tolerant mechanism*

A. INTRODUCTION

Due to the rapid development of cloud computing and the advent of the internet of thing (IoT) era, the scale of the internet has been expanding in an amazing speed. However, the network traffic has become so huge that the current IP network instrastructure, routers execute various routing protocols to assist in packet forwarding. To solve the above-mentioned problem with the traditional IP network infrastructure, a software-defined networking (SDN) architecture is proposed. The SDN architecture was officially introduced by Professor McKeown in a keynote speech in 2009.

We have implemented our fault management schemes in the Mica2 motes on the SOS kernel [5]. We perform a series of statistical fault injection studies to evaluate the coverage and detection latency provided by our software-based fault detection schemes. According to the experimental results, our proposed fault detection schemes not only have short detection latency but also provide high fault coverage. For example, for permanent processor faults, our fault detection rate is about 98%. Besides, current networks also must meet other require-ments. Fault tolerance, reliability, and resiliency to failures are amongst the most important ones. Usually, cloud providers, big data services, and cellular network carriers must follow strict service level agreements (SLAs), which frequently specify desired values for metrics such as response time, processing time, the rate of failure, maintenance time and data losses. Failures have a major financial impact on service providers. For example, from 2007 to 2013, cloud networks from 28 cloud providers amass losses estimated at US$273 million and 1,600 hours of disruptions, due to application and infrastruc-ture failures [5].

The proposed methodology introduces a fundamentally dif-ferent mindset compared to the techniques applied today. Rather than facilitating easier interaction of the design engineers with the system representation or focusing on the system behavior rather than the system architecture, the proposed method pro-vides a coherent description of the system architecture that can be

processed by a computer. To this end, consistent operation rules based on logic operators are proposed, and adherence to these operation rules is defined as the key measure of the accuracy of the system representation. In the final outlook, any fuzzy or gradually changing parameter aggregates into a binary decision; for fault tree analysis, it is one of the conditions that should be satisfied for the application of minimal cut set analysis [11], while for logistics purposes, it corresponds to the decision whether to replace a unit or not. The proposed technique functions by this paradigm, where the system organization is described in terms of absolute dependencies between elementary system components. Using this binary approach, a method is developed to test the integrity of systems description, isolate faults in a system, analytically assess the BITE coverage, and generate proposals for BITE locations in a system. The method can be developed further to automate the FMECA and fault tree analysis, significantly reducing the need for the opinion and time of expert engineers.

To control the effects of these faults, several techniques are used such as system state monitoring, fault detection, localization and resolution, and fault tolerance mechanisms. These techniques are collectively referred to as fault management techniques. In the domain of SDN, multiple recent investigations have been conducted on fault management solutions, including software fault troubleshoot-ing policy conflict arbitration, forwarding path verification network behavior inspection network measurement as well as fault recovery and tolerance design These studies have greatly contributed to improving the reliability of SDN. However, we find that most such studies resolve SDN faults from only a partial perspective, not a global one; this may result in incomplete and flawed solutions and may even induce other side effects. More seriously, as the network paradigm evolves, more potential faults are being exposed. Thus, it is necessary to conduct a comprehensive and systematic survey of SDN faults and related management solutions, accompanied by an in-depth discussion and analysis, to provide researchers and engineers with a foundation for motivating continual improvements in SDN fault management

B. CONCEPT AND APPROACH

In this section, we introduce the concept and approach of the proposed Consensus Protocol for SDN (*CPSDN*). The proposed *CPSDN* is used to solve the consensus problem in SDN. First, each controller will select an initial value from the domain range $D = \{0,1\}$. Next, the controllers will exchange its initial value with other controllers in the SDN. After completing the message exchange, each controller will use the collected messages to compute its consensus value. That is, there two phases in the proposed *CPSDN*, the two phases are message exchanging phase and consensus making phase.The pseudo code of the *CPSDN* protocol is shown in Fig. 2. The functions involved in the proposed *CPSDN* protocol are listed as follows:

- *crt*(p, *v*): create the vertex p, and set *val*(p) =*v*.
- *pack*(*i*, *msg*): according the structure of level *i* of the SDN- tree, pack level *i* of the SDN-tree to message *msg*.
- *unpack*(*i*, *msg*): according the structure of level *i* of the SDN- tree, unpack message *msg*.
- *send*(Z, MSG, *nj*, *msg* Z, *nk*): send a MSG message with the value *msg* proposed by controller n_j to n_k.
- *dormt*(n_j): check the controller n_j is a dormant controller or not
- del_{rpt}(SDN-tree): delete the vertices with duplicate signs in the SDN-tree.

Software defined networks

In traditional networks, control plane and data plane are tightly coupled in network devices. Conversely, SDN keeps only data forwarding functionality in network devices, whereas it delegates control plane functionality to a physically separate layer, composed of one or more network entities called con-trollers, which provide the interface between the high-level

network applications and the network devices. As depicted in figure 1, SDN architecture comprises three layers (in the course of this work, we use the terms layer and plane interchangeably):

Infrastructure layer: Also known as the data plane, it is responsible for data forwarding and statistics storage; it may include physical and virtual switches. Network devices must comply with a standard interface (e.g. OpenFlow protocol [20]) that is used by the control plane to manage the devices.

Control layer: The control layer consists of one or more SDN controllers. The main function of the control layer is to maintain a logically centralized network view that allows network applications reason about network prop-erties and behavior. The control layer observes network state through the open interface with the devices, and provides an application programming interface (API) to construct network applications using high-level terms (e.g. host names instead of IP addresses).

Application layer: Uses the API provided by the control layer to implement network business applications (e.g. firewall, load balancer), enforcing networking policies and requirements.

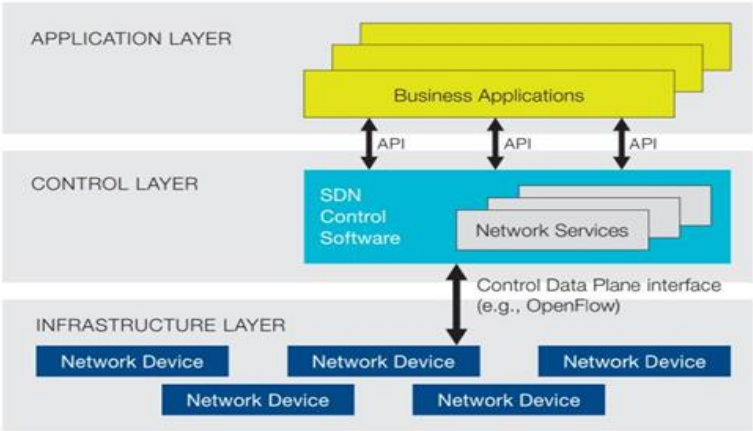

Fig. 1: Software defined networking architecture

C. CONCLUCION

This work presents a comprehensive view of fault manage-ment in SDN. Our goal was to identify which fault manage-ment issues are present in SDN, how current efforts address those issues, what are the major contributions of those efforts and what are the major gaps in the ongoing researches.

We have identified fault management issues of each layer/interface. We have observed that most of fault man-agement issues raised by SDN are related to its layered architecture and logical centralization of control. Faults in each layer may affect other layers in different aspects, for example, a faulty application may cause a black hole in the network, as well as failures in communication between layers (e.g., controller-switch communication). A logically centralized control plane is radically different from legacy networks, raising new issues, such as controller placement, control channel reliability and controller failure.

REFERENCES