Automatic Detection of Elephant flows through Openflow-based OpenvSwitch

Research Project
MSc Cloud Computing

Spurthi Mallesh
x15047539

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet – 2016/2017
School of Computing

Student Name:	Spurthi Mallesh
Student ID:	x15047539
Programme:	MSc Cloud Computing
Year:	2017
Module:	Research Project
Lecturer:	Vikas Sahni
Submission Due Date:	16/08/2017
Project Title:	Automatic detection of elephant flows through openflow-based openvswitch
Word Count:	5646

I hereby certify that the information contained in this (Automatic detection of elephant flows through openflow-based openvswitch) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	15th August 2017

PLEASE READ THE FOLLOWING INSTRUCTIONS:
1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Automatic Detection of Elephant flows through Openflow-based OpenvSwitch

Spurthi Mallesh
x15047539
MSc Research Project in Cloud Computing

15th August 2017

Abstract

Software Defined Networking (SDN) has proved its importance as it allows users to create a Virtual network with just a few lines of code. It provides a vision of the network with the control plane separated from the data plane and is managed centrally. It also enables easy integration of new functionalities. While SDN is rapidly spreading through the network industry, it still faces issues with managing, monitoring, and controlling the network centrally. One of the major issues faced is the monitoring of the traffic flowing in the network. As there is rapid growth in data, there is an increase in data flow in the network every day. This leads to traffic congestion, and monitoring the traffic is a challenging task. The traffic can be caused due to several reasons, and one of the major reasons is due to large flows. These large flows are known as Elephant flows. This paper proposes an algorithm to detect the elephant flows in an Openflow network automatically. This algorithm sets a predefined bandwidth for the flow, and if the bandwidth exceeds the limit, the flows are automatically captured and presented in the form of a graph. This project involves Open Flow controller, OpenvSwitch and Open Flow protocol. In SDN, Open Flow provides additional features to help in monitoring the traffic. Previously, a lot of efforts have been made in this field to improve the quality of the network. The prime focus in this project is to collect the flow details and detect the elephant flows in the network automatically. The analysis and the enhancements carried out in this project have proved to be a better detector of the elephant flows in the network.

Keywords : SDN, mininet, Openflow, OpenvSwitch, sFlow, Elephant-flows.

1 Introduction

Virtualization concept has seen a rapid and a drastic growth in the IT industry. It has become a critical and an essential part of the industry. Virtualization technology helps in providing the maximum utilization of the underlying resources. This has become a boon and an added advantage. Unlike Virtual devices, Physical devices fail to utilize the resources to the maximum level. Virtualization has spread across different arenas like the storage, networking, application and so on. Network virtualization is introduced as it has many advantages over the hardware network devices, like the network connectivity can be done logically rather than having many physical connections connecting the servers, switches, routers.

Network Virtualization is made easy by introducing the concepts like the SDN. Software Defined Network has enabled the network connectivity through commands. It allows the creation of virtual devices like the switches, routers, controllers, and many more. Traditionally, the network devices like the switches and the routers included control and data planes on the same device. Due to this, it proved difficult to manage, monitor a network with huge number of switches, routers, and hosts. According to the research by (Bernal et al.; 2016) SDN detached the control plane from the data plane as shown in 1(Behance; n.d.) where the data plane remained intact within the device and the control plane was integrated within a separate device known as the controller, which was centralized. Earlier, the number of control planes present in the network would be equal to the number of network devices such as the switches connected. With the emergence of SDN, a network topology can have one or more controllers based on the size of the topology. SDN can be subcategorized into three versions namely: Controllers, South and Northbound APIs. Controllers in SDN is a programmable central system, which controls the entire network as it will have the information of all the resources (like the switches, routers) connected to this network. Southbound APIs are the medium for the Controllers to transfer the information to the resources like the switches and the routers. One of the most commonly used standard southbound APIs is the Openflow. To interact with the applications, SDN uses northbound APIs. These are usually used by the administrators, who are monitoring and managing the traffic in the network. The major advantage of SDN is that,the application can be easily deployed and modified as per the requirement, as it is programmable.

Controllers in SDN play a vital role as they are responsible for managing the entire network centrally and instruct the underlying devices like the switch to perform the forwarding action for the data flow. It contains detailed information about all the underlying devices and its connectivity status. It also includes the details of the flows that enter the network, these details are stored in the form of table known as the flow-entry table. OpenFlow controller is one of the most commonly used controller in SDN, which controls and manages the flow. Every request, in the form of packets or the dataflow must traverse through the Openflow controller and then through SDN, as controller is the one which manages the traffic of the data and decides and provides instructions to SDN. In the topology, all the switches must be connected to at least one controller. Whenever the data enters the network, the Openflow controller obtains the information about the data from the switches. The controller then checks for the flow entry in the flow table, if the flow entry exists, it sends back the instructions to the switches with further actions to be taken by the switch.

SDN helps in creating virtual network devices such as the virtual switch, virtual routers. These virtual switches are a set of programmable code which act as an application. This application helps in connecting the virtual machines for communication purpose. Unlike the physical switch, which has a main purpose of redirecting the packets, Virtual switches outperform physical switch as it does not always depend on the controller for the instructions on redirecting the packet. Virtual switches also help in connecting the physical devices to the network. There are several virtual switches developed with a purpose to solve the drawbacks of the physical switches. Few virtual switches available

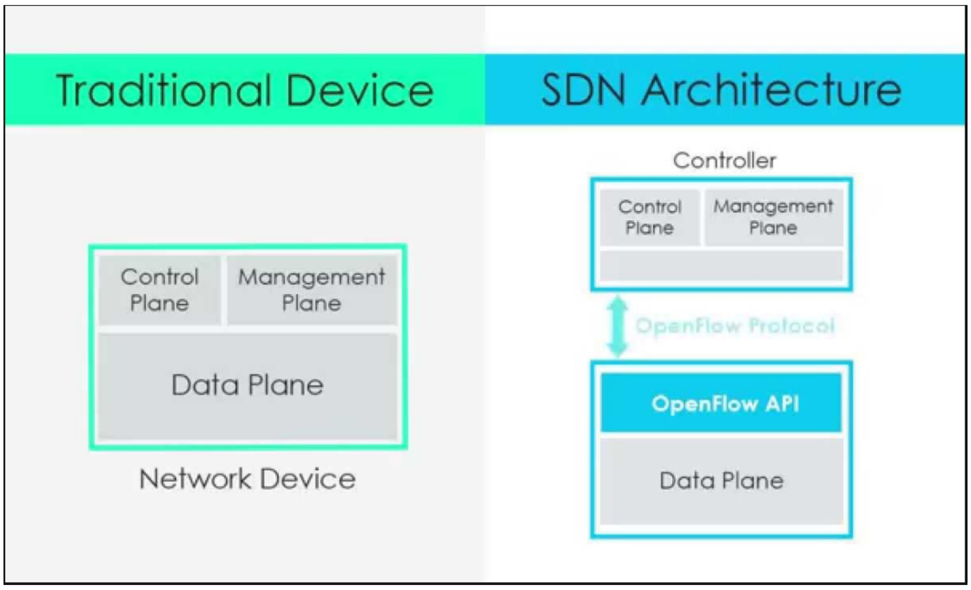


Figure 1: : SDN Architecture

are the OpenvSwitch, Indigo Virtual switch, OpenContrail, etc. According to (Rahimi et al.; 2016) among the various virtual switches mentioned, Openvswitch is proved to be one of the most efficient and widely used virtual switch. According to (Gorja and Kurapati; 2014) Openvswitch is also known as the OVS is a multi-layered open-sourced virtual switch, as it provides support for layers from L2 to L4 in the networks architecture. It outperforms other virtual switches through the following enhanced features like the supporting of OpenFlow protocol, IPV6, tunnelling protocols like GRE and has a feature of multi-table forwarding. OpenFlow is one of the most common protocol or the standard communication medium between the two planes i.e., Control plane and the forwarding plane. According to (Cheng et al.; 2015), these OpenFlow protocols are responsible for transmitting the data from the application to the network devices via controller and SDN. Controller and SDN instruct the switches and routers about the dataflow and how the data packets need to be redirected.

To evaluate different types of network topologies in small scale, network simulators play an important role. There are several network simulators available namely: Mininet, GNS3, EsiNet and so on. According to (Rehman et al.; 2014) these simulators provide a platform to set a network topology as a replication to the real-world environment. This project requires an open-source network simulator to generate the traffic and analyse its flow. Mininet is an efficient open-source network simulator which supports protocols like the OpenFlow along with ethernet network to provision virtual switches like OpenvSwitch. These simulators help in conducting experiments to analyse the network prior to its launch into the real environment. Mininet also allows you to create different kinds of topology based on Python scripts and supports multiple resources like the Controller, virtual switches, hosts, and links. Mininet provides information like the data log, trace logs and many more for the analysis. Mininet is a CLI enabled simulator which supports the use of analysis tools like the sflow, netflow, Rmon and so on.

It is necessary to monitor the network to improve the performance and to develop different methods to handle the traffic flow in the network. One such monitoring tool is SFlow, (Swapna et al.; 2016), which is an open-source sampling tool used for measuring and monitoring of traffic. SFlow is a monitoring too,as ahown in figure 2 (Brocade; n.d.) which is compatible with OpenFlow network. It consists of two elements, namely SFlow agents and collector where an agent which is installed in the device and it collects the data flow samples in the form of datagrams. The collector collects the datagrams from the SFlow agent and the collectors are installed in the host that is responsible for monitoring. This tool analyses the data and provides the result in the form of graph and statistics. This helps in understanding the data in a better way.

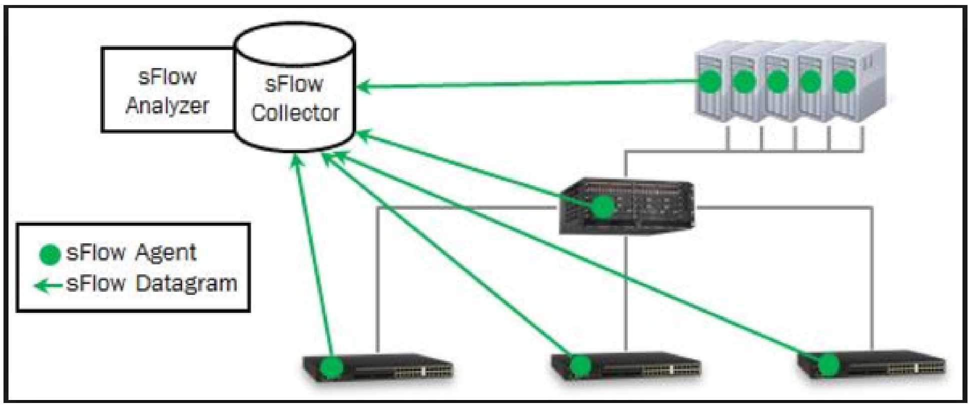


Figure 2: : sFlow Architecture

Data that is transmitted in the network can be either in the form of packets/blocks of data or complete data in the form of flow. In this project, the main focus is on flow based data. The data that flows through the network has shorter bandwidth. These flows do not cause the traffic in the network. According to (Afaq et al.; 2015), there are flows which have higher bandwidth and these flows tend to cause traffic in the network. This leads to delay in transmission of the data, which leads to decrease in the quality of performance of the network. The flows in the network can be classified into two types: Elephant-flows: which have higher bandwidth and Mice flows- which have lower bandwidth. These flows are usually generated if there is a huge transfer of data like backups. My project involves in analysing these Elephant-flows in the network automatically.

The research Question- **Can the elephant flows in openflow-based openvswitch be detected automatically?** This question is of importance and interest in determining the cause for the traffic in the network. This project provides information about the elephant flows that are detected along with its source IP and the port number.

To elucidate the determination of this research, this paper is categorized into two subsections: first subsection deals with the literature review. This subsection focuses on the past researches that are conducted to detect the elephant flows in the network, its encounters that are faced throughout the execution and to analyse the breaches that are to be bridged.

Second subsection is about methodology. This section explains about the different techniques and methods that are revised in order to obtain the proposed result. This comprises of a series of steps that are involved in identifying the Elephant flow and analysing the flow in the network.

2 Related Work

To appreciate the novelty of the paper, it is very much essential to comprehend the equivalent work and the energy level exhibited by the previous papers. These help in gaining knowledge and bridging the gap for the future improvement in the same field. The related work can be better understood, when explained in subsections. This section has 5 major subsections namely: 2.1, 2.2, 2.3, 2.4, 2.5

2.1 Importance of Openvswitch in SDN

Virtual switches were designed as they could be eradicating the drawbacks of the physical switches. These virtual switches are mainly used to connect multiple virtual machines. They are easy to be deployed, and maintained as they contain a set of programmable code. Few researchers conducted a series of experiments on virtual switches, to check the performance of the virtual switches with the real network. According to (Pettit et al.; 2010), they found that among all the virtual switches available for the experiment, they found that openvswitch,as shown in the architecture figure 3 (N; n.d.) a multilayer switch, is more compatible and easy to deploy. It can be integrated with the traffic flow in the network using Netflow, Sflow, In their experiment, during the process of communication, the OVS must inherit an external interface which should be compatible with Openflow and Openswitch. But in contrary another experiment was conducted by (Bui and Aberkane; 2016), OVS do not require any external interface to interact with the openflow and the OVS, as a new generic interface was introduced for the virtual switches, to overcome the above-mentioned problem. The paper by (Hamadi et al.; 2014), has proposed a methodology to increase the speed of the OVS, which are in the overlay network, by using the offloading technique.

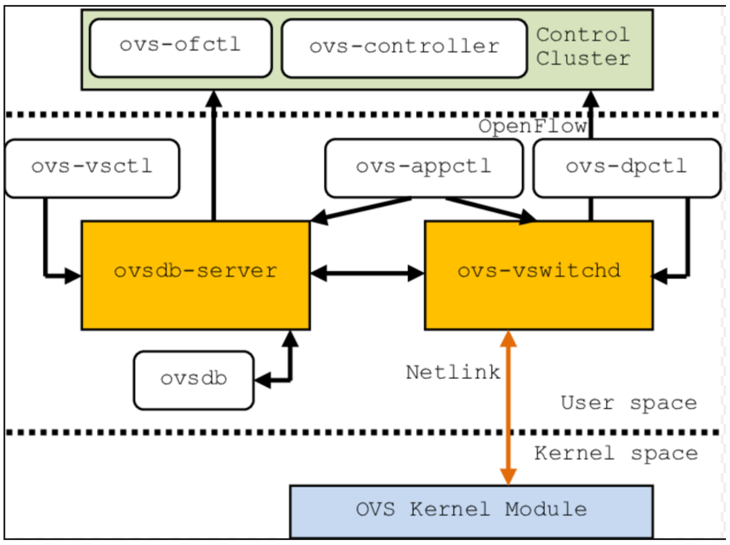


Figure 3: : The-Architecture-of-Open-vSwitch-and-Main-Components

2.2 Efficiency of Openflow(controller and protocol)

Openflow also plays an equally important role as the OVS. It is necessary to understand the evolution in the performance of the Openflow as many researchers have conducted experiments to improve the same. OpenFlow plays a vital role in transmitting the data between the virtual devices, SDN and the OVS. Openflow is used in switches to amend the flow tables and update them with the latest information on regular basis. An experiment conducted by (Rahimi et al.; 2016), proposes that, openflow can not only manage simple flow-tables and entries which are required when multiple heads are used. In addition to managing complex flow-tables,as shown in the figure 4 (Hedlund; n.d.) its performance can be increased by integrating a new functionality, which replicates the network because of the data flow information. new functionalities were introduced and tested by (Iguchi et al.; 2014), These tests were used to analyse the incoming data and take decisions based on the data flow. The performance of the Openflow can be increased by introducing a new function in the Openflow controller. The assumption and the test was conducted by (ejka and Krej; 2016), in which they have proposed that by introducing a new functionality on the Openflow controller, as shown in the below figure, the performance of the openflow increases. The functionality introduced is OVSDB, which is implemented as a functionality inside the Openflow controller. OVSDB is responsible for maintain all the flow tables that are present in the network and with frequent updates and new entry additions.

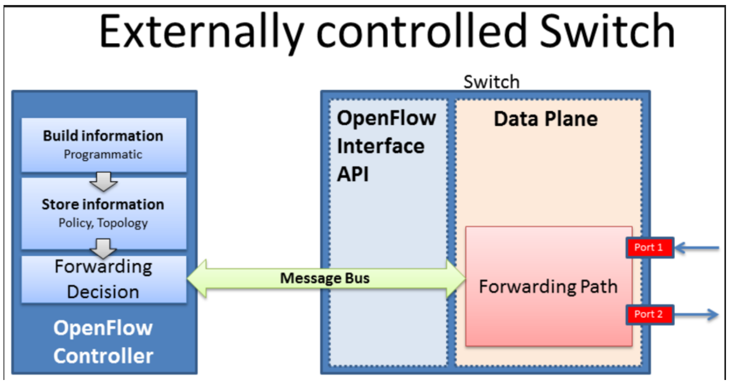


Figure 4: : The-Architecture-of-openflow-controller and protocol

2.3 Measuring the performance of Mininet

Mininet is a network emulator which help in evaluating new algorithms or new features in SDN. many experiments have been conducted to analyse and improve the performance. The performance of mininet with other testbeds were compared and analysed by (Barrett et al.; 2017), they proved that mininet emulators are better performer for testing. It is compatible with Openflow-based controllers and switches. Mininet simulators are advantageous as they help in replicating the real-time network which consists of topologies and protocols to connect them as shown in the figure 5 (Seetharaman.s; n.d.) . To check the performance and compatibility factors for mininet are experimented by

(Ortiz et al.; 2016). They replicated a large network topology that is usually created for large datacentres. Their results proved that mininet can perform well based on the underlying hardware specified and provide potential results on different tests. According to (Erel et al.; 2015), mininet not only performs the functionality of simulator, but also helps in simulating FAC-Flow Admission Control. They have used openflow switches and openflow protocol for the communication between the switch and the controller which is deployed on mininet. To overcome the problem of monitoring, managing and control the network flow and improve the Qos of the network, (Hamad et al.; 2015), has conducted an experiment to measure the traffic and to obtain the data from the underlying network devices. They have used Mininet simulator to create a network topology consisting of floodlight controller, CPqD switch and a openflow protocol. The above observations provide a stronger reason to choose mininet over other emulators.

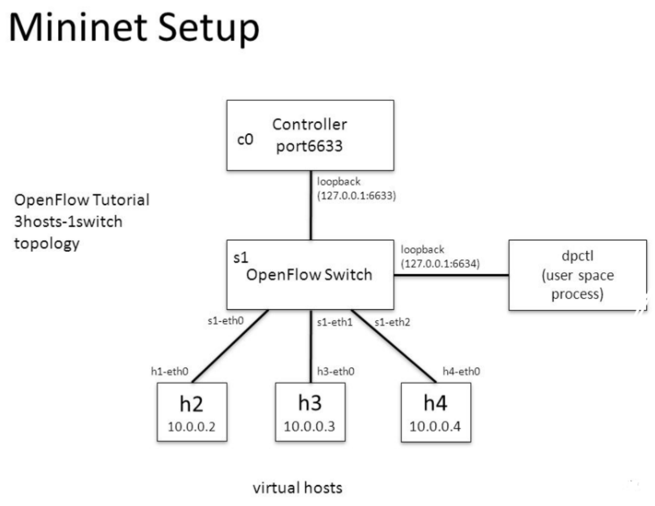


Figure 5: : Mininet topology

2.4 Evaluation of sFlow tool

The proposed paper focuses on analysing the data flow in the network and there are several monitoring tools available and researchers have contributed to improve the tools with advanced features and to provide accurate monitoring results. According to (Rehman et al.; 2014), they have experimented on monitoring the flow on OF@TEIN simulator. They have compared the performance of sFlow tool with other monitoring tools such as NetFlow, RSPAN ports. It resulted in proving that sFlow is a better performer in terms of providing a network-wide visibility. (Swapna et al.; 2016) has done a research regarding the working procedure involved the process of flow monitoring for security analysis in the wireless network using the tools such as sFlow and flowVisor. In this paper, they have conducted an experiment to study the security for data flow. They have analysed the threat created for the data in the network using STRIDE model. SFlow and the flowVisor have proved to be useful in collecting the data and analysing the same. (Jasinska; 2006), has conducted an experiment on how the monitoring tool named sFlow can be used to monitor the traffic in the network and provide information to the users to optimize the usage of network. The experiment was conducted for the Internet Exchange in Amsterdam. SFlow tool can be used to monitor sampling topology, which enables it to also monitor high-speed networks. Since Amsterdam Internet exchange is one of the largest exchange with highest data flow, and the experiment resulted in proving that sFlow tool is efficient in handling large traffic. The experiments conducted proved that sflow is a better monitoring tool, hence choosing the same for my project.

2.5 Analysis of Elephant flows

Sometimes the data will flow will be larger than the normal data flows. These tend to cause congestion in the traffic and thus delay in flow. These large flows are categorized as elephant flows. Researchers have made efforts in detecting the elephant flows. One such experiment was conducted by ((Afaq et al.; 2015), who tried to detect the elephant flows and improve the quality of service of flow which will be administered by the controller that will be centrally located in the network. They have tried to shape the traffic by limiting the rate of the flow. Experiment was conducted by (Knob et al.; 2016), to detect the elephant flows in IXP network. IXP (Internet Exchange Points) is cost-efficient independent systems, in which they have tried to develop a better monitoring and detection system for the controller to detect the elephant flows in a scalable and efficient manner. In order to predict and observe the future traffic (Xiao et al.; 2015), have conducted an experiment to classify the flows using a cost-sensitive technique to detect elephant flows. They have used cost-effective decision trees which would help in testing large datasets and to overcome the overhead problem. One of the major issues faced by the datacentre is the network congestion due to the transmission of large amount of data at once, which leads to a creation of elephant flows. (Liu et al.; 2014), has conducted an experiment to determine the elephant flows and split the elephant flows to balance the load in the network. It was tested on openflow testbed. They were successful in balancing the load and increasing the throughput for the network. Since the controller is centrally managed and integrated with many functionalities, which leads to overhead (Hong and Wey; 2017), conducted an experiment to detect the flows entries which consume higher bandwidth. They have also focused on identifying security-sensitive flows. In this the controller uses access control rule which will manage the elephant and security-sensitive flows. Overall elephant flows are proved to play a major role in causing network traffic. Many efforts are made to reduce the elephant flows in network.

3 Methodology

In a challenge to meet the requirements of the rapid growth in the data flow in network, network congestion, plays a major role. Many efforts and methodologies are adapted to deal with the traffic congestion. This research paper has as well attempted to make a humble effort to deal with congestion issue by detecting the elephant flows in the network using a virtual switch, Openvswitch.

The paper delivers several techniques used to detect and analyse the elephant flows in the network. This paper has an algorithm designed to detect the type of incoming flow into the network. This algorithm segregates the elephant flows from the mice flow. Based on this information, the controller will be able to identify the source and the destination of the flow. This algorithm will be embedded in the controller to centrally manage the data flow and the traffic generated through the flow in the network.

To analyse the traffic in the network, this paper concentrates on the essential rudiments that are selected. These rudiments are introduced and briefly enlightened as shown in the figure 6 (Hofstede et al.; 2014)in the earlier section. This paper also explains the necessary steps that are involved in handling the traffic, by detecting the elephant flows in the network. Customarily, various monitoring methods demands for installation of additional hardware and third-party software configurations, which would result in dreary, extensive, and affluent task. Emergence of Software-defined network, with openflow has eradicated the issue, as openflow has all the obligatory interfaces that are entrenched in it. One of the major rudiment involved in this paper is the Openflow controller along with openflow protocol, and openvswitch as they provide all the necessary interfaces required for this project.

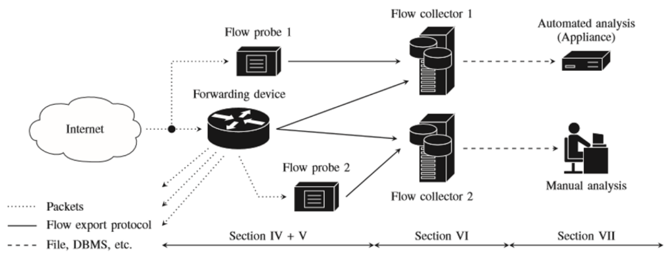


Figure 6: : Data Flow mechanism in network

Main aim of this project is to detect the elephant flow that causes traffic in the network. This can be attained by using the message protocol provided by the openflow. (Hamad et al.; 2015)These message protocols provide the data and the outcome that can be archived and analysed. The two types of message protocols involved are: STATISTICS REQUEST and STATISTICS REPLY. **Statistics Request**: A message is transmitted from the Openflow controller to the Openvswitch to request for the details like the flows and the ports involved, in the form of data. **Statistics Response/reply**: A message transmitted from the Openvswitch back to the openflow controller with the requested data.

There are basically two ways to measure the traffic flowing in the network, namely: **active network measurement** and **the passive network measurement**. Active network measurement basically involves an additional network flow (sample test flow) that is generated to observe the outcomes. These observations are made based on the single point in the network. Additional network flow is induced into the network along with the original network flow and observations like the quality and the performance is measured. The other type is the passive network measurement. In this type, it does not involve any additional flow like the active network measurement. In this the network is measured based on the real traffic, which will help in obtaining the results of loss of packets in the flow, accuracy of the flow and many more. This paper chooses an active network measurement in order to obtain the results of the performance of the network, by analysing the elephant flow and also to monitor the behaviour of the flow.

Network measurements are backed by infrastructural resources like the switches, virtual switches, Routers, virtual routers and many more. Traditionally, the measurements were made based on the ports and this would restrict in determining the quality and the latency etc of the flow. With the emergence of SDN and the virtual resources, the granularity of the measurement has increased tremendously. Since my focus is to measure the traffic in the network using the virtual switches, the underlying resources include a virtual switch (Openvswitch)(vSwitch; n.d.), a network simulator (mininet)(Mininet; n.d.), network measuring protocol and a graph generator (sflow) and a protocol to connect the switch, controller, and the hosts (openflow protocol). The observations obtained from the test conducted, the further section, will consolidate the data and send the data as inputs to the sflow. This sflow is customized to read the input values and to provide a graph. This graph represents the traffic congestion in the network. The paper collaborates the algorithm, for detecting the elephant flow, along with sflow. Hence the graph obtained will also represent the type of flow along with the traffic flow.

Amalgamation of the underlying resources in a most effectual manner with the necessary algorithms induced in them. This integration and its architecture is described in the further sections.

4 Implementation

To grasp the systems method, we need to appreciate the complete architecture of the system. Openflow (N; n.d.), an SDN based protocol empowers in transmitting the data in the form of flows and not as packets. This ensures in the increase in performance of the network from the source to the destination and vice versa. The openflow-based controller has functionalities like adding and deleting of the flows. Every action taken on the flow, will have the data of the flow, updated in the flow-table. Openvswitch and the openflow-based controller interact with each other through the queue system, where the controller will send the information or the flows in the form of queue to the switch. Queuing depends on the number of links/paths that are formed by the number of switches that are connected to the controller. The flows that enter the network are matched with the flow table provided by the controller, which includes fields such as the type of ethernet, type of protocol used, MAC and IP address and so on. Controller pre-defines the policy and flows that enter the network are detected and checked for the existence of its information in the flow table and matches the policy set. If the flow does not match flow table, the controller adds the flow to the flow table with all the information updated and sets a new policy for the flow

Since the technology has evolved to a virtualized environment from the traditional physical environment, the data growth has also increased. Due the multiple virtual switches, routers, hosts and controllers, it is hard to determine the cause for the traffic and from which virtual resource is it generated. This proposed architecture enables multiple virtual resources like the openvswitches, hosts and openflow controller and protocol to connect on a single machine using mininet, a network simulator. The overview design of the Mininet simulator figure 7(Rahulait; n.d.) with openflow controller and number of Openvswitches.

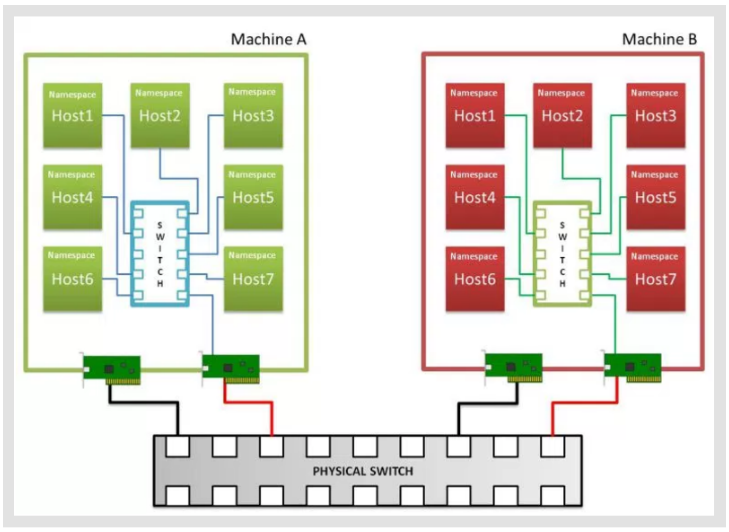
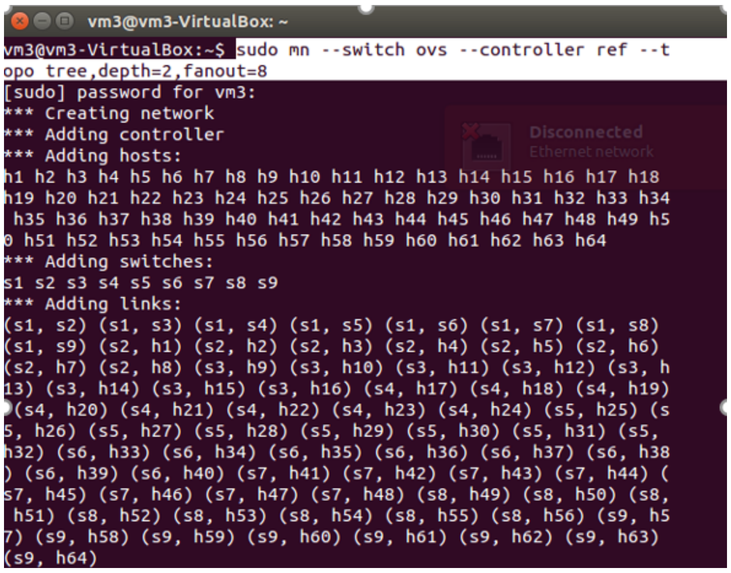


Figure 7: Mininet integrated with OVS and OF

This projected architecture consists of a host computers physical switch, with the help of mininet, the physical switch is virtually disintegrated as virtual NICs for the virtual machines. Each virtual machine in the mininet has several hosts, virtual switches and controllers interconnected. The above architecture describes the integration of openvswitch with the controller via openflow protocol.

4.1 Creation of Network Topology

In order to have a flow generated in the network, mininet allows the resources to communicate with each other by creating a network topology. Mininet uses linux commands to create a network topology as shown below:



```
vm3@vm3-VirtualBox:~$ sudo mn --switch ovs --controller ref --t
opo tree,depth=2,fanout=8
[sudo] password for vm3:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34
 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h5
0 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s1, s5) (s1, s6) (s1, s7) (s1, s8)
(s1, s9) (s2, h1) (s2, h2) (s2, h3) (s2, h4) (s2, h5) (s2, h6)
(s2, h7) (s2, h8) (s3, h9) (s3, h10) (s3, h11) (s3, h12) (s3, h
13) (s3, h14) (s3, h15) (s3, h16) (s4, h17) (s4, h18) (s4, h19)
 (s4, h20) (s4, h21) (s4, h22) (s4, h23) (s4, h24) (s5, h25) (s
5, h26) (s5, h27) (s5, h28) (s5, h29) (s5, h30) (s5, h31) (s5,
h32) (s6, h33) (s6, h34) (s6, h35) (s6, h36) (s6, h37) (s6, h38
) (s6, h39) (s6, h40) (s7, h41) (s7, h42) (s7, h43) (s7, h44) (
s7, h45) (s7, h46) (s7, h47) (s7, h48) (s8, h49) (s8, h50) (s8,
 h51) (s8, h52) (s8, h53) (s8, h54) (s8, h55) (s8, h56) (s9, h5
7) (s9, h58) (s9, h59) (s9, h60) (s9, h61) (s9, h62) (s9, h63)
(s9, h64)
```

Figure 8: creation of network topology on mininet

The created network topology has the following as shown in the figure 8:

- Tree topology
- 64 hosts
- 9 switches

The tree topology that is formed with 1 openflow controller which controls 9 OpenvSwitches and all the 64 hosts are connected, with each host being connected to at least one switch. Elephant flows are the large flows that interrupt the smooth flow, thus causing traffic. In the topology, we can define the elephant flows by providing the range of the bandwidth. If the flow exceeds the given range, those flows are considered to be elephant flows.

The advantage for using mininet simulator is that the network topology, that is created using the linux commands, is static.

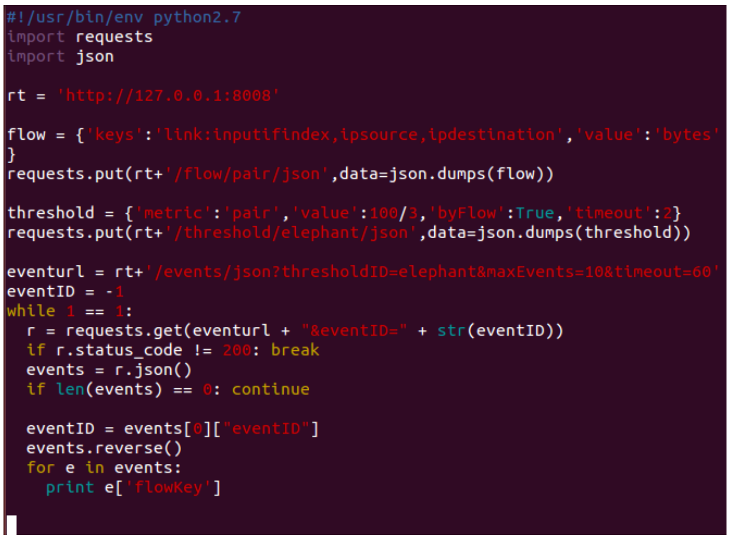
The static nature of the network helps in testing, observing and analysing the type of the flow. Every host in the topology is defined with a bandwidth of 10Gps to transmit the data flow across the network. This framework is designed to generate the different flow types such as UDP, ICMP, and TCP. These flows are generated using the Linux commands, at the host layer and are transmitted across the switches. The controller captures the flows and looks for the flow information in the flow-entry tables. Required actions are undertaken by the controller according to the flow entry information, policy and instructs the switch to forward the flow to the respective destination.

With all the connections intact, the hosts can generate the different traffics (TCP, UDP and ICMP). The mininet is compatible with visualization software namely: wireshark, sFlow, NetFlow etc. sFlow has been chosen for this project and it is integrated with the openvswitch. Once the traffic is generated, the visualization framework is executed (start.sh). This allows a graphical representation of the traffic across the network. An algorithm(elephant.py) is integrated with the sFlow, to detect the elephant flow in the network. This algorithm is linked with the topology that is created and in this algorithm the bandwidth, for the flow that can enter the network is defined. SFlow is configured with mininet, openvswitch and openflow controller. The below pseudo code sets a bridge connection between the resources to communicate with each other. Once the network topology is created with the help of Openvswitch and Openflow controller, sFlow (huawei; 2017) captures the information of the topology and creates a flow graph.

Unlike other tools, sFlow monitoring tool does not require functionalities like hashing, flush, decoding and flow cache (Blog.sflow.com; 2015) sampling function includes information about the flow and these samplings can be used for analysing the traffic. Polling function collects the information about the interfaces that are encountered during the transmission.

4.2 Detection of Elephant flows

The installation of the openvswitch, controller and sflow is complete, code configured and executed to create a communication bridge between the resources as shown in the figure 9. The below graph shown in figure 11 is linked with the topology creation and configuration of the flow. In this algorithm, the threshold value for the elephant flows are defined and the values are passed to the sflow. The graph then show the source and the Ip address of the elephant flows:

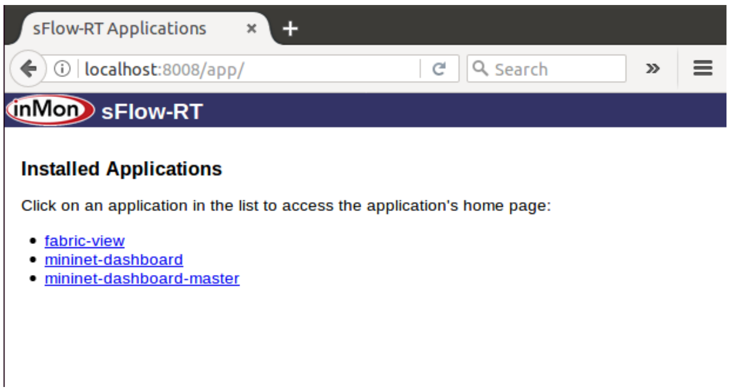


```
#!/usr/bin/env python2.7
import requests
import json

rt = 'http://127.0.0.1:8008'

flow = {'keys':'link:inputifindex,ipsource,ipdestination','value':'bytes'
}
requests.put(rt+'/flow/pair/json',data=json.dumps(flow))

threshold = {'metric':'pair','value':100/3,'byFlow':True,'timeout':2}
requests.put(rt+'/threshold/elephant/json',data=json.dumps(threshold))

eventurl = rt+'/events/json?thresholdID=elephant&maxEvents=10&timeout=60'
eventID = -1
while 1 == 1:
  r = requests.get(eventurl + "&eventID=" + str(eventID))
  if r.status_code != 200: break
  events = r.json()
  if len(events) == 0: continue

  eventID = events[0]["eventID"]
  events.reverse()
  for e in events:
    print e['flowKey']
```

Figure 9: Elephant flow code

In order to obtain better results of the flow, I have integrated three types of graph views as shown in figure 10 in the sFlow tool namely: fabric-view, mininet dashboard view,mininet dashboard- master:

In Fabric-view the flows are classified as elephant and mice flows and this view provides a near real time visibility of the network. It banquets the traffic transversely to the switches and the links which will help in upsurge of the bandwidth. It identifies the protocol used and also detects the elephant flows in the flow (Blog.sflow.com; 2015)



Installed Applications

Click on an application in the list to access the application's home page:

- fabric-view
- mininet-dashboard
- mininet-dashboard-master

Figure 10: Http view of the sFlow

Mininet dashboard is a web interface which provides information about the top flows and ports along with the topology diameter. The top flows are the flows which are active at all times and top ports are those ingress ports which are active at all times.

5 Evaluation

The elephant flow algorithm is able to detect the elephants or the large flows that are created in the network. Below is the flow chart (figure 12 which represents the process in which the openvswitch is integrated with sFlow and the elephant flow algorithm. Firstly, the elephant flow algorithm is defined in the sflow using python language. Next, the threshold for the elephant flows is defined and this will be integrated by sflow, so both definition and the threshold values are pushed to sflow. On the generation of a TCP/UDP or ICMP flow, sflow extracts the information from the flow generated and compare it with the definition and threshold value. If the flow exceeds the threshold value, then it considers the flow as elephant flow and collects the values from the flow and stores it in the flow key. If the flow is not an elephant flow, it will analyse the next flow. A network topology of 1 controller, 9 switches and 64 hosts is created. Alongside sflow monitoring tool is started with the command **./start.sh**. The http format of sflow is started with the url: localhost:8008. This is to obtain the graphs of the flows.

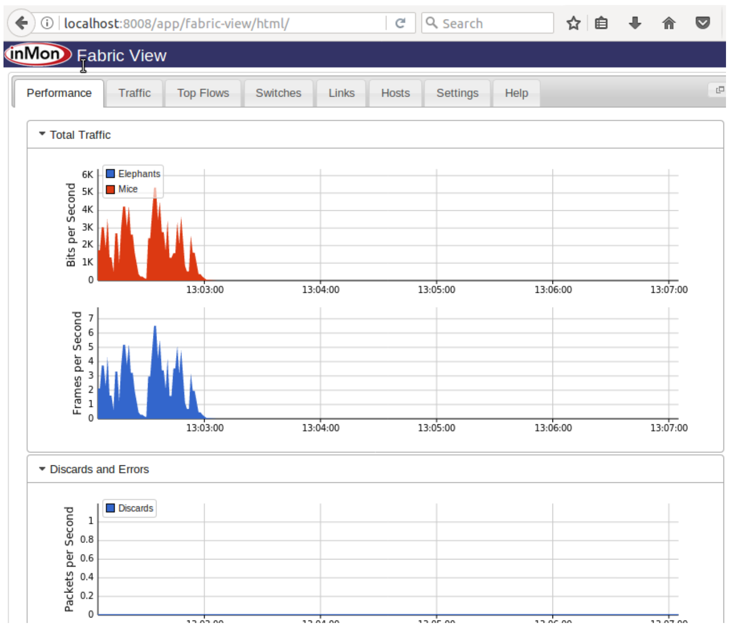


Figure 11: : Elephant flows

Following graph represents the elephant flows that are detected during the flow. These elephant flows are calculated as frames per second and also provides the details such as the source IP address and the respective port number. In this the threshold is set as 100000/8 bps. Below are the diagrams which display the information of the flow in detail along with the information about switches and the links. The results obtained is for the test executed for five minutes

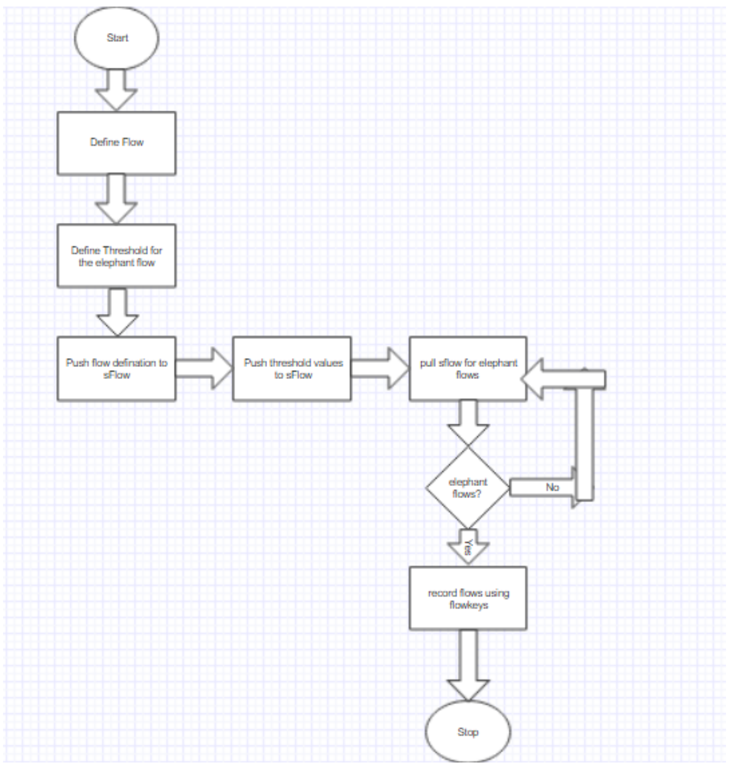

Start
Define Flow
Define Threshold for the elephant flow
Push flow defination to sFlow
Push threshold values to sFlow
pull sflow for elephant flows
elephant flows?
No
Yes
record flows using flowkeys
Stop

Figure 12: : Flowchart represent of the process

6 Conclusion and Future Work

This project has implemented a new algorithm which can detect the elephant flows in the network. The controller analyses the flow and provides the flow details to sFlow. Several studies and observations are made using different monitoring tools to detect the elephant flows. This paper focuses on detecting elephant flows for Openflow-based controller and Openvswitch. Experiments have been conducted in the past to detect the elephant flows for openvswitch, but they tried with only one type of flow-TCP. This project extends the detection method by testing it on two types of flows and proved that large flows can be detected for ICMP flows as well. The performance of the network can be improved in future by detecting the elephant flows and predicting the pattern of the flow. The performance can be improved in future work to detect the elephant flows and change the routing table accordingly.Another possibility is to study the traffic and split the larger flows into smaller flow to improve the throughput and to decrease the overall latency.

Acknowledgements

I Ms. Spurthi Mallesh, would like to place on record the invaluable contribution, guidance, encouragement and support by my mentor (Vikas Sahni). I would also like to thank our programme director (Dr Horaccio) for guiding me throughout the course. I would like to acknowledge the support by my parents who have encouraged all the time. I also would like to show my gratitude to all friends who have been given wonderful support and encouragement during the course of preparation of the project.

References

Afaq, M., Rehman, S. U. and Song, W. C. (2015). Visualization of elephant flows and qos provisioning in sdn-based networks, *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 444–447.

Barrett, R., Facey, A., Nxumalo, W., Rogers, J., Vatcher, P. and St-Hilaire, M. (2017). Dynamic traffic diversion in sdn: testbed vs mininet, *2017 International Conference on Computing, Networking and Communications (ICNC)*, pp. 167–171.

Behance (n.d.). Centralized control over wireless network using software defined network, https://www.behance.net/gallery/15670859/Presentation-for-Final-year-Project-based-on-SDN Month =.

Bernal, M. V., Cerrato, I., Risso, F. and Verbeiren, D. (2016). Transparent optimization of inter-virtual network function communication in open vswitch, *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, pp. 76–82.

Blog.sflow.com (2015). Fabric view, http://blog.sflow.com/2015/10/fabric-view.html.

Brocade (n.d.). Campus network solution, best practice-sflow monitoring for brocade products, http://community.brocade.com/t5/Campus-Networks/Campus-Network-Solution-Best-Practice-sFlow-Monitoring-for/ta-p/36688. Month =.

Bui, D. T. and Aberkane, K. (2016). A generic interface for open vswitch, *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pp. 53–57.

Cheng, Y. J., Huang, D., Lee, C. L., Lee, M. C., Chuang, B. W., Tsai, M. C., Huang, X. and Hsu, C. H. (2015). Towards a detailed openflow emulator, *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 127–132.

Erel, M., Teoman, E., zevik, Y., Seinti, G. and Canberk, B. (2015). Scalability analysis and flow admission control in mininet-based sdn environment, *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pp. 18–19.

Gorja, P. and Kurapati, R. (2014). Extending open vswitch to l4-l7 service aware openflow switch, *2014 IEEE International Advance Computing Conference (IACC)*, pp. 343–347.

Hamad, D. J., Yalda, K. G. and Okumus, I. T. (2015). Getting traffic statistics from network devices in an sdn environment using openflow, *ITaS* pp. 951–956.

Hamadi, S., Snaiki, I. and Cherkaoui, O. (2014). Fast path acceleration for open vswitch in overlay networks, *2014 Global Information Infrastructure and Networking Symposium (GIIS)*, pp. 1–5.

Hedlund, B. (n.d.). On data center scale, openflow, and sdn, http://bradhedlund.com/2011/04/21/data-center-scale-openflow-sdn/. Month =.

Hofstede, R., eleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A. and Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with netflow and ipfix, *IEEE Communications Surveys Tutorials* **16**(4): 2037–2064.

Hong, E. T. B. and Wey, C. Y. (2017). An optimized flow management mechanism in openflow network, *2017 International Conference on Information Networking (ICOIN)*, pp. 143–147.

huawei (2017). sflow configuration, http://support.huawei.com/enterprise/docinforeader/loadDocument

Iguchi, N., Tsutsumi, H. and Watanabe, K. (2014). Development of automatic network replication system for openflow network, *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems*, pp. 620–623.

Jasinska, E. (2006). sflow–i can feel your traffic, *23C3: 23rd Chaos Communication Congress*.

Knob, L. A. D., Esteves, R. P., Granville, L. Z. and Tarouco, L. M. R. (2016). Sdefixidentifying elephant flows in sdn-based ixp networks, *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pp. 19–26.

Liu, J., Li, J., Shou, G., Hu, Y., Guo, Z. and Dai, W. (2014). Sdn based load balancing mechanism for elephant flow in data center networks, *2014 International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 486–490.

Mininet (n.d.). Mininet walkthrough, http://mininet.org/walkthrough/part-3-mininet-command-line-interface-cli-commands. Month =.

N, M. (n.d.). Openflow: Enabling innovation in campus networks, http://archive.openflow.org/documents/openflow-wp-latest.pdf Month =.

Ortiz, J., Londoo, J. and Novillo, F. (2016). Evaluation of performance and scalability of mininet in scenarios with large data centers, *2016 IEEE Ecuador Technical Chapters Meeting (ETCM)*, pp. 1–6.

Pettit, J., Gross, J., Pfaff, B., Casado, M. and Crosby, S. (2010). Virtual switching in an era of advanced edges.

Rahimi, R., Veeraraghavan, M., Nakajima, Y., Takahashi, H., Nakajima, Y., Okamoto, S. and Yamanaka, N. (2016). A high-performance openflow software switch, *2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*, pp. 93–99.

Rahulait (n.d.). Using openvswitch to communicate between two different hosts on different machines, https://rahulait.wordpress.com/2014/08/17/using-openvswitch-to-communicate-between-two-different-hosts-on-different-machines/. Month =.

Rehman, S. U., Song, W. C. and Kang, M. (2014). Network-wide traffic visibility in of@tein sdn testbed using sflow, *The 16th Asia-Pacific Network Operations and Management Symposium*, pp. 1–6.

Seetharaman.s (n.d.). On data center scale, openflow, and sdn, http://slideplayer.com/slide/2576827/. Month =.

Swapna, A. I., Reza, M. R. H. and Aion, M. K. (2016). Security analysis of software defined wireless network monitoring with sflow and flowvisor, *2016 International Conference on Communication and Electronics Systems (ICCES)*, pp. 1–7.

vSwitch, O. (n.d.). Why open vswitch? open vswitch 2.8.90 documentation, http://docs.openvswitch.org/en/latest/intro/why-ovs/. Month =.

Xiao, P., Qu, W., Qi, H., Xu, Y. and Li, Z. (2015). An efficient elephant flow detection with cost-sensitive in sdn, *2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom)*, pp. 24–28.

ejka, T. and Krej, R. (2016). Configuration of open vswitch using of-config, *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 883–888.