

EXTRACTION OF TRAFFIC FEATURES IN SOFTWARE-DEFINED NETWORKS USING AN SDN CONTROLLER

S.S. Volkov^{1,2,a}, I.I. Kurochkin³

¹ *Peoples' Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation*

² *Federal Research Center "Computer Science and Control" RAS, Moscow, Russia*

³ *IITP RAS, Moscow, Russia*

E-mail: ^a volkserg1@gmail.com

Machine learning methods can be used to solve the problems of detecting and countering attacks on software-defined networks. For such methods, it is necessary to prepare a large amount of initial data for training. Mininet is used as a modeling environment for SDN. The main tasks of modeling a software-defined network are studying traffic within the network, as well as practicing various scenarios of attacks on network elements. The SDN controller ONOS (Open Network Operating System) is used as the network controller. Various network topologies are considered in the modeling. The possibility of analyzing information about traffic within the network using an SDN controller in real time is investigated, as well as the possibility of collecting information in the form of a set of features. Modeling of software-defined networks under different initial conditions and for different attack scenarios can be carried out on a distributed computing system. Since the computational problem to be solved can be divided according to the data into many autonomous tasks, it is possible to use desktop grid system and voluntary distributed computing to speed up the process.

Keywords: software-defined networks, traffic features, network modeling, desktop grid

Sergey Volkov, Ilya Kurochkin

Copyright © 2021 for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1. Introduction

Software-Defined Network (SDN) [1][2] is now becoming more popular technology using in large data processing centers because it allows you to dynamically control network traffic flows. The main feature of SDN is the separation of the network control layer from the packet transmission layer.

This work is devoted to the issue of modeling SDN in order to obtain data for the further operation with machine learning algorithms. The main task of the study is to select tools and develop a complete data collection process. The work considers the following aspects of SDN modeling:

- Software-defined network emulator.
- SDN controller.
- Development of network topologies.
- Network traffic generation.
- Traffic data collection and features extraction.

The study attempts to solve two tasks. The first is collecting traffic flows data within a software-defined network using a SDN-controller. The second task is extraction of traffic features which can be used for machine learning algorithms in intrusion detection systems.

2. SDN Modeling

2.1 SDN emulator

Among the existing network emulators, the following can be distinguished:

- Mininet [3] – computer network emulator that can host SDN on a single machine (virtual or physical). Mininet is well suited for deploying test environments. It is also suitable for testing custom SDN topologies. The virtual network is deployed with all switches, controllers and hosts, and then its performance can be verified using scripts.
- Ns-3 [4] – simulator for discrete-event network modeling. It is also can be used for testing SDN environments.
- OpenNet [5] – the SDN emulator that is based on the two previous tools (Mininet and ns-3).
- Containernet [6] – fork of Mininet for working with application containers. Docker containers act as hosts of emulated networks.
- TinyNet [7] – A lightweight library that helps you quickly prototype SDN networks. But this tool is not suitable for emulating large-scale networks due to limited functionality.
- MaxiNet [8] – tool that allows you to use Mininet on multiple physical machines and work with large-scale SDN networks. Each of the machines runs Mininet and emulates its part of the general network. Switches and hosts communicate with each other using GRE tunnels. To manage the components of such a network, MaxiNet provides an API.



Since for this task we do not need too overloaded functionality, and we also need the ability to work with custom topologies, it was decided to use Mininet. The computer network in Mininet is deployed within one virtual machine. A computer network means simple systems that consist of hosts, switches and OpenFlow-controllers.

2.2 SDN Controller

The study examines various network topologies. Some topologies have loops. One of the main criteria for choosing a controller is its ability to manage a network with loops in a topology. During the study, two controllers were selected: ONOS [9] and Open Daylight [10]. Table 1 shows a small comparison of these controllers.

Thus, the ONOS controller turned out to be the most suitable option for our task. The ability to graphically display statistics in the web interface and support for loops in the topology make this controller a more versatile solution.

Table 13 – Features of two compared SDN-controllers

	
<ul style="list-style-type: none"> ● Release: 0.4.4-Beryllium-SR4 (2016) ● Topology Loop Support ● Availability of a web interface ● Later versions lack the web interface ● Later versions do not support topology loops 	<ul style="list-style-type: none"> ● Release: 2.4.0 Uguisu (2020) ● Topology Loop Support ● Availability of a web interface ● Ability to view data flow and connection load in real time ● Traffic analysis services ● Services for viewing statistics

2.3 Network topologies

Figure 1 presents the types of network topologies that were tested for selected SDN controller.

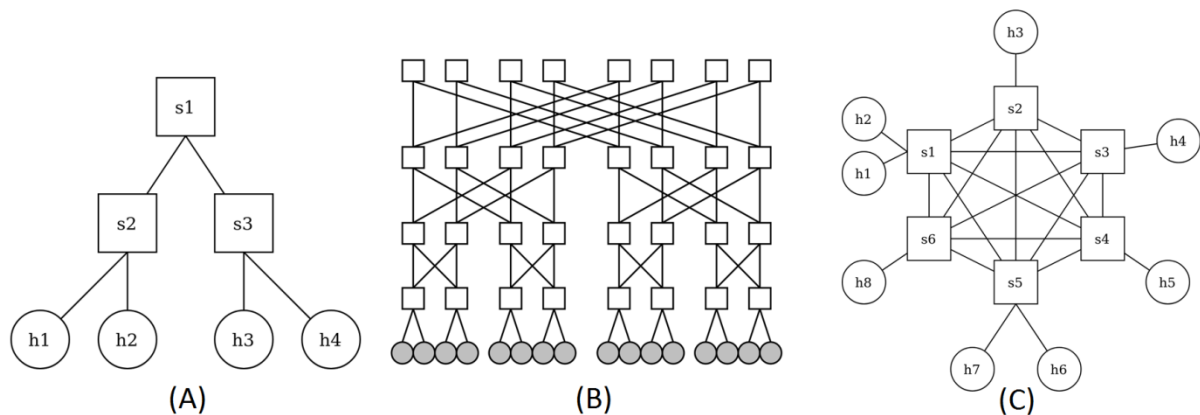


Figure 31. Examples of network topologies. (A) – tree topology. (B) – FatTree topology. (C) – Dragonfly topology.

Several topologies have been tested with the following common parameters:

- Number of traffic flows: 200 000
 - Number of switches: 40
 - Number of links: 100
- And the following various parameters:

- Links between switches
- Switches throughput
- Start time, end time and direction of each flow

2.4 Traffic generation

Mininet offers convenient functionality for simulating host activity. It is possible to execute console commands (ping, wget, curl etc.) by any host using a global terminal, as well as from a personal terminal of the host. Since the ability to use a global terminal is implemented, you can run custom scripts. All hosts also have shared access to files, which is very convenient for running scripts.

In order not to restrict traffic generation to simple commands, you can use traffic generation utilities. One of the most suitable utilities for generating dissimilar network traffic is D-ITG [11]. It implements the following features:

- Single-flow, Multi-flow, Daemon modes
- Detailed setting (generation time, delay, size / number of packets)
- Different types of protocols (UDP, TCP, ICMP, SCTP, DCCP)
- Traffic emulation

- Telnet - Telnet traffic emulator
- DNS - DNS traffic emulator
- Quake3 – Quake 3 traffic emulator
- Csa – Counter strike traffic emulator of active player
- Csi – Counter strike traffic emulator of inactive player
- VoIP – Voice-over-IP traffic emulator

Using this functionality, you can implement the generation of different traffic for each host.

2.5 Traffic capture and feature extraction

To solve the task of capturing traffic, one of the most popular utilities was used – Wireshark [12]. It provides the ability to monitor all network traffic passing through the host. Since the main idea is to monitor traffic at the controller level, this utility is deployed in a separate virtual machine along with the SDN-controller as shown on figure 2.

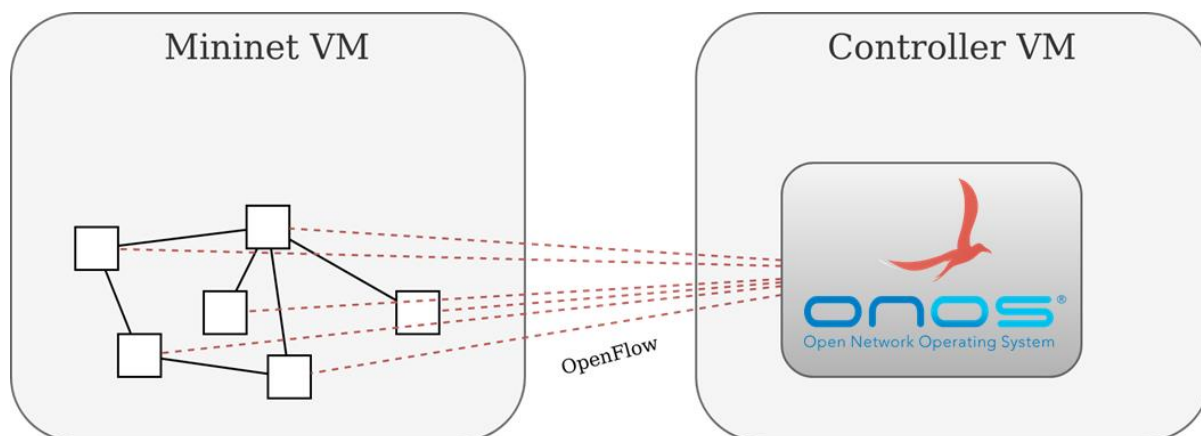


Figure 32. Example of system structure

Thus, the controller receives only information about the network using the OpenFlow protocol. Host-to-host packet transfer information remains inside the Mininet virtual machine. In this case, the OpenFlow 1.3 protocol was used to transfer information. Table 2 shows a list of features that can be captured. They contain key information about traffic flows for further analysis. Wireshark documentation contains a complete list of the parameters available for capturing [13].

Table 14. List of network traffic features captured using Wireshark tool

flow_removed.byte_count	flow_removed.cookie	flow_removed.duration_nsec	flow_removed.duration_sec
flow_removed.idle_timeout	flow_removed.packet_count	flow_removed.priority	flow_removed.reason
length	match.length	match.pad	match.type
oxm.field	oxm.hm	oxm.length	oxm.value_ethersaddr
oxm.value_uint32	type	version	Xid
flow_removed.hard_timeout	flow_removed.table_id	oxm.class	oxm.value_ethertype

3. Conclusion

As a result of this work, we have formed an approach to modeling software-defined networks to solve the problem of collecting data on traffic flows within the network. In the course of the research, the most convenient and compatible tools were identified. The presented set of tools is suitable for modeling within a single physical machine, as well as on desktop grid systems to create larger scale network models. However, for modeling large networks, it is worth using modeling tools with broader functionality (e. g., MaxiNet). Future work in this direction will be related to the development of a large-scale software-defined network and generating network traffic.

4. Acknowledgement

The reported study was funded by RFBR according to the research project No. 18-29-03264.

References

- [1] Casado, Martin, et al. "SANE: A Protection Architecture for Enterprise Networks." USENIX Security Symposium. Vol. 49. 2006.
- [2] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74
- [3] Mininet – An Instant Virtual Network on your Laptop – www.mininet.org (accessed 16 september 2021)
- [4] Ns-3 Network Simulator – <https://www.nsnam.org/> (accessed 16 september 2021)
- [5] OpenNet SDN emulator – <https://github.com/dlinknctu/OpenNet> (accessed 16 september 2021)
- [6] Containernet SDN emulator – <https://containernet.github.io> (accessed 16 september 2021)
- [7] Tinynet SDN emulator – <https://github.com/John-Lin/tinynet> (accessed 16 september 2021)
- [8] MaxiNet SDN emulator – <http://maxinet.github.io/> (accessed 16 september 2021)
- [9] Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions – <https://opennetworking.org/onos/> (accessed 16 september 2021)
- [10] OpenDaylight Documentation – <https://docs.opendaylight.org/en/latest/> (accessed 16 september 2021)
- [11] Distributed Internet Traffic Generator – <http://traffic.comics.unina.it/software/ITG/> (accessed 16 september 2021)
- [12] Wireshark – <https://www.wireshark.org/> (accessed 16 september 2021)
- [13] Wireshark. Display Filter Reference: OpenFlow 1.3 – https://www.wireshark.org/docs/dfref/o/openflow_v4.html (accessed 16 september 2021)