TREMA BASED DEPENDENCY ANALYSIS SYSTEM FOR LEARNERS IN BUILDING SDN NETWORK CONSTRUCTION EXERCISES

Hlwam Maint Htet

Faculty of Computer Systems and Technologies, University of Computer Studies, Yangon,
Shwe Pyi Thar Township, Yangon, 11411, Myanmar
hlwammainthtet@ucsy.edu.mm
https://ucsy.edu.mm

Amy Tun

Faculty of Computer Systems and Technologies, University of Computer Studies, Yangon,
Shwe Pyi Thar Township, Yangon, 11411, Myanmar
amytun@ucsy.edu.mm
https://ucsy.edu.mm

Abstract

In the field of Software-defined networking (SDN), it has gained significant traction in research and innovation, with universities and research labs being pivotal in fostering academic advancements that accelerate industrial change. This research work has been motivated especially for e-Learning platform and have been developed to educate learners. While the users build SDN construction exercises, novice learners often struggle to understand network behavior and meet communication data requirements. In this system, learners use Trema to construct SDN exercises leveraging the OpenFlow protocol for communication between controllers and switches. Learners encounter challenges, such as ping failing to identify delivery routes through switches, switches lacking the functionality to log rules for output port selection, and Trema being unable to track execution statements for setting rules. To mitigate these challenges, the system offers visual configuration to help learners and resolve incorrect communication issues.

Keywords: SDN, network construction exercises, Trema, OpenFlow.

1. Introduction

With the rapid growth of science and technology, virtualization technology has gained significant momentum and remains a highly efficient tool in the networking era. SDN, a form of virtualization technology, is widely adopted. Educational institutions have developed SDN-based network construction exercises, providing valuable experience for network administrators, application programmers, and system designers. However, novice learners often struggle to understand network behavior and identify bugs in their settings when using Trema for SDN exercises.

Trema [1] is a modular, open-source framework designed for developing and testing Software-Defined Networking (SDN) applications. It provides a comprehensive environment for building SDN controllers by supporting the OpenFlow protocol, which facilitates communication between the control plane and the data plane in an SDN architecture. Trema enables developers to write, test, and debug their SDN applications efficiently by offering tools and libraries that simplify the process of managing network traffic and resources through software rather than traditional hardware-based network devices [3]. Its flexibility and ease of use make it a valuable tool for both educational purposes and practical implementations in the field of network management and operations. High level architecture of Trema is depicted in figure 1 [3].

Unlike other OpenFlow controller platforms such as NOX/POX [4], Beacon [5], Floodlight [6], and ONIX [7], Trema stands out for its strong emphasis on boosting the productivity of network researchers. Trema is more than just an OpenFlow controller; it serves as a comprehensive OpenFlow programming framework that encompasses the full development lifecycle, from programming and testing to debugging and deployment.

To assist them, the system offers three clues to help narrow down missettings in SDN construction exercises. SDN, or Software-Defined Networking, separates the control plane from the data plane, enabling dynamic network management through software controllers instead of traditional hardware-based devices.

2. Problem Definition

Trema [1] is a framework for developing OpenFlow controllers in Ruby and C. Open vSwitch [2] is a virtual switch that conforms to OpenFlow. In the SDN construction exercises examined in this study, students implement controller programs with Trema and build software-defined networks on their own PCs with the controllers and Open vSwitches. The exercise problems consist of settings of data plane networks, shell scripts for building the networks, whole network topologies, and requirements of reachability and communication routes. Students' networks must satisfy the requirements. However, students often fail to solve the problems because they cannot find errors in their controller programs, and thus cannot correct them.

There are causal connections between packet transmissions in the data plane and transmissions in the control plane and statement executions in the controller program. For example, a switch receives a packet in the data plane and then inquiries how to process the packet to the controller. The controller receives the query from the control plane and then processes the query by executing the statements. Students are expected to succeed in debugging their controller programs by analyzing the causal connections.

3. Related Work

This section describes the related works for narrowing down missettings in SDN construction exercises. In [2], authors describe the method of students' learning state by generating colour-coded time chart of learning status in which students are experiencing learning setbacks. “A fast method of verifying network routing with back-trace header space analysis”, Toshio Tonouchi Satoshi Yamazaki, 2015 IEEE proposed a new method of fast verification that is called back-trace header space analysis (B-HSA), a variation of the HAS in [3]. It provides a proof that the proposed method can correctly verify reachability and isolation. Here, it cannot detect a loop route, which is a critical failure in the network and it does not calculate the header-spaces at outgoing ports.

“Determining Learning Status in SDN Construction Exercises”, Takashi Yokoyama, Hisayoshi Kunimune, Masaaki Niimura, Shinshu University Japan, E-Learn 2016 - Washington, DC, United States, November 14-16, 2016, focus on detecting students who experience learning setbacks. The purpose of this paper is to identify students experiencing setbacks in writing source code for an OpenFlow controller in real time [2]. Proposed a method for identifying struggling students by coloring on time chart about learning states.

A System for Generating Hints on Network Construction Exercises for Beginners”, Yuichiro Tateiwa, Nagoya University, Japan, ICCE 2016 IEEE in [4]. The aim of the system is to facilitate the learning process by giving clear, visual, and descriptive feedback on mistakes, helping beginners understand and correct their errors more effectively that were faced by beginners in network construction exercises. Traditional exercise problems are often described in natural language, making it difficult to logically discuss and identify fault regions. To overcome this, a formal representation defining the composition of these exercise problems is required.

4. Theory and Background

This section describes how the system was proposed and which methods are used in this system.

4.1. *SDN and Openflow Controller*

The concept of Openflow is decoupling control and data planes**.** It enables the separation of network control and forwarding functions, allowing centralized control over how traffic is forwarded in a network.

The key elements of Openflow are;

Controller-Device Communication: OpenFlow establishes a standardized communication protocol between the centralized controller and the network devices (switches and routers).

Flow Table: Network devices maintain flow tables that define how incoming packets should be processed. These tables contain rules (flows) that dictate packet handling based on fields like source/destination IP addresses, ports, VLAN tags, etc.

It may have the following challenges:

- **Security:** Centralized control introduces potential security vulnerabilities, requiring robust security measures to protect the controller and network.
- **Integration:** Integrating OpenFlow into existing network infrastructures can be complex and requires careful planning.

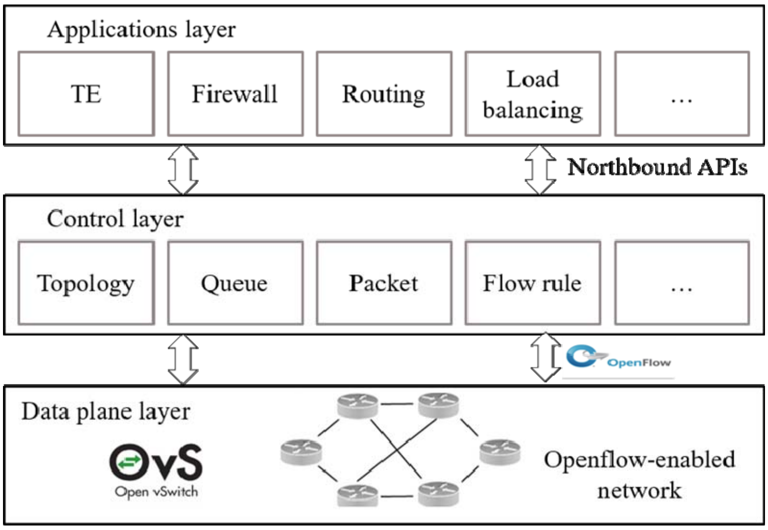

Figure 1. Software Defined Networks Architecture.

4.2. *Trema*

Trema serves as a toolkit for developers working with OpenFlow controllers and SDN applications. It facilitates the creation of custom network control software, allowing developers to manage and monitor network flows dynamically.
To use Trema effectively, developers typically follow these steps:

- **Installation**: Start by installing Trema and its dependencies on your development environment.
- **Development**: Write SDN applications using Trema's Ruby-based APIs, implementing control logic and flow management.
- **Testing**: Use Trema's tools to test and debug your applications, ensuring they perform as expected under different network conditions.
- **Integration**: Integrate Trema with OpenFlow-compatible switches and network hardware to manage and control network traffic according to your application's requirements.

The key features of Trema include;

- **Ease of Use**: The Ruby API makes it easy for developers to create and manage OpenFlow applications.
- **Modularity**: Trema's modular architecture allows for easy extension and customization.
- **High Performance**: The C core ensures efficient packet processing and low-level operations.
- **Comprehensive Documentation**: Trema offers extensive documentation and examples to help developers get started quickly.

4.3. *Plant UML*

PlantUML[12] is a tool that allows you to create diagrams from plain text descriptions. It's particularly useful for generating UML diagrams in an easy-to-read format, which can then be integrated into various documentation and codebases. The PlantUML code will generate a diagram illustrating the key elements of the system, how they interact, and the flow of the process from learners participating in exercises to receiving help through the analysis provided by the system.
To run PlantUML, we need a Java Runtime Environment (JRE) installed on our system, as PlantUML is a Java-based tool. Here's a detailed list of the requirements and steps to get started with PlantUML:

1. J Java Runtime Environment (JRE):
    - PlantUML requires Java to be installed on your system. You can download it from the official Java website.
    - Verify your Java installation by running java -version in your command prompt or terminal.
2. PlantUML Jar File:
    - Download the PlantUML jar file from the official PlantUML website.

5. System Implementation

The objective of this paper is to assist learners in identifying errors during network construction exercises by providing useful clues. To achieve this goal, we incorporated a standard output function (comprising prefix, line number, and clock) into all statements of the controller program.

In the proposed system, users first submit the network configuration information and controller program for their exercise. The data collection function then initiates packet capture and Trema, logging the user's communication test activities. Upon completion of the communication test and the user's termination request, the data collection function halts packet capture and Trema, retrieves packets, OpenFlow messages, and execution history, and stores them in the log DB as packet information, OpenFlow information, and executable statement information.

Following the conclusion of data collection, the system sequentially executes the time series flow table reproduction function, route selection imitation function, and transmission route estimation function, generating and saving the time series flow table, route selection history, and transmission route information in the log DB. The clue generator function extracts the necessary information from the log DB and provides the user with the clues to help identify and resolve errors. Figure 2 shows an example output of the system.

6. Experiment

In this system, we set a communication network with three different controller programs as sample exercises with one testbed network topology in Figure 2. The achievement condition is all host can communication with ping each other. The evaluation environment includes Host OS: Windows 10, Host CPU: Intel(R) Core (TM) i7-6500U CPU @ 2.59GHz, Virtual Machine: VMware Workstation 15 Player, Guest OS: Ubuntu 16.04LTS and Guest memory assignment is 1GB. The visualizer is implemented with Ruby and C language, Plant UML is used to convert the description to graphics. It might make it possible for us to achieve our goals. The system output result is shown in Figure 2 and the analysis result is in table 1.

Table 1. Analysis result

Controller Program	Controller Evets	Network Reachability	Packet Information (KB)	Openflow Information	Executed Statement Information
1	High	Yes	18	38	2
2	Medium	Yes	22	24	3
3	Low	Yes	25	20	4

Figure 2. Output of the system

7. Conclusion and Future Work

In our system, we have conducted an analysis of the dependency between communication activities in SDN construction exercises to help learners easily identify missettings. Utilizing the OpenFlow controller Trema, we implemented the simulator with Ruby and C [11], while the visualizer is created with the Plant UML Library [12]. This configuration allows us to effectively achieve our objectives.

Our work proposes a system for analyzing dependency in communication activities within SDN construction exercises using Trema. Learners often cannot identify missettings in SDN controllers due to various reasons, such as ping not finding delivery routes including switches, and switches not having the function to log rules. For learners facing difficulties in narrowing down errors, the proposed system provides three valuable clues from the dependency analysis: identifying rules, narrowing down switches, and pinpointing executed statements that lead to incorrect communication. Additionally, we evaluated the system's performance and effectiveness in aiding students' debugging processes.

Conflict of Interest

Declared conflicts of interest do not exist for the authors. There are no financial conflicts to disclose,

Acknowledgments

I would like to thank Dr. Amy Tun, Faculty of Computer Systems and Technologies from University of Computer Studies, Yangon, Myanmar and Professor Dr. Yuichiro Tateiwa, Nagoya Institute of Technology, for their kind guidance and supervising to complete and publish this research work.

References

[1] "Trema: A Brief Introduction and Tutorial", Shuji Ishii, Eiji Kawai, APAN 32nd Future Internet Testbed Workshop, 2011.

[2] Takashi Yokoyama, Hisayoshi Kunimune, Masaaki Nimura, "Determining learning status in SDN construction exercises", E-learn 2016, Washington, DC, United States, November 14-16, 2016

[3] "Programmable Network Using OpenFlow for Network Researches and Experiments", HIDEyuki Shimonishi, Yasuhito Takamiya, Yasunobu Chiba, Kazushi Sugyo, Youichi Hatano,Kentaro Sonoda, Kazuya Suzuki, Daisuke Kotani, and Ippei Akiyoshi,, ICMU 2012, Japan.

[4] NOX, POX, available at http://www.noxrepo.org/

[5] Beacon, https://openflow.stanford.edu/display/Beacon/Home

[6] Floodlight, http://floodlight.openflowhub.org/

[7] T. Koponen, M. Casado, N. Gude, J. Stribling, L.Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A Distributed Control Platform for Large-scale Production Networks. In the Proc. of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI10), 2010.

[8] "A fast method of verifying network routing with back-trace header space analysis", Toshio Tonouchi Satoshi Yamazaki, 2015 IEEE.

[9] "A System for Generating Hints on Network Construction Exercises for Beginners", Yuichiro Tateiwa, Nagoya University, Japan, ICCE 2016 IEEE.

[10] Alexander Shallimov, Dmitry Zulkov, Daria Zimarina, Vasily Pashikov, Ruslan Smeliansky, Advanced Study of SDN/OpenFlow Controllers, CER-SCER'13 Moscow, Russia

[11] "Proposal of an event visualization system for debugging in software-defined networking exercises using Trema", Yuichiro Tateiwa, Nagoya Institute of Technology, Japan, ICCE 2020 IEEE.

[12] PlantUML: https://plantuml.com/

Authors Profile

In 2013, **Hlwam Maint Htet**, earned her Master of Computer Science (M.C.Sc) degree with credits from the University of Computer Studies, Pakokku, Myanmar. She is currently pursuing her Ph.D at the University of Computer Studies, Yangon, Myanmar. She is currently working as a Staff Officer in the Internal Revenue Department, Ministry of Planning and Finance, Myanmar and working as a Chief System Manager of IT Infrastructure and Cyber Security Section. Her other research interests are Networking, Virtualization Technology and Cyber Security. She can be contacted at email: hlwammainthter@ucsy.edu.mm.

Amy Tun, is currently working as a Professor at the Faculty of Computer Systems and Technologies, University of Computer Studies, Yangon, Myanmar. Her other research interests are Internet of Things and Embedded System. She can be contacted at email: amytun@ucsy.edu.mm.