A POX Controller Module to Collect Web Traffic Statistics in SDN Environment

Wisam H. Muragaa, Kamaruzzaman Seman, Mohd Fadzli Marhusin

Abstract—Software Defined Networking (SDN) is a new norm of networks. It is designed to facilitate the way of managing, measuring, debugging and controlling the network dynamically, and to make it suitable for the modern applications. Generally, measurement methods can be divided into two categories: Active and passive methods. Active measurement method is employed to inject test packets into the network in order to monitor their behaviour (ping tool as an example). Meanwhile the passive measurement method is used to monitor the traffic for the purpose of deriving measurement values. The measurement methods, both active and passive, are useful for the collection of traffic statistics, and monitoring of the network traffic. Although there has been a work focusing on measuring traffic statistics in SDN environment, it was only meant for measuring packets and bytes rates for non-web traffic. In this study, a feasible method will be designed to measure the number of packets and bytes in a certain time, and facilitate obtaining statistics for both web traffic and non-web traffic. Web traffic refers to HTTP requests that use application layer; while non-web traffic refers to ICMP and TCP requests. Thus, this work is going to be more comprehensive than previous works. With a developed module on POX OpenFlow controller, information will be collected from each active flow in the OpenFlow switch, and presented on Command Line Interface (CLI) and wireshark interface. Obviously, statistics that will be displayed on CLI and on wireshark interfaces include type of protocol, number of bytes and number of packets, among others. Besides, this module will show the number of flows added to the switch whenever traffic is generated from and to hosts in the same statistics list. In order to carry out this work effectively, our Python module will send a statistics request message to the switch requesting its current ports and flows statistics in every five seconds; while the switch will reply with the required information in a message called statistics reply message. Thus, POX controller will be notified and updated with any changes could happen in the entire network in a very short time. Therefore, our aim of this study is to prepare a list for the important statistics elements that are collected from the whole network, to be used for any further researches; particularly, those that are dealing with the detection of the network attacks that cause a sudden rise in the number of packets and bytes like Distributed Denial of Service (DDoS).

Keywords—Mininet, OpenFlow, POX controller, SDN.

I. INTRODUCTION

SDN is a new technology designed to make our network more agile. Today's networks are often quite static, slow to change and dedicated for single services. With SDN one can create a network with more services in a dynamic fashion allowing us to consolidate multiple services onto one common infrastructure for both service providers and carriers.

The idea behind SDN is the idea of pulling the intelligence of the network away from the hardware. Decoupling the intelligence from the network hardware is achieved by separating the data plane from control plane as Open Networking Foundation (ONF) has defined [1]. Data plane or infrastructure layer is the bottom layer in SDN structure and that is where the network forwarding equipment is situated. The control layer which is the middle layer in SDN architecture is responsible for configuring the infrastructure layer; it does that by receiving a service request from the third layer which is the application layer. SDN controller is the intelligence of SDN structure; unlike the traditional networks, where the intelligence is the Network Operating System (NOS). OpenFlow controllers in SDN are logically located in control plane, which is the middle layer meant to control and manage the requests of top layer (application layer), and to instruct the bottom layer (infrastructure layer). There are several types of SDN controllers; it can be mainly categorized based on the programming language of which the controller is made and the purpose of innovation [2]. One of it is the chosen POX controller.

Communications between controller and both upper and lower layers in SDN structure are done by northbound and southbound Application Programming Interfaces (APIs). Northbound APIs are used to communicate between controller and the running applications over the network; while southbound APIs are responsible for the communications between controller and the packet forwarding hardware. The API presence enables the network to be directly programmable, and it gives the software developers the opportunity to develop their own APIs and applications and implements it instantly [2].

In SDN terminology, communication that occurs between control layer and forwarding layer is called southbound communication. The standard mechanism that allows them to communicate with each other is known as OpenFlow protocol. OpenFlow protocol allows the controller to have a direct access to and manipulation of the forwarding devices of data plane. OpenFlow has different versions; the one that will be used in this study is OpenFlow 1.1.0, where POX controller currently supports this version [2]-[3].

As a related work, the authors in [4] concentrate on measuring non-web traffic statistics in SDN environment. Their method collects port and flow statistics from the running switch(s) and presents these statistics on a web interface called weathermap. Their method focuses on the rate of the packets

Wisam H. Muragaa and Mohd Fadzli Marhusin are with Faculty of Science and Technology, Universiti Sains Islam Malaysia, Bandar Baru Nilai, 71800, Nilai, Malaysia (e-mail: phd.wisam@gmail.com, fadzli@usim.edu.my).

Kamaruzzaman Seman is with Faculty of Engineering and Build Environment, Universiti Sains Islam Malaysia, Bandar Baru Nilai, 71800, Nilai, Malaysia (e-mail: drkzaman@usim.edu.my).

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:12, 2016

and bytes. Some of port statistics collected from their study include received and transmitted packets rate, received and transmitted bit rate and received and transmitted dropped packets among others; while the collected main flow statistics include byte and packet rates, action, duration per second, hard timeout and some other statistics. In fact, these statistics are too many; as they are not meant for any specific purpose.

In this study, we have prepared an accurate and concise list of most important statistics elements to be used for further researches; particularly, those that are related to networking traffic disruption.

II. MEASURING IN SDN

Every measurement technique in the traditional networks requires the separation of hardware installation or software configuration. It makes the implementation of this technique a tedious and expensive task. Meanwhile, OpenFlow networks provide necessary interfaces to implement most of the traditional measuring methods at a lower cost and higher efficiency [5]. Statistics collection in this study is going to be achieved by using two OpenFlow protocol messages described in the OpenFlow specifications [3]. The two OpenFlow messages are statistics request message and statistics reply message. Statistics request message is a message sent from the controller to the switch requesting its current ports and flow statistics. While the statistics reply message is the message sent back from the switch to the controller in reply to its request.

Network measurements are often performed on application layer or network layer [6]. Application layer measurements are designed to measure the application performance. Network layer measurements are meant to use infrastructure forwarding components (such as switches and routers) [7].

The aim of this study is to get actual network traffic information that can be used in verifying any variation that might happen in the network performance.

III. DESIGN AND DEVELOPMENT

In this paper, we have created a Python module to work on POX OpenFlow controller, and it is specifically designed to collect web traffic statistics as well as non-web traffic statistics to be a more comprehensive work. This study is aimed to extract specific types of statistics from the generated traffic in the network using POX controller. The designing and the development of this module are described in the next two paragraphs.

A. Design

Due to the design of OpenFlow, flow entries are stored in the switch flow tables and then forwarded to their destinations based on the instructions received from the OpenFlow controller. One of the switch flow table entries is counters entry. Counters entry contains packet counter and byte counter. All received packets and bytes in SDN are stored in these two counters. Our module is designed to extract the actual numbers stored in these two counters, and it is used to give the numbers of flows that contain these counters

repeatedly and in a very short time.

B. Development

In order to create the presented design, Python programming language is used. The reason behind using Python is that POX OpenFlow controller is a Python-based. In this module, a statistics collecting function is implemented on POX controller in order to query the packet and byte counters periodically. The numbers accumulated in these counters, as well as the number of flows, will be displayed on the CLI in a short period of time. The interval time between each appearance and the next for the statistics is five seconds. Wireshark built-in tool is used to analyze the traffic in order to show a depth detailed list. This list shows the type of protocol used to generate the traffic, source IP address, destination IP address and some extra information about segmenting. Wireshark and CLI are used simultaneously. Mainly, this POX controller module is developed to measure web traffic statistics; but at the same time, it can be used to measure non-web traffic as well. This module concentrates on measuring HTTP requests that use the application layer in SDN. TCP and ICMP requests that use network layer can be measured by using this module as well. Thus, our POX controller module can be considered as a multi-functional module.

IV. EXPERIMENT AND RESULTS

Capable network topologies in SDN can be achieved by Mininet. Mininet [9] is network emulation software that allows the creation of a realistic virtual network and running end-hosts, switches, routers and application codes on a single Linux kernel. Controllers, switches, and hosts all can be launched by Mininet in seconds with a single command on a virtual machine known as Mininet VM.

We use Mininet CLI to create our topology in the SDN environment. The topology used in this experiment contains one OpenFlow controller, one OpenFlow switch and four end-hosts as presented in Fig. 1.

The OpenFlow controller used in this experiment is POX controller. POX is originated from NOX where NOX is the original OpenFlow controller. The POX repository has multiple branches. And for this experiment, we use POX (dart) branch. After the network topology is created, the POX controller must be run on another terminal. As shown in Fig. 2, when POX controller works, the OpenFlow switch is connected to the network directly to be ready to receive the instructions or the modification messages.

Our module is meant to be run along with the forwarding.l2_learing component.

Reading state messages in SDN structure is done by using two messages: OFPT_STATS_REQUEST message and OFPT_STATS_REPLY message [3]. During the running of the system, the datapath may be queried about its current state using the OFPT_STATS_REQUEST message; while the switch responds with one or more OFPT_STATS_REPLY messages. In both the request and response messages, the type field specifies the kind of information being passed and determines how the body field is being interpreted.

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
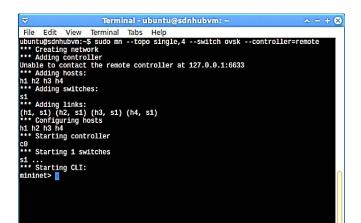Vol:10, No:12, 2016

Fig. 1 Topology used in the experiment

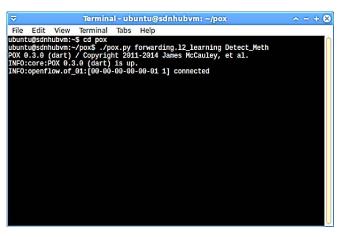(a) Sending HTTP requests from host 1

Fig. 2 Switch connected and ready to receive instructions

In order to get switch statistics in SDN, OpenFlow provides port level functionalities. Using its mechanism, the OpenFlow switch generates the response message "OFPortStatsReply" when it receives an "OFPortStatsRequest" message requesting its current statistics. By using these two messages, more details about packets that are generated in the network can be collected. "OFPortStatsRequest" message requests port statistics either for a single port by specifying the exact port number in port_no field or for all switch ports by typing "OFPP_ANY" in the same field [3]. By querying the ports iteratively, the controller can be notified with the updated numbers of packets and bytes that are stored in the statistics counters after each using of the switch ports. Even more, port statistics can give more information about both (transmit and receive state) such as a number of dropped packets and bytes, errors, and collisions which can be used to measure the path loss rate in the network [8]. The body of the reply is consisting of number of received packets, number of transmitted packets, number of received bytes and number of transmitted bytes [3].

Flow statistics in SDN can be either requested for individual flows or multiple flows. Information about individual flows is requested with the OFPST_FLOW stats request "ofp_flow_stats_request". The body of the reply is partly consisting of the number of packets and bytes in each individual flow [3].

(b) Received HTTP requests on host 2

Fig. 3 Web traffic generated between host 1 and host 2

In OpenFlow, we use aggregate flow statistics (request and reply) messages for querying the number of flows in the OpenFlow switch table. Information about multiple flows is requested with the OFPST_AGGREGATE stats request "ofp_aggregate_stats_request". The body of the reply consists

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:12, 2016

of the number of flows. The number of packets and bytes in all flows are mentioned in the same reply message as well [3].

From the Mininet VM terminal where our network topology was created, we open four terminals for host 1, host 2, host 3 and host 4. Host 1 works as a web server to generate HTTP traffic to the host 2 that works as a receiver to this web traffic as shown in Figs. 3 (a) and (b).

As presented in Fig. 4, our POX controller module begins sending its requests to the switch to collect statistics every five seconds iteratively. If there is any traffic generated between hosts in the network, the module can detect this traffic and list the statistics obtained from the switch on the CLI. If the packets generated between hosts are increased or decreased, our POX module can show that clearly on its list. We notice that there is no traffic in the network if the module shows the list with empty fields as manifested in Fig. 4. Empty fields mean the fields carrying no numbers.

Fig. 4 Statistics obtained by the POX controller module

Fig. 5 Details about sending and receiving HTTP requests and TCP segments in between

While POX module is collecting the statistics from the switch, the traffic generated between host 1 and host 2 is analyzed by wireshark software. Each sending and receiving of HTTP request is displayed in details on the wireshark interface. Fig. 5 shows that the traffic is divided into TCP segments after sending the HTTP request, and it is reassembled again at the receiving side to create a text/html file. Moreover, our POX module has the capability to measure ICMP and TCP traffic, as well as HTTP traffic simultaneously. Therefore, our module is more comprehensive in terms of giving more statistics about the traffic generated in the network. And it also displays it on the CLI and wireshark interfaces. While the HTTP traffic is being generated between host 1 and host 2, we run a small Python program to generate ICMP traffic from host 3 to host 4. Running ICMP Python generator on host 3 is presented in Fig. 6.

To approve that our module can measure traffic generated by different protocols at the same time, one must see what is displayed on wireshark interface as shown in Fig. 7. The screen shot shows a number of ICMP requests and replies, a number of OpenFlow stats and reply messages, a number of OpenFlow instructions and modification messages and HTTP requests with TCP segments.
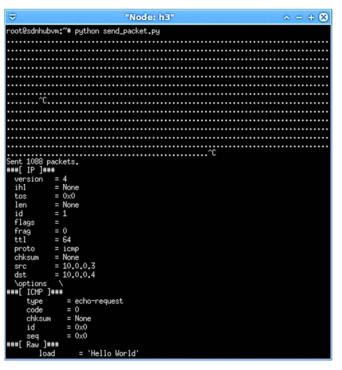
World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:12, 2016

Fig. 6 Details of sending ICMP requests from host 3 to host 4 using a
Python program created to achieve the purpose

V. CONCLUSION

In this paper, an attempt is made to produce a feasible method for measuring statistics of both web and non-web traffic. This study is aimed to provide the developers, who deal with the detection and mitigation of network attacks, with an accurate statistics list. It is a concise list of the most important statistics in the network (packets and bytes). This list can be used as an indicator of any sudden or unjustified rise in the network progress. By keeping the network controller updated with any changes that might happen in the network, any network developer or network administrator can intervene, in order to find a solution to any problem that tends to arise in the network.

Fig. 7 Measuring statistics for traffic generated by different network protocols

REFERENCES

[1] Open Networking Foundation "Software Defined Networks: *The new Norm of Networks*" White paper 2012 Available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf
[2] POX controller. Available: http://www.noxrepo.org/pox/about-pox/
[3] Pfaff, B. (2011). OpenFlow Switch Specification Version 1.1. 0 Implemented (Wire Protocol 0x02). URL: http://www. openflow. org/documents/openflow-spec-v1, 1.
[4] Hsu, C. Y., Tsai, P. W., Chou, H. Y., Luo, M. Y., & Yang, C. S. (2014). A Flow-based Method to Measure Traffic Statistics in Software Defined Network. Proceedings of the Asia-Pacific Advanced Network, 38, 19-22.
[5] Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., & Madhyastha, H. V. (2013, March). Flowsense: Monitoring network utilization with zero measurement cost. In International Conference on Passive and Active Network Measurement (pp. 31-41). Springer Berlin Heidelberg.
[6] Yassine, A., Rahimi, H., & Shirmohammadi, S. (2015). Software defined network traffic measurement: Current trends and challenges. IEEE Instrumentation & Measurement Magazine, 18(2), 42-50.
[7] Mohan, V., Reddy, Y. J., & Kalpana, K. (2011). Active and passive network measurements: a survey. International Journal of Computer Science and Information Technologies, 2(4), 1372-1385. J.

[8] Hu, F. (Ed.). (2014). Network Innovation through OpenFlow and SDN: Principles and Design. CRC Press.

[9] Mininet. Available: http://mininet.org/