2016 IEEE 27th International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC): Services Applications and Business

Real-time monitoring of SDN networks using non-invasive cloud-based logging platforms

Bartlomiej Siniarski* Cristian Olariu*, Philip Perry*, Trevor Parsons§ and John Murphy*
*UCD, School of Computer Science and Informatics, Belfield, Dublin 4, Ireland
§ Logentries, Rapid 7, 26-28 Lombard Street, Dublin 2, Dublin, Ireland
bartlomiej.siniarski@ucdconnect.ie, cristian.olariu@ucd.ie, philip.perry@ucd.ie
trevor.parsons@logentries.com, j.murphy@ucd.ie

***Abstract*—The Software Defined Networking (SDN) paradigm enables quick deployment of software controlled network infrastructures, however new approaches to system monitoring are required to provide network administrators with instant feedback on a network's health. This paper details the deployment of an SDN system architecture featuring the integration of a cloud-based, real-time log-analysis platform. The proposed architecture uses log data collected from host machines, OpenFlow switches and the SDN controllers in a non-invasive style. This work uses a commercially available correlation platform to provide network administrators with a real-time view of the network status and the approach is validated under two scenarios:- network overload and a security attack.**

I. Introduction

The use of wireless networks in enterprise deployments is becoming increasingly important to the functions of enterprises as the use of Bring-Your-Own-Device (BYOD) services proliferates and business functionality is increasingly pushed onto employees smart phones. It is therefore interesting to explore ways that can ease the burden of managing such a network without the need to employ a networking expert with deep knowledge of the peculiarities of the interaction between such networks and the heterogeneous traffic that they must carry.

The Software Defined Networking (SDN) paradigm allows the centralisation of network intelligence in an SDN Controller Node. The general task of a controller is to identify data flows in a network and manage network behaviour for each flow, using protocols, such as OpenFlow. This approach allows the network to behave differently for each flow for each user of the network which yields both a tailored service and efficient resource utilisation. This approach can allow the efficient deployment and management of wired and wireless systems within a range of enterprise scenarios where deep knowledge of the network performance is often outsourced. However, the dependence on a small number of software nodes opens the possibility of performance bottlenecks, single point of failure and security attacks. In this work we focus on the performance issues that can arise and use these as an indicator of other undesirable activity.

Like most software systems, the SDN controller is typically equipped with logging features that can be used at run time to provide insight into the behaviour of the software and the hardware that it is running on. The amount of logging depends on developer style, however typically it will log the occurrence of major SDN functions such as flow table modification, network component connection status and topology characteristics. The SDN controller log files contain valuable raw information about the network events processed by the software modules, however those logs do not reflect the overall performance and state of the network.

In this paper we combine the information gathered from the SDN controller and individual network components in order to view a wide range of event types, proactively monitor the system performance, build real-time alerts and notifications, identify anomalies and allow the administrator to perform real-time health monitoring of the SDN network without the need for an in-house networking expert. We also gather information from the host machines to ensure that the compute platforms hosting the software functions are not a limiting factor the the system performance. All collected data is processed in real-time in a non-invasive style, having negligible impact on the overall system performance. In this work we use a cloud-based log analysis tool to analyse logs from SDN controllers and switches. The tool used here is developed by Logentries which grew out from an anti-pattern research work [1] and later generalised as a cloud-based real-time logging tool aimed at enterprise software systems. Applying the tool to SDN is a novel use of the tool, as it requires extensive knowledge of logging, log analysis, network behaviour and SDN domain-specific knowledge.

The biggest advantage of real-time logging platforms is that the log data is analysed and processed as it is streamed to the cloud server. Unlike traditional approaches, real-time logging systems provide insight within seconds and enable immediate notifications, alerts and visualisation of events. This approach means that the enterprise network administrator does not need to develop a bespoke analytics platform to observe the impact of traffic variation that can be particularly troublesome in the wireless segments of the network.

This paper is structured as follows: Section III describes the proposed system architecture and provides the reader with a detailed description of system components. Section IV is an overview of tools and techniques used to simulate the SDN network environment used for testing. The results for two test scenarios are presented in section V and contain observations and findings. This paper presents the results in section V

978-1-5090-3254-9/16/$31.00 ©2016 IEEE

followed by conclusions.

II. Related works

Background research of previous work in the area of SDN network performance measurement revealed attempts [2] to overload the SDN controllers, using Cbench [3] and hcprobe [4], however simulated network components including imitations of switches, do not provide realistic results reflecting the limits of the SDN controller, they also do not provide us with a realistic view of the SDN network testing environment. The evaluation of network emulator capabilities supported by the results from [5] in the field of network emulation was carried out in order to estimate network limits and approve the set up scenarios for the experiments.

The author of another work [6] based around network health monitoring made a valid point claiming that network health monitoring against network failures, congestions, misconfigurations, and security attacks should be an integral part of creating dependable network services. Network management system, then, should be able to track a detected problem's root cause in real time in order to remove the problem within the contained network space.

Further research in the area of SDN network health monitoring revealed, there are very few tools to meet a requirements of network monitoring in such a wide scope, especially in the field of SDN/NFV. Log event analysis in SDN networks is a novel approach to tackle the problem of network device failure, congestion detection and other issues that may arrise, while running a virtualised network such as misconfiguration or security attacks. In this work we use a cloud-based real-time log management tool from Logentries [7] to analyse logs from the SDN controller, host machines and OpenFlow switches. Logentries is a leading log management service among other services such as, Splunk [8], Logstash [9], able to dynamically scale and adapt to data volume or infrastructure.

Continuation of research in the field of SDN networking, especially in the field of SDN security [10] confirmed that new methods and techniques must be considered and explored in order to enable the dynamic configurations in SDN security monitoring, detection, prevention and recovery capabilities. Another extensive SDN security survey [11] by the same author claimed that logging the network events can be valuable to network operations and can improve the security and reliability of the infrastructure.

III. System architecture

In our proposed system architecture depicted in Figure 1, all log events produced by a controller or a set of controllers are collected by a log management agent and periodically pushed to the log management and analysis platform deployed in the cloud. The same agent monitors the performance of the server running the SDN controller[s] and periodically pushes data to the log management platform. The server logs include performance metrics such as CPU usage, memory usage, statistics per network interface, and disk statistics.

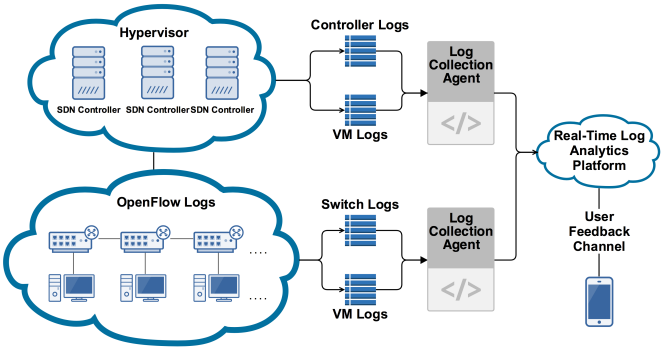

Fig. 1. Proposed system architecture

Similarly to the Control Plane, all logs produced by a single machine or multiple machines running network components in the infrastructure layer are collected by log management agents and pushed to the log management platform for real-time analysis via TCP connections. Network administrators are now able to view, correlate and analyse SDN network logs in real-time or refer to historical data. To assure the reliability of the monitoring data, TCP should be used to transport log messages to the analysis platform.

The log management agent is a program that automates data collection from the log files and forwards the collected data to the log management platform within seconds. Thus, the log file contents must be read as the events happen and streamed to the log analysis platform.

A. Log Event Format

The log analysis platform must be able to ingest any type of log data produced by virtualised network architecture. Most of the well known logging platforms provide agents that will help simplify and automate log collection in users environment. Ideally the log files are in a structured format, e.g. JSON, however a logging platform should also be able to process unstructured log files.

The format of an SDN controller's log files may vary depending on the programming language used, debugging strategy and the programmers' style. The log management platform should be able to understand the Key-Value-Pair format, however the platform should include the support for Regular Expressions, such as in the work of Yu et al. [12], in order to extract values from unstructured log data or scan packet content if necessary.

B. Feedback Channel

The User Feedback Channel in Fig 1, is built around notification capabilities of the log analytic platform which can be used to: i) notify the network administrator about all potentially critical events that may affect the overall performance of the network, ii) possible network attacks and the context of the malicious behaviour, and iii) report the overall health of the network.

IV. Experimental Setup

Experiments are carried out in a virtualised environment depicted in Figure 2 and is able to host the SDN network and the SDN controller. The machine used for testing is a Dell T5500 server which has a 4-core, Intel Xeon E5630 CPU, running at 2.53 GHz with hyper-threading enabled, 24 GB DDR3 memory running at 1600 MHz. The server is running Lubuntu 14.01 with XEN as the virtualisation hypervisor. The SDN network is emulated inside a virtual machine that runs on 2 cores (4 threads in total) and 12 GB of RAM. The control plane virtual machine runs on 1 core (2 threads) and 10 GB of RAM. The virtual machines are connected using Open vSwitch. Internet access is granted to both Fully-Virtualized VMs in order to communicate with the logging platform.

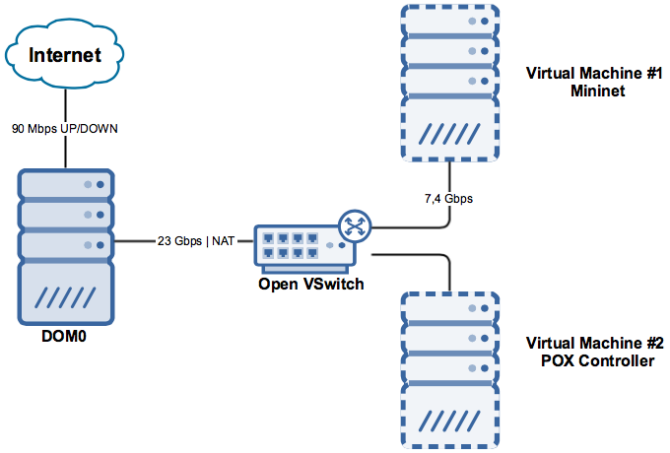

Fig. 2. Experimental setup

In order to minimise the uncertainty of non-realistic network environments we have developed the SDN Network Test Manager to run a series of tests. We used an instance of a SDN controller running a Layer 2 forwarding application that is connected to a Mininet [13] network. The emulated environment in Mininet was used here to provide a well controlled but realistic test bed. Mininet is used to create SDN network topologies that consist of a number of OpenFlow-enabled switches [14] that serve network traffic between various hosts running a controllable traffic source. Tests are executed on i) linear topology, where the number of hosts/switches is between 2-48, ii) tree topology, where the depth is between 2-8, iii) single topology, where the number of hosts is between 2-128. The emulated SDN network is connected to a virtual machine instance of the POX [15] controller.

A. Test Manager

The test manager is a collection of Python scripts developed to simulate numerous network scenarios and network traffic is generated using the Scapy [16] library. The user testing the environment is able to choose from the following options:

1) Stress the network or/and the SDN controller with predefined traffic profiles by increasing or decreasing the network load and/or SDN controller load at any given stage. This is done by sending additional simultaneous packets to through switches or SDN controller.
2) Choose from a range of packet types, such as ARP, DHCP, DNS, ICMP, IPv4, LLDP, TCP, UDP.
3) Security attacks and their severity: MAC flooding, DHCP attack, malformed packets flooding, Ping-of-death, ARP cache poisoning.

B. Log Events

In this paper we recommend a list of log events that should be considered in monitoring the SDN network. All of those events were considered in experiments section.

SDN Controller logs: SDN controller is the ”brain” of the SDN network, hence it can produce valuable log events in terms of network monitoring, such as:

- Controller going up/down/went down - messages sent by the controller upon a start up, shut down or restart.
- Echo request/reply - communication messages sent periodically, usually every 5-20 seconds, between the controller and all discovered switches on the SDN network in order to keep the connection live.
- Link timeout and new link detected - these events are triggered upon link creation or link failure. Link events can be correlated with switch logs that consist of more detailed information. It is important to note that the log management system should provide the user with the context of log event.
- New flow, new packet in, packet type, malformed packet - one of the duties of the controller is to investigate the arriving packets and validate their structure. The controller outputs the type of packet that arrives at the controller port. Some of these messages are generated from switch messages and validated by the SDN controller.
- New switch discovered, switch disconnected, switch connection closed - similarly to Link events, switch activity on the network is reported by the SDN controller as well as stored in switch logs for further analysis.
- Barrier request/reply - those messages are sent between the controller and virtual switches. This ensures that the switch processes all message dependencies and sends all notifications for completed operations before proceeding with new requests. When the switch completes an operation, it sends a reply message back to the controller.

Host Server logs: These statistics are collected from the computing platform (e.g. virtual machine running the SDN controller) by the log management agent. The agent in this work collects platform metrics such as CPU and RAM usage.

- Current per core CPU usage including total user/system usage and percentage of idle time - plotting and tagging critical events related to CPU usage help us to understand the state of the network and the controller.
- Current memory statistics including available, used, free, active, inactive memory, buffer and cache stats - memory in switches is utilised by various switch software components such as storing data structures and SDN flow

tables. For example a switching system using a single routing protocol with 200,000 routes learned, consumes around 492 MB of memory [17].
- Network interface statistics: current network statistics for each network interface including bytes sent, bytes received, packets sent, packets received, error in, error out, drop in and drop out. The log event manager should be able to pick up those events per network interface and use this data to report the traffic load on each component.

OpenFlow Switch logs: OpenFlow switch logs include detailed activity of each SDN switch. Those logs can be correlated with SDN controller logs to report unusual or critical behaviour in the data plane.

- Connecting switch to the controller - is a standard message containing detailed connection information about a joining SDN switch.
- Connection closed by the SDN controller - in most of the cases, this event is logged when the user intentionally disconnects the controller from the network.
- Dropping packet_in messages - drop of multiple control messages due to queue overflow on the switch. This can be considered as a critical event and the network administrator should be alerted by the log management platform to investigate this further.
- Added interface on bridge - is a standard event logged by the OpenFlow switch containing details of a newly enabled port.
- Percentage CPU usage by individual switch - indicates the amount of resources used by the CPU of the switch to run the switch's operating system.
- Timeout events - are warning messages logged by OpenFlow switch indicating a connection timeout. Those events should be followed by wake-up events reported by the switch, however if wake-up events are not discovered, the network administrator should be able to correlate the number of timeout events and the number of echo replies reported by the controller in order to confirm the definite timeout of the switch.

C. SDN Network Health Ranking

The tests presented in this work and a number of side experiments are used to propose a preliminary SDN Network health ranking scale based on symptoms gathered from log intelligence.

Level 5 - No critical events observed, CPU usage and Memory usage on both controller and network server are below a defined threshold (50% in this case).

Level 4 - The CPU usage of the controller host or network host is above 50%.

Level 3 - Controller went down, however it recovered and a "controller up" event was discovered. Switch connection was lost or link failure events were reported.

Level 2 - The CPU usage of the controller or most of the network components is above the threshold of 70%. Open vSwitches start to complain about packet drops.

Level 1 - Log management platform reported "controller down" events, but did not report "controller up" events. Log inactivity alerts enabled on the platform send warning messages using the user feedback channel. Controller does not send discovery packets.

V. Results

In this work we consider two SDN scenarios to demonstrate the functionality of our proposed architecture. Initially, the controller is kept in an idle state for a period of time. The SDN network emulated in Mininet is then connected to the SDN controller and lightly loaded for a period of 10-20 minutes in order to create a clear baseline of standard network behaviour. After the warm-up period, test scenarios are triggered by the test manager.

A. Scenario 1: Real-time switch activity monitoring

The cloud-base log analysis engine is used to extract the values from the SDN controller logs and provide insight into some of the processes in the network. Further grouping of events allows us to monitor the workload per network component, in our case - each OpenFlow switch. A linear topology of 5 switches and 1 host per switch is constructed for this experiment. The switch identifier is visible in the legend of Figure 3 which depicts flows per second for each switch.

The test manager is instructed to start the simulation of low (5 flows/second), medium (100 flows/second) and high (1000 flows/second) network traffic and to present the workload per network switch. Sender and receiver nodes are located at least 1 switch apart from each other, in order to make sure that the measurements for the intermediate switches are correctly recorded. The results presented in Figure 3 show that it is possible to obtain accurate switch workload measurements using the extraction and analysis of the SDN controller logs.

B. Scenario 2: DHCP Starvation Attack

In this experiment DHCP packets are injected at the highest possible rate for the duration of 60 minutes to prevent hosts from gaining access to the network by denying them acquiring an IP address from a DHCP server, or simulating a DOS attack on a DHCP server. Our main goal is to effectively correlate log events in real-time and evaluate the nature of the attack triggered by a simulated malicious set of users.

The results for this scenario are presented on 3 graphs (See Figures 4,5,6) that were generated in real-time. Graphs presented are divided into four intervals:

Interval A is a representation of a baseline network behaviour. In this work we decided to instruct the controller to install 10 flows on all the switches on every second. This activity is not very costly as the controller consumes on average 6% of available CPU resources on the controller host and 22% of the host machine running network components. These low levels of usage indicate that the system performance is not being limited by the specific deployment scenario and give confidence that the results are not being impacted by hardware limitations. In this specific interval we do not observe any packet drops complaints by the switches. Network status is

ranked at level 5, which corresponds to an excellent network health.

Interval B shows the period in which the Test Manager carried out a medium severity DHCP attack. In this work we use Logentries tagging features to mark all the events listed in "Log Events" section of this paper. The alerting system in the Logentries manager reports unusual activity within the first 5 seconds of the attack and plots the average number of DHCP packets in Figure 4. At the same time, all of the 5 available Open vSwitches report in Figure 5 packet dropping events during the duration of the test. The CPU usage increases by 64% on the controller host and it is now using 100% of the CPU resources on the Mininet host. It is important to note that we observe Open vSwitches reporting very high CPU usage, however the controller logs do not report this unusual activity. This is a great example of a remote controller failing to detect new flow attack described in another work[6]. For example, when a switch's CPU is already saturated by an instantaneous new flow attack, the actual outgoing number of new flow requests (i.e. 100 pps) sent to controller can be far less than the real incoming packet counts (i.e. 4000 pps). It results a false positive decision that can be detected by proposed system architecture. Network health status drops down to level 2.

Interval C is a representation of the Test Manager reducing the number of DHCP packets sent per second by 50% for the duration of 10 minutes and then increasing the number of DHCP packets sent per second. This short intermediate test is invoked in order to observe the activity reported by Open vSwitches, which drop less amount of packets than previously. Network performance improves, hence network health is at level 4. Despite the fact that the CPU usage is still about 50%, there are no critical alerts.

Interval D is a representation of the test manger stopping the DHCP attack and putting the network back to baseline traffic load. Network health is ranked at level 5, which is an indicator of a good health of an SDN network. This test is invoked in order to make sure that our readings and analysis of logs are correct.

It is also important to mention that the communication between hosts in the network at the time of the attack was lost, however the controller and switches recovered after completing some of the tasks buffered during the attack.

VI. Conclusion

This work has shown the usefulness of cloud-based log analysis engines for the evaluation of SDN networks performance for enterprise networks that are managed by a non-expert using an off the shelf tool. The experiments presented here have demonstrated how to monitor the network health and successfully identify when the system is being overloaded, identify malicious attacks and identify if the system has recovered successfully from an attack. We propose a list of logs for the SDN controller that can be used as a reference by network administrators and adapted, regardless on a network topology or architecture. This preliminary work will be extended by enlarging the knowledge base of log events from system components and creating the Controller Feedback Channel and create a policy language that can be used in network environments using heterogeneous SDN controllers.

VII. Future Work

In this work we envisage extending from a simple alert-based feedback to a human to an autonomous Controller Feedback Channel (CFC) as part of the architecture. This would utilise the intelligence gathered by log management system, and send high-priority instructions to the SDN controller based on previously defined policies [18] in order to e.g. avoid the single point of failure situation, recover from the malicious activity, or stop unwanted behaviour on the network if the controller is unable to do so.

Acknowledgment

Supported, in part, by Science Foundation Ireland grant 10/CE/I1855 and by Science Foundation Ireland grant 13/RC/2094 and by Enterprise Ireland Grant IP20140344.

References

[1] M. Wang, V. Holub, T. Parsons, J. Murphy, and P. O'Sullivan, "Scalable run-time correlation engine for monitoring in a cloud computing environment," in *Engineering of Computer Based Systems (ECBS), 2010 17th IEEE International Conference and Workshops on*. IEEE, 2010, pp. 29–38.

[2] A. Shalimov, D. Zimarina, and V. Pashkov, "Advanced Study of SDN / OpenFlow controllers." Computer-Communication Networks.

[3] GPLv2, "Cbench Benchmarking Tool." [Online]. Available: https://github.com/andi-bigswitch/oflops/tree/master/cbench

[4] OpenSource, "Hcprobe Benchmarking Tool." [Online]. Available: https://github.com/ARCCN/hcprobe

[5] C. Thorpe, C. Olariu, A. Hava, and P. McDonagh, "Experience of developing an openflow SDN prototype for managing IPTV networks," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 966–971.

[6] S. Song, "Improving network health monitoring accuracy based on data fusion for software defined networking," in *Future Information Technology*. Springer, 2014, pp. 469–472.

[7] R. Ltd., "Logentries." [Online]. Available: http://www.logentries.com

[8] Splunk, "Splunk Log Analytic Tool." [Online]. Available: http://www.splunk.com

[9] Logstash, "Logstash Log Analytic Tool." [Online]. Available: https://www.elastic.co/products/logstash

[10] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "Sdn security: A survey," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For*. IEEE, 2013, pp. 1–7.

[11] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks."

[12] F. Yu, Z. Chen, Y. Diao, T. Lakshman, and R. Katz, "Fast and memory-efficient regular expression matching for deep packet inspection," *2006 Symposium on Architecture For Networking And Communications Systems*, 2006.

[13] OpenSource, "Mininet Network Emulator." [Online]. Available: http://mininet.org

[14] OpenFlow, "he OpenFlow Switch Specification." [Online]. Available: http://OpenFlowSwitch.org.

[15] OpenSource, "POX Controller." [Online]. Available: http://www.noxrepo.org/pox/about-pox

[16] G. Philippe Biondi, "Scapy - packet manipulation tool." [Online]. Available: http://www.secdev.org/projects/scapy/

[17] Cisco, "Memory Utilization on Cisco Catalyst 4500 Series Switches," Cisco, Tech. Rep., 2011. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-4500-series-switches/white_paper_c27-554637.pdf

[18] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

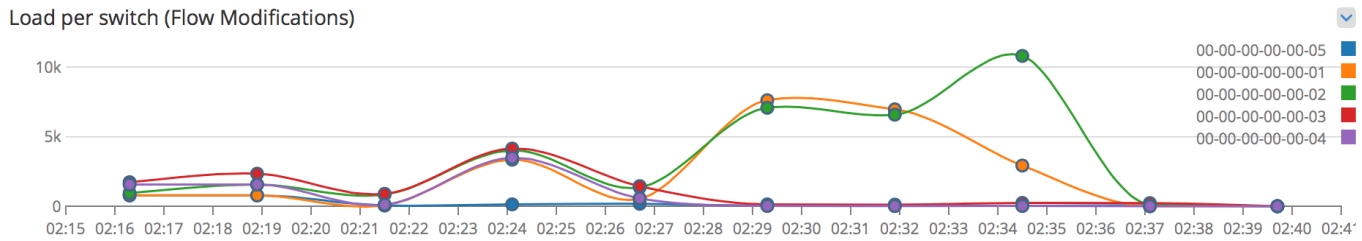

Fig. 3. Number of flows installed per switch (Source of data: POX controller logs)

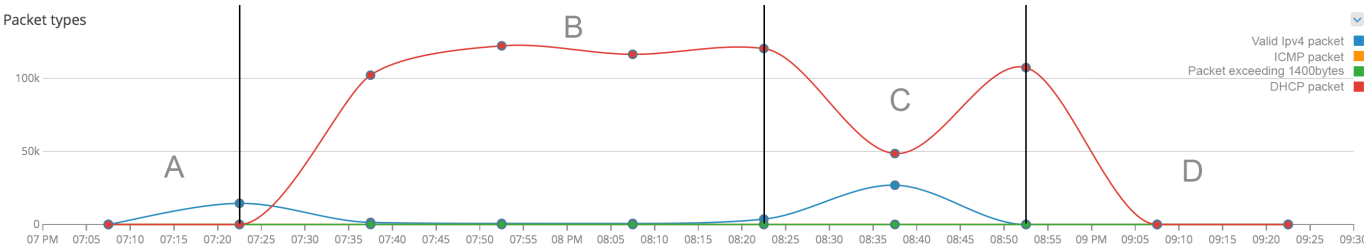

Fig. 4. Network packets grouped by packet category (Source of data: POX controller logs)

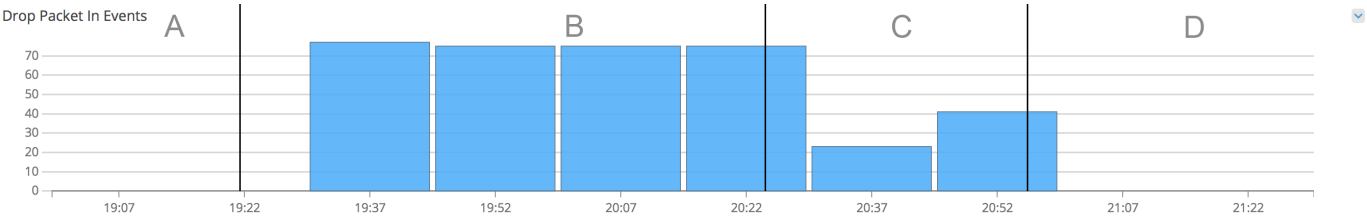

Fig. 5. Drop Packet In events collected from the network switches logs. (Source of data: Open vSwitch logs)

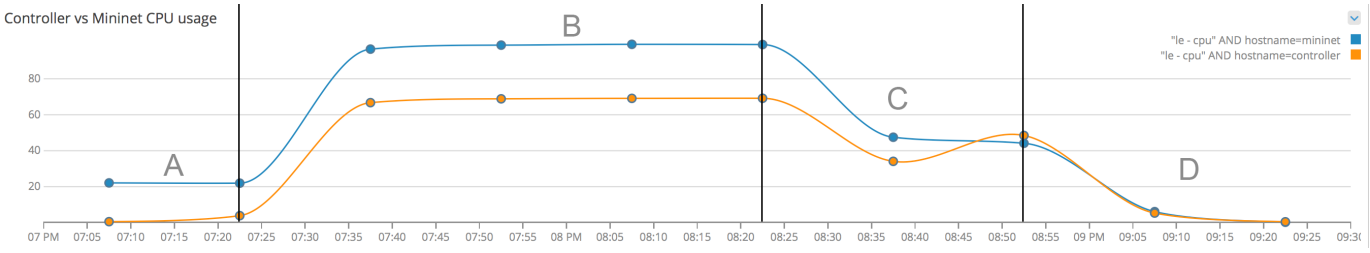

Fig. 6. Average CPU usage. (Source of data: Log events collected from servers hosting POX controller and Mininet)