Develop a fully integrated Software Defined Networking (SDN) test bed for Educational and Research Purposes

- Semester 7 Report -

Amarasinghe A.A.D.K.P
Hashantha H.L.S
Tharaka E.L.Y

Department of Computer Engineering
University of Peradeniya

Final Year Project (courses CO421 & CO425) report submitted as a requirement of the degree of
B.Sc.Eng. in Computer Engineering

October 2018
Supervisors: Dr. Asitha Bandaranayake (University of Peradeniya), Dr.Suneth Namal (University of Peradeniya)

I would like to dedicate this thesis to my loving parents and “teachers” . . .

Declaration

We hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my/our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

Amarasinghe A.A.D.K.P
Hashantha H.L.S
Tharaka E.L.Y
October 2018

Acknowledgements

And We would like to acknowledge Dr Asitha Bandaranayake ,Dr Suneth Namal and all who helped us.

Abstract

Software defined networking (SDN) is a new approach of networking that revolutionized the traditional way of networking. It makes easy to designing, building, and managing networks by separates the networks control plane and forwarding plane. Centralizing the control of the network and network devices become very simple hardware units, SDN offers very easy way of defining policies of the network and flexible ways of control and monitor traffic flows. Objective of this project is to build a SDN Testbed for educational and research purposes using commodity embedded hardware (e.g.: PCs, servers, NetFPGA). The ultimate goal of this is make an educational tool which offers a visualization of traffic flows and several advanced network functionalities like network slicing, dynamic resource allocation and network function virtualization.

Table of contents

List of figures **viii**

List of tables **ix**

Nomenclature **x**

1 Introduction **1**

1.1 Background 1

1.1.1 SDN Architecture 1

1.2 Motivation to the problem 2

1.3 The proposed solution 3

1.4 Milestones 4

2 Related work **5**

2.1 Related works done 5

2.1.1 Developing a Cost-Effective OpenFlow Testbed for Small-Scale Software Defined Networking 5

2.1.2 SDN Testbed for Undergraduate Education 5

3 Methodology **7**

3.1 Methodology 7

3.2 Physical Implementation of SDN testbed 8

3.2.1 Net-FPGA based SDN switch 8

3.3 Virtual Implementation of SDN Testbed 8

3.3.1 OpenDayLight SDN Controller 9

3.3.2 MiniNet SDN Emulator 9

3.3.3 Virtual Box 10

3.3.4 ELK stack 10

4 Experimental Setup and Implementation **11**
4.1 Configure the Net-FPGA and build the OpenFlow switch 11
4.2 Select most preferable SDN controller 13
4.3 Build the simple SDN network 13
4.4 Develop an application for analyzing network statistics and packet visualization. 15

5 Results and Analysis **17**
5.1 Results 17
5.2 Result Analysis 17
5.2.1 Inbound traffic of the network 17
5.2.2 Outbound traffic of the network 18
5.2.3 Received and Transmitted bytes of the network 19
5.2.4 Packet drops and errors of the network 19

6 Conclusions and Future Work **21**
6.1 Conclusion 21
6.2 Future works 21

References **23**

List of figures

1.1 SDN architecture 2
1.2 Proposed solution 3

2.1 Related work solution 1 6
2.2 Related work solution 2 6

3.1 Net-FPGA 1G CML Board 8
3.2 OpenDayLight Architecture 9
3.3 Mininet Architecture 9
3.4 Crossover of ELK stack 10

4.1 Net-FPGA in PCIe mode 12
4.2 Built SDN network 13
4.3 Connection between VMs 13
4.4 Configuration of VM1 14
4.5 Configuration of VM2 14
4.6 Interface configuration of Switch 14
4.7 ER-Digram for Database 15
4.8 JSON File format 16

5.1 Inbound traffic of the network 18
5.2 outbound traffic of the network 18
5.3 Received bytes of the network 19
5.4 Transmitted bytes of the network 19
5.5 Packet drops and errors of the network 20

List of tables

Nomenclature

Acronyms / Abbreviations

BEM	Boundary Element Method
DEM	Discrete Element Method
FEM	Finite Element Method
FVM	Finite Volume Method
PFEM	Particle Finite Element Method

Chapter 1

Introduction

1.1 Background

SDN is an alternative new paradigm to address the traditional networking paradigm problems like management issues, scalability issues. There are many researches undertaking many sub areas of SDN. Data center level there are many use cases of SDN networks. Programmers can create applications that perform actions on the underlying SDN network and can achieve flexibility, efficiency and most importantly resiliency for data centers.

1.1.1 SDN Architecture

Traditionally, control plane and data plane of a traditional networking architecture were bind together. In OpenFlow standard which was created in 2008, was recognized as the first SDN architecture which defined how should control and data planes would be separated and communicate with each other [4].SDN architecture divided into three separate sections,

Application Layer: In application layer SDN applications are running. SDN Applications communicate with the Control layer via well-defined application programming interface (APIs). In addition, the applications can build an abstracted view of the network by collecting information from the controller. These applications can perform networking management, analytics, security and business applications used to run large data center level.

Control Layer: The SDN Controller does maintain communication between networking component and the network application. To do that controller has well defined API called North bound and south bound. North bound API enable applications to

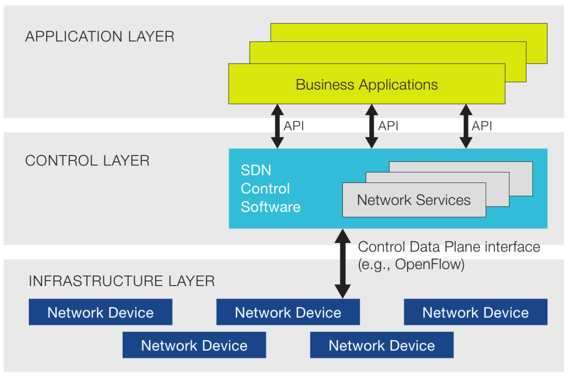


Fig. 1.1 SDN architecture

communicate with the controller and south bound API enable communication between controller and the network hardware. That defined controller as a logical entity that receives instructions or requirements from the SDN Application layer and relays them to the networking components. Also controller extracts information about the network from the hardware devices and communicates back to the SDN Applications with an abstract view of the network.

Forwarding Layer: The network devices which control the forwarding data and processing data for the network. Open flow is just a part of the SDN stack called as a south bound protocol [4]. This enable communication between controller and the network devices.Open flow is just a part of the SDN stack called as a south bound protocol [4]. This enable communication between controller and the network devices.

1.2 Motivation to the problem

Main issue of the SDN network implementations is initial cost of the implementation. This lead to doing researches on this area discouraged. To address this problems SDN networks for research emerged. For example, OF@TEIN, GENNY. But at beginners level it is hard to do simple experiments in this kind of complex networks. Then a cost effective, easy to implement SDN network implementation is required. Also it would be better for beginners to get first hands on SDN network experience before going in to large complex networks.Also the undergraduates who are eager to explore the different branches of networking , this would become facilitated platform. This Project is to build an SDN Test bed for educational and research purposes in order to facilitate the comprehension about Programmable Network using commodity embedded hardware (e.g.: PCs, servers, NetFPGA). The ultimate goal here is to make cost effective SDN test bed both virtual and realistic ways(hardware implementation) for beginners which offers a visualization of traffic flows and several advanced network functionalities like network slicing, dynamic resource allocation and Network Functions Virtualization (NFV), that could be used as a teaching aid, as well as a research test bed.

1.3 The proposed solution

Our objective is creating an extended SDN test bed with these functionalities and create documentations and tutorials based hands on experience SDN learning platform for beginners. New approach included NetFPGA based SDN switch for performance increment.

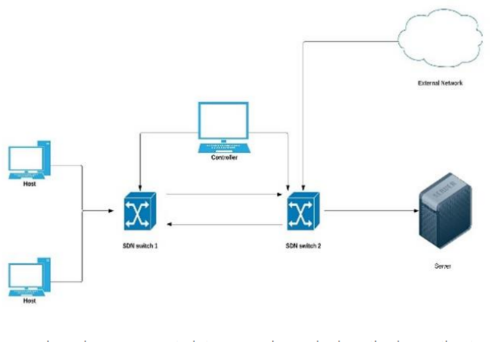


Fig. 1.2 Proposed solution

Programming NetFPGA board as a SDN switch is a complex task. described a Verilog implementation of a SDN switch and source code can be found on a GitHub repository [5]. Source code compatible with NetFPGA 10G variation. The paper helped to understanding the underline implementation of the source code [1]. Selected NetFPGA was NetFPGA-1G-CML is a versatile, low cost network hardware development platform featuring a Xilinx® Kintex®-7 XC7K325T-1FFG676 FPGA and includes four Ethernet interfaces capable of negotiating up to 1 GB/s connections. 512 MB of 800 MHz DDR3 can support high-throughput packet buffering while 4.5 MB of QDRII+ can maintain low-latency access to high demand data, like routing tables. Rapid boot configuration is supported by a 128 MB BPI Flash, which is also available for non-volatile storage applications. The standard PCIe form factor supports high-speed x4 Gen 2 interfacing. The FMC carrier connector provides a convenient expansion interface for extending card functionality via Select I/O and GTX serial interfaces. The FMC connector can support SATA-II data rates for network storage applications, and can also be used to extend functionality via a wide variety of other cards designed for communication, measurement, and control. There are several different SDN controllers which has different functionalities. In this approach it considered multiple SDN controllers like POX, Ryu, Trema, FloodLight, and OpenDaylight [7]. Here Controllers choose by it provided properties (like java interface provided default by the controller) and GUI functionality provided. Also tested all the different SDN controller by implementing them in a virtual network with Virtual machines. Among all of the controllers OpenDaylight was selected. It is planned to use two host computers and one server for first basic implementation. On top of this network design applications developed. When requirement changed it is expected to further expand the network.

Another main objective is increase the scalability of the SDN test bed .virtually created test bed perform and increase the scalability rather than physical SDN testbed .it can act as the middle platform on physical SDN test bed .All network applications development , network expanding approaches can built on virtual test bed .the testing applications and functionalities deployment processes can handle through the virtual testbed.it caused to cost reduction and avoid the unnecessary errors when dealing with hardware level of SDN testbed .finally both virtual and physical systems can manipulate as single system to get higher performance for research and educational aspects.

1.4 Milestones

There are few stages to pass for archive above described objective in phase one

- Configurations of hardware level is essential .Net-FPGA board is configuring as a SDN switch(implementations of hardware)
- The controller manipulate the whole SDN network.choose the prefer SDN controller and configure host with it.
- Create and deploy simple SDN network virtually(using virtual machines and proper SDN emulators) and implement using hardware level.
- Write and introduce an application which can show and analysis the packet delivery and statics.

Chapter 2

Related work

2.1 Related works done

Few similar approaches has been identified in SDN test bed implementations

2.1.1 Developing a Cost-Effective OpenFlow Testbed for Small-Scale Software Defined Networking

This has implemented.[1] such SDN testbed using Raspbery-pi module, virtual SDN switches and "Floodlight" controllerBut Raspberry-PI's processing power and resources is not enough when virtual switch instance running. It causes the raspberry-PI nodes to fail. Then constant operation was not possible. Also Raspberry PI consist of on Ethernet port. Multiple hosts should connect to SDN switches running on the Raspberry-PI. In this approach it used virtual Ethernet ports to do that. But it leads to performance of the system drops.Also lack of detailed documentations are caused to implementation and maintenance errors.

2.1.2 SDN Testbed for Undergraduate Education

most similar effort has been done by Computer Engineering department of University of Peradeniya[2] generally this approach tend to be covered the aspect of educational purposes.but as mentioned in its conclusion of report **"The tested performance statistics were much lower comparing to a network built on a Hardware OpenFlow switches like NetFPGA because of the speed limitations of USB to Ethernet converters and Raspberry Pi processor. But for non-performance critical SDN research and educational purposes statistics were in an acceptable**

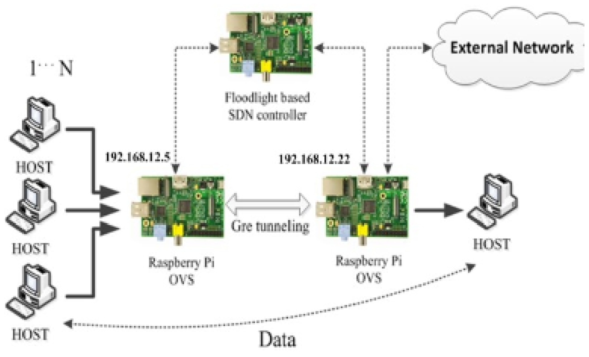


Fig. 2.1 Related work solution 1

range. In the future, The SDN testbed will be extended in to a full featured small-scale SDN offering additional features like traffic visualization, network slicing, dynamic resource allocation or network function virtualization”

new work suggest methods and expected to overcome performance issues using Net-FPGA to hardware level implementations.Also expected to expand to fully featured small scale SDN test bed with functionalities like network visualizations with combination of both virtual and physical implementations.

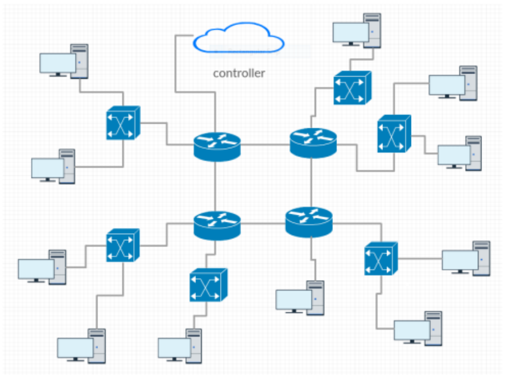


Fig. 2.2 Related work solution 2

Chapter 3

Methodology

3.1 Methodology

The whole SDN testbed creation process can separate to main two parts.built a physical and virtual deployed SDN testbed. Basically Net-FPGA[3] board was used to create physical SDN testbed.Further,"Zodiac-FX" educational SDN routers has similar ability to implement physically .Virtual testbed is running on MiniNet SDN emulator .the OpenDayLight SDN controller controls the network.All network data can get through a controller to visualization and monitoring tools for analyzing purposes.

Proposed network topology implemented inside Mininet emulator[4].Mininet runs on separate virtual machine.And the OpenDayLight Controller runs on another separate virtual machine.All the IP address and port numbers are automatically assigned to the network by Mininet emulator[5].This two virtual machines are connected to wifi uisng "network bridge interfacing" .when network functionalities begin , the statistics of network data is received to controller through OPenFlow protocol. Then the network application that we developed, gather those statistics and data using API calling .The data is build as JSON document. Using java application the JSON file processed and extract the required data. then push it to MySQL database.this procedure automatically repeat within particular pre-defined time periods.Separate java application gets the data from SQL database and process that data for compatible with ELK stack.

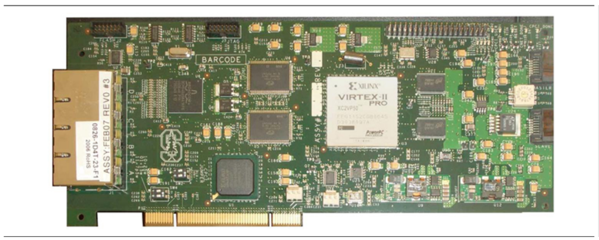


Fig. 3.1 Net-FPGA 1G CML Board

3.2 Physical Implementation of SDN testbed

3.2.1 Net-FPGA based SDN switch

The NetFPGA is the low-cost reconfigurable hardware platform optimized for high-speed networking. The NetFPGA includes the all for the logic resources, memory, and Gigabit Ethernet interfaces necessary to build a complete switch, router, and/or security device. Because the entire datapath is implemented in hardware, the system can support back-to-back packets at full Gigabit line rates and has a processing latency measured in only a few clock cycles.

Stanford university has developed Net-FPGA 1G board as the OpenFlow switch.https://github.com/NetFPGA/netfpga/wiki/OpenFlowNetFPGA100This project provides a hardware table implementation of an OpenFlow reference switch.Project was released version is OpenFlow spec1.0 and based on NetFPGA Base Package 2.2.0 .

Hear same Net-FPGA board was used to create above described SDN switch and followed the official documentation instruction to install "Base package of Net-FPGA" and then installed the OpenFlow switch on Net-FPGA board. below shows the commands that used to install process.

- sudo yum clean all
- sudo yum install netfpga-openflow-switch

The recommended operating system for Net-FPGA is "Fedora 20"

3.3 Virtual Implementation of SDN Testbed

Virtual SDN testbed is highly demanded for increase the scalability and cost reduction.Virtually implemented SDN testbed facilitated to get first hand experience and knowledge of SDN and it can be used as a basic practical tool.

3.3.1 OpenDayLight SDN Controller

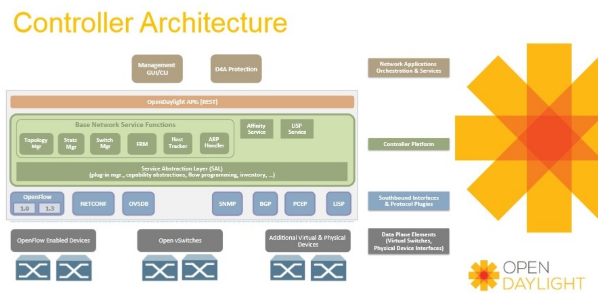

Controller Architecture
Management GUI/CLI
D4A Protection
Network Applications Orchestration & Services
OpenDaylight APIs (REST)
Base Network Service Functions
Topology Mgr
Stats Mgr
Switch Mgr
FRM
Host Tracker
ARP Handler
Affinity Service
LISP Service
Controller Platform
Service Abstraction Layer (SAL)
(plug-in mgr., capability abstractions, flow programming, inventory, ...)
OpenFlow
1.0
1.3
NETCONF
OVSDB
SNMP
BGP
PCEP
LISP
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)
OPEN DAYLIGHT

Fig. 3.2 OpenDayLight Architecture

OpenDayLight[] is SDN controller that used to manipulate the proposed network.Basically OpenDayLight act as collaborative open source project hosted by Linux foundation.OpenDayLight nitrogen 0.7.3 release was adequate to the requirements.

IP address and port number of the controller was added to the configurations of the network as a parameters.When network begin to initializing it automatically add to the network.

3.3.2 MiniNet SDN Emulator

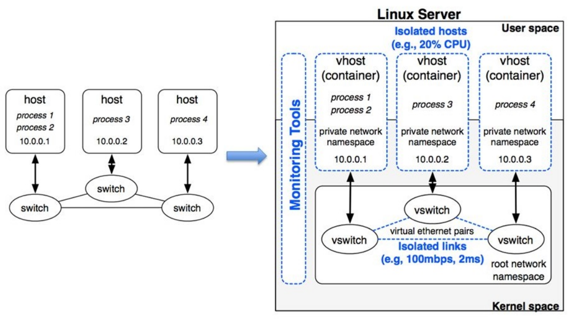


Fig. 3.3 Mininet Architecture

Mininet is SDN emulator that used to create above proposed network topology.It runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. It uses lightweight virtualization to make a single system look like a complete network, running the same kernel, system, and user code. Mininet host behaves just like a real machine.Mininet's virtual hosts, switches, links, and controllers are the real thing and they are just created using software rather than hardware and for the most part their behavior is similar to discrete hardware elements. It is usually possible to create a Mininet network that resembles a hardware network.

3.3.3 Virtual Box

All virtual machines are run on Oracle VirtualBox.Ubuntu operating system run inside all VM.

3.3.4 ELK stack

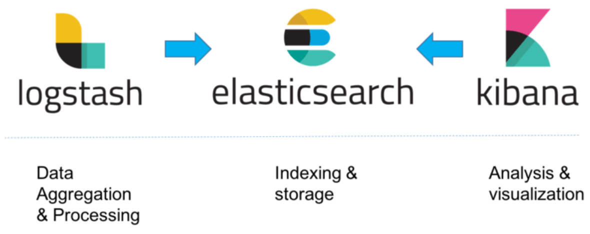


Fig. 3.4 Crossover of ELK stack

"ELK" is stand for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that inputs data from multiple sources simultaneously, transforms it, and then sends it to a receiver like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch.

Kibana was used to visualize the traffic statics of network.all the data of OpenvSwitch instances controlled by OpenDayLight controller. Mininet emulator and OpenDayLight controller runs on two separated VM.

Chapter 4

Experimental Setup and Implementation

Project has divided to four main milestones.

- Configure the Net-FPGA and build the OpenFlow switch
- Select most preferable SDN controller
- Build the simple SDN network.
- Develop an application for analyzing network statistics and packet visualization.

4.1 Configure the Net-FPGA and build the OpenFlow switch

System Requirements

A host system with a PCI Express-compatible motherboard and a dual-port NIC are needed to run the reference designs. During development and testing of the Net-FPGA-1G-CML designs, the following components are required:

ASUS M5A99FX PRO R2.0 motherboard Intel 82571EB dual-port NIC

Operating System:

Fedora 20 64-bit (kernel version 3.14.7 at the time of writing)

Xilinx Tools:

ISE Design Suite 14.6 Vivado Design Suite 15.2

Currently, only these versions of the tools are compatible with the repository without modification of source files.

First the Git repository cloned to local machine.Then the operating system was configured by installing all required libraries. Then NetFPGA can connect to host computer by using Standalone mode or PCIe mode. The PCIe mode was used and Board directly connected using PCI port of the host computer.

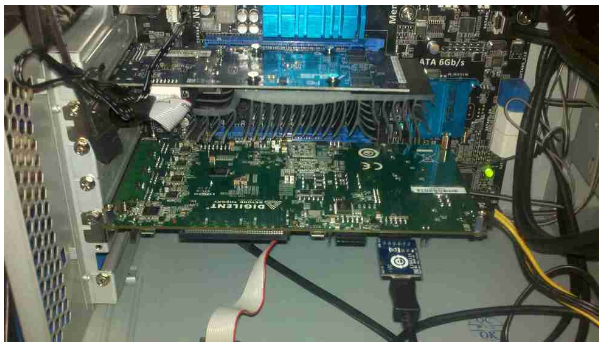


Fig. 4.1 Net-FPGA in PCIe mode

Then buid environment of the xilinx tools was done.Setting the Environment PATH of Xilinx and Vivado was done.

Next, set up the environment with the paths required by the NetFPGA-live build system. In the root directory of the repository there was a file **bashrc-addon-NetFPGA-10G**. In that file **NF-ROOT** set to point to the base directory of the repository and **NF-DESIGN-DIR** set to point to the folder of the project you wish to run.Then source it from the command line to update the environment variables. Then build the Libraries and IP required and building the hardware test libraries for the hardware and simulation tests were done.

Then environment setup for building the project was completed and reference-SDNSwitch project was build. building process will run through synthesis, implementation, place and route, and bit-file generation using the Xilinx tools.When the process is completed, the bit-stream can be downloaded to the board. Downloading process configure the FPGA by sending the bit-stream. But In this scenario the Net-FPGA was not detected by the System. Best reason could be the motherboard discompatibilities. Used motherboard was P4 motherboard and required one was ASUS M5A99FX PRO R2.0 motherboard. System tested on Fedora 20, CentOS 6, Ubuntu 16.04 with 2.8, 3.14 and 3.2 kernal versions. But the problem was not solved. Then building the SDN network was could not completed in expected time period. Then we moved to Virtual network implementation with use of OpenVSwitch as the SDN switch.

4.2 Select most preferable SDN controller

First,Virtual SDN network was build using Mininet Emulator.Selected SDN controllers were OpenDayLight, FloodLight ONOS and Ryu.All these controllers used to manipulate the network that build on Mininet. Also the Rest APIs. According to the results based on scalability, stability, intreaction of the end user and pluggin return language .OpenDayLight[6] was selected as the most preferable controller.

4.3 Build the simple SDN network

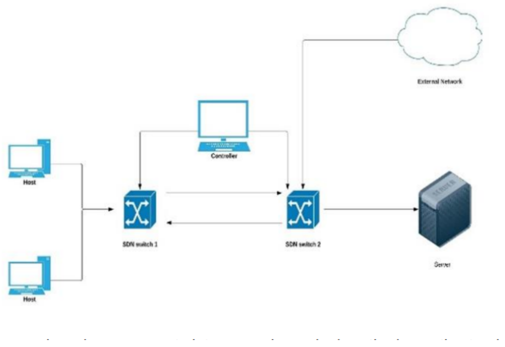


Fig. 4.2 Built SDN network

Ubuntu VM was created using Oracle virtualbox. five VM were created by cloning it.two VMs were used to as host , One for Server and another two separated VM were used to SDN switches.

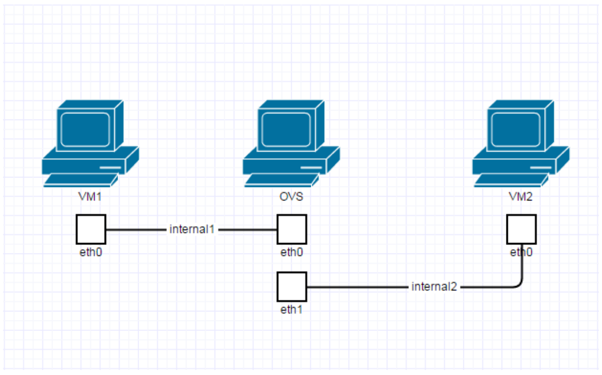
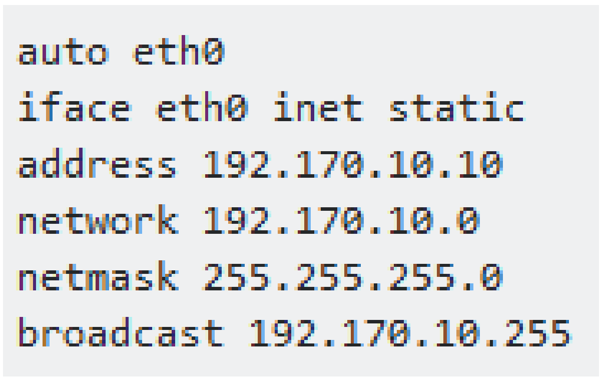


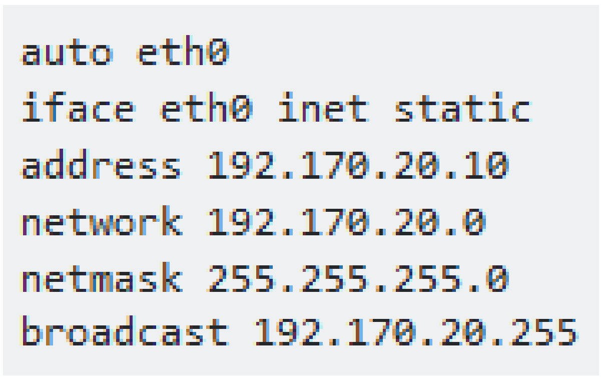
Fig. 4.3 Connection between VMs

Below steps show the how connections are made between VMs.
interfaces configured in /etc/network/interfaces file. following line ware added.



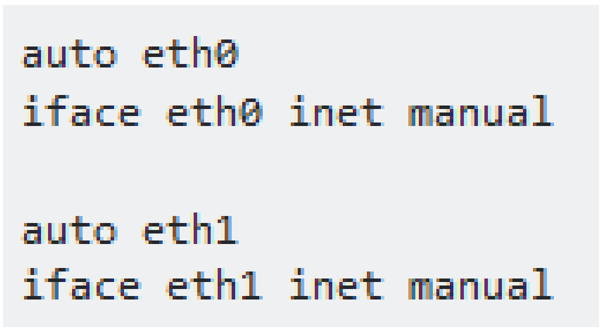
```
auto eth0
iface eth0 inet static
address 192.170.10.10
network 192.170.10.0
netmask 255.255.255.0
broadcast 192.170.10.255
```

Fig. 4.4 Configuration of VM1



```
auto eth0
iface eth0 inet static
address 192.170.20.10
network 192.170.20.0
netmask 255.255.255.0
broadcast 192.170.20.255
```

Fig. 4.5 Configuration of VM2



```
auto eth0
iface eth0 inet manual

auto eth1
iface eth1 inet manual
```

Fig. 4.6 Interface configuration of Switch

Switch configuration did using terminal commands as followings,

First two switch bridges ware created (br0 and br1) and respective eth1 and eth2 interfaces connected to bridges. Then the bridges ware connected using patch interfaces which can created using the **ovs-vsctl** utility. After that we can bring up the bridges and set respective IP addresses.Those bridges can connect to SDN controller using below command

ovs-vsctl set-controller bridge-name tcp:ip-address:port

This procedure was applied to all host VMs with suitable IP addresses.For configuration OpenvSwitch above described step were used. But this virtual network required large amount of RAM when all the VM up and in functional state our computers got strucked and lagged. Then for demonstration purposes Mininet based virtual network was used.

4.4 Develop an application for analyzing network statistics and packet visualization.

In application level first, Stored data of network that received from SDN controller was written on the JSON file.Java application was used to process the JSON and extract the required data and push in to the MySQL database.This happened in pre-scheduled time periods automatically.

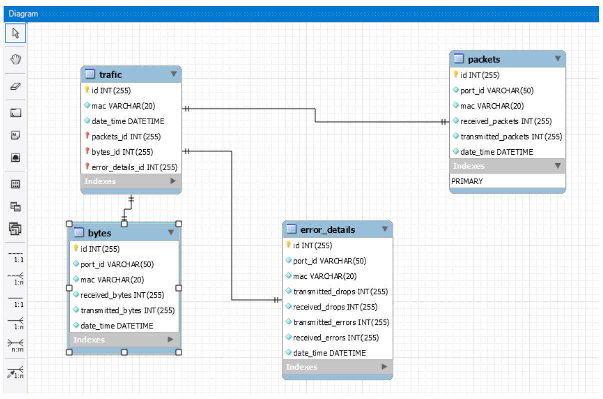
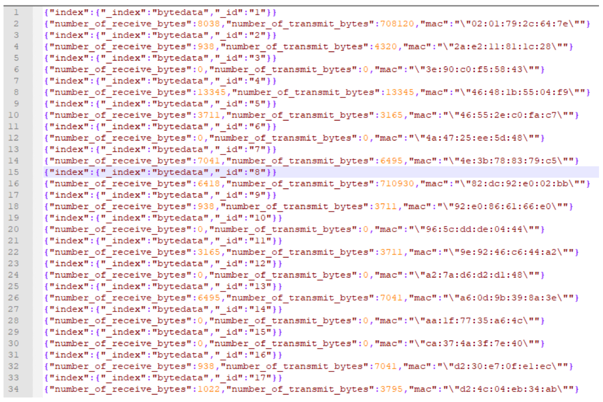


Fig. 4.7 ER-Digram for Database

Another Java application gets data from the MySQL database and process them to be compatible with the ELK Stack input. Generated file pushed in to the ELK stack database using bellow command

Invoke-RestMethod "http://localhost:9200/bytedata/doc/bulk?pretty" -Method Post -ContentType 'application/x-ndjson' -InFile "bytedata.json"

After, created an index pattern to handle uploaded data.Using that index pattern the visualization process can handle. Ghraphs ,tables , Bar charts and Geometrical views are some of representing forms available in Kibana.Also filters and advance search, analyzing options can add to this.



```
{"index":{"_index":"bytedata","_id":"1"}}
{"number_of_receive_bytes":8038,"number_of_transmit_bytes":708120,"mac":"\"02:01:79:2c:64:7e\""}
{"index":{"_index":"bytedata","_id":"2"}}
{"number_of_receive_bytes":938,"number_of_transmit_bytes":4320,"mac":"\"2a:e2:11:81:1c:28\""}
{"index":{"_index":"bytedata","_id":"3"}}
{"number_of_receive_bytes":0,"number_of_transmit_bytes":0,"mac":"\"3e:90:c0:f5:58:43\""}
{"index":{"_index":"bytedata","_id":"4"}}
{"number_of_receive_bytes":13345,"number_of_transmit_bytes":13345,"mac":"\"46:48:1b:55:04:f9\""}
{"index":{"_index":"bytedata","_id":"5"}}
{"number_of_receive_bytes":3711,"number_of_transmit_bytes":3165,"mac":"\"46:55:2e:c0:fa:c7\""}
{"index":{"_index":"bytedata","_id":"6"}}
{"number_of_receive_bytes":0,"number_of_transmit_bytes":0,"mac":"\"4a:47:25:ee:5d:48\""}
{"index":{"_index":"bytedata","_id":"7"}}
{"number_of_receive_bytes":7041,"number_of_transmit_bytes":6495,"mac":"\"4e:3b:78:83:79:c5\""}
{"index":{"_index":"bytedata","_id":"8"}}
{"number_of_receive_bytes":6418,"number_of_transmit_bytes":710930,"mac":"\"82:dc:92:e0:02:bb\""}
{"index":{"_index":"bytedata","_id":"9"}}
{"number_of_receive_bytes":938,"number_of_transmit_bytes":3711,"mac":"\"92:e0:86:61:66:e0\""}
{"index":{"_index":"bytedata","_id":"10"}}
{"number_of_receive_bytes":0,"number_of_transmit_bytes":0,"mac":"\"96:5c:dd:de:04:44\""}
{"index":{"_index":"bytedata","_id":"11"}}
{"number_of_receive_bytes":3165,"number_of_transmit_bytes":3711,"mac":"\"9e:92:46:c6:44:a2\""}
{"index":{"_index":"bytedata","_id":"12"}}
{"number_of_receive_bytes":0,"number_of_transmit_bytes":0,"mac":"\"a2:7a:d6:d2:d1:48\""}
{"index":{"_index":"bytedata","_id":"13"}}
{"number_of_receive_bytes":6495,"number_of_transmit_bytes":7041,"mac":"\"a6:0d:9b:39:8a:3e\""}
{"index":{"_index":"bytedata","_id":"14"}}
{"number_of_receive_bytes":0,"number_of_transmit_bytes":0,"mac":"\"aa:1f:77:35:a6:4c\""}
{"index":{"_index":"bytedata","_id":"15"}}
{"number_of_receive_bytes":0,"number_of_transmit_bytes":0,"mac":"\"ca:37:4a:3f:7e:40\""}
{"index":{"_index":"bytedata","_id":"16"}}
{"number_of_receive_bytes":938,"number_of_transmit_bytes":7041,"mac":"\"d2:30:e7:0f:e1:ec\""}
{"index":{"_index":"bytedata","_id":"17"}}
{"number_of_receive_bytes":1022,"number_of_transmit_bytes":3795,"mac":"\"d2:4c:04:eb:34:ab\""}
```

Fig. 4.8 JSON File format

Chapter 5

Results and Analysis

Network data were collected and stored into MySQL database in real time.As above described all data and informations collect by pre-scheduled time period. These data processed before storing in the database to extract the necessary data. Retrieving data from the database analyzed and segmented using an algorithm.

5.1 Results

Application shows the network statistics of

- Inbound and Outbound traffic of each ports of switches.
- Received bytes and transmitted bytes statics
- Received and Transmitted drops statics
- Received and Transmitted packets errors statics
- Received and Transmitted packets CRC errors statics

this all statics data get as raw input data. then process those data for compatible with Kibana.

5.2 Result Analysis

5.2.1 Inbound traffic of the network

This graph show the variance of the inbound traffic of the network against the MAC address of the each port.

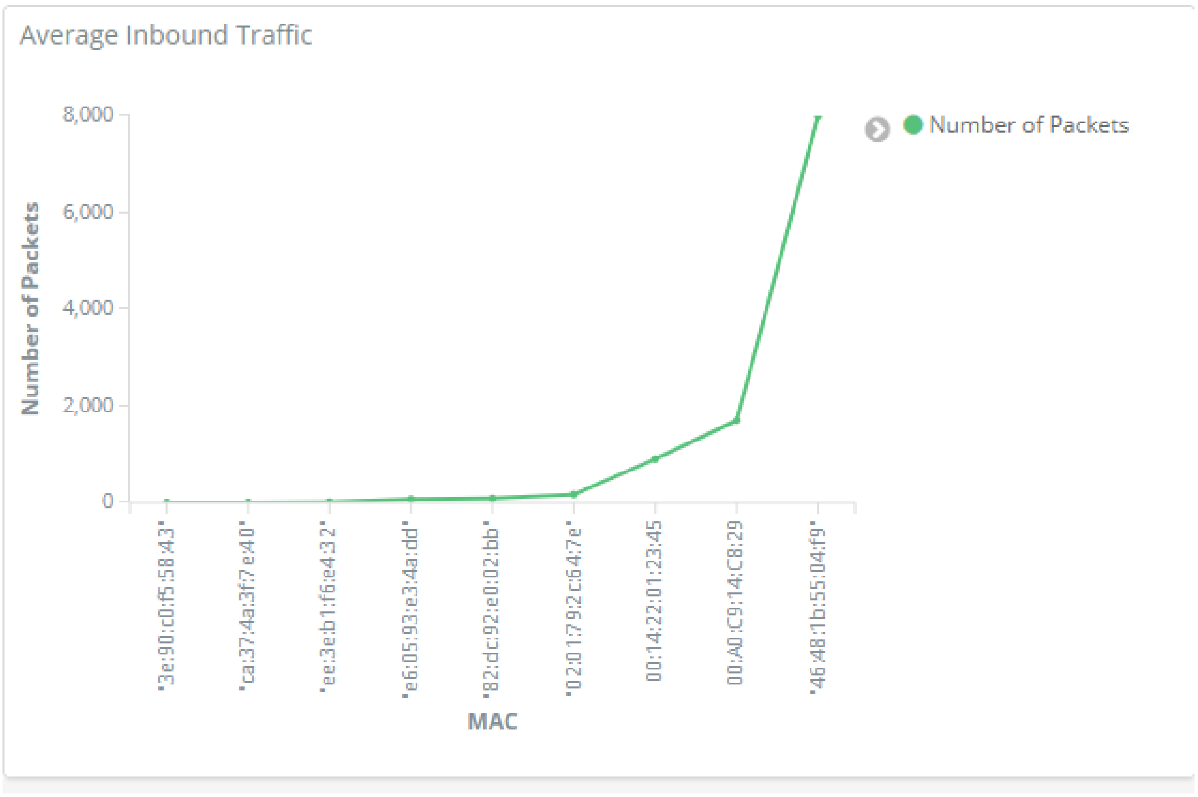


Fig. 5.1 Inbound traffic of the network

5.2.2 Outbound traffic of the network

This graph show the variance of the inbound traffic of the network against the MAC address of the each port.

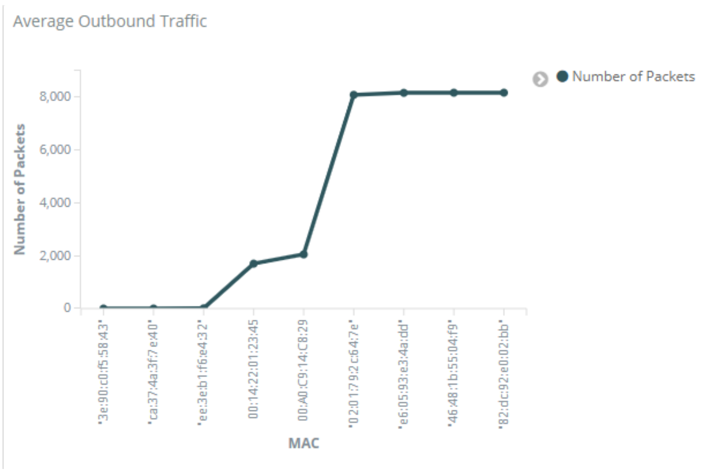


Fig. 5.2 outbound traffic of the network

5.2.3 Received and Transmitted bytes of the network

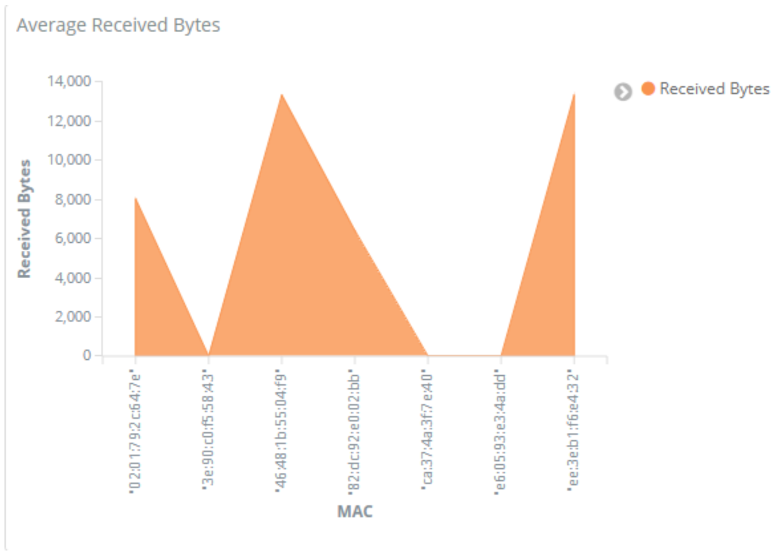


Fig. 5.3 Received bytes of the network

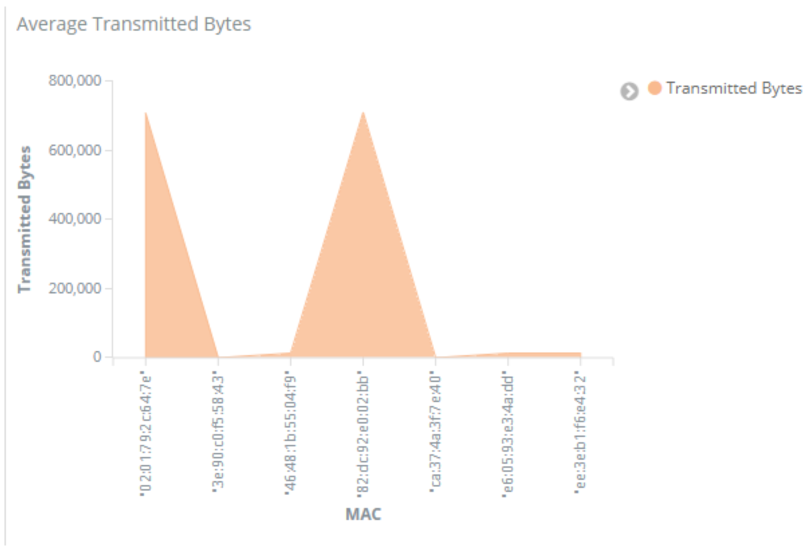


Fig. 5.4 Transmitted bytes of the network

5.2.4 Packet drops and errors of the network

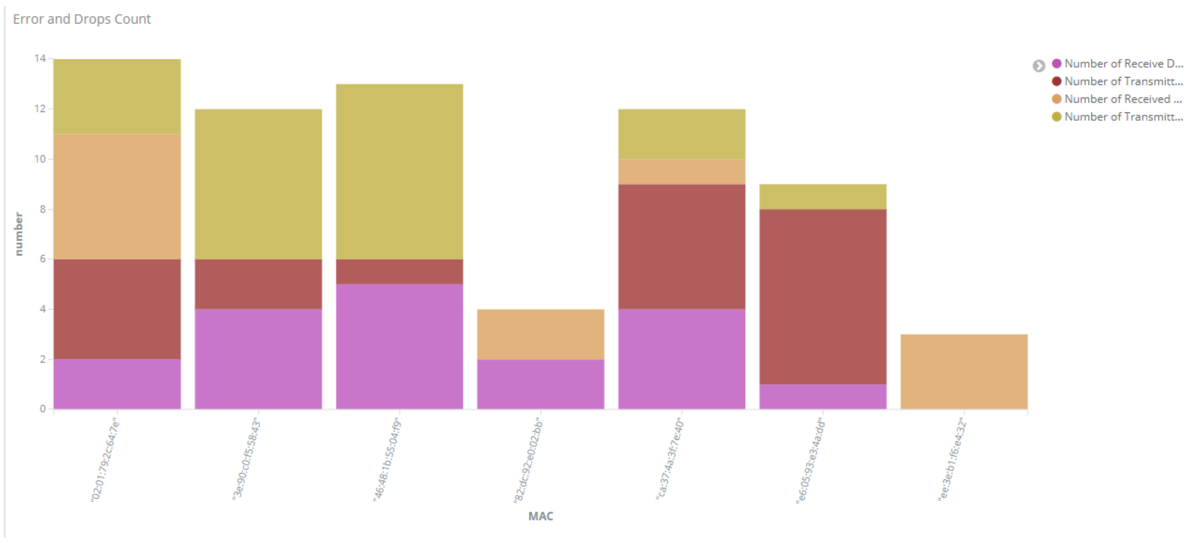


Fig. 5.5 Packet drops and errors of the network

Chapter 6

Conclusions and Future Work

6.1 Conclusion

SDN is a new field of research which rapidly growing. There are many interests and opportunities ahead in SDN. Leading university students toward that side will make a huge difference. To fulfill that intention, the first bottleneck is infrastructures to do experimentations. Also understanding core concepts of SDN paradigm properly is important. Instead of depend only on theoretical knowledge of SDN, Hands on experience and practical knowledge will helps to build well knowledged SDN experts. Building a platform to gain practical and theoretical knowledge will be a major step to that. The low cost SDN testbed can be used for research and educational purposes. Also when designing such a testbed it is necessary to consider how mush student oriented and user friendly it is. Student should be able to expand the network also. Integrated network with virtual and physical networks will be scalable in to more complex network with cost effectiveness. This approach will open new design of low cost SDN network with much more capabilities. As opposed to the previous projects done in this area, we provide complete methodology and documentation for building a similar SDN testbed for interested third parties

6.2 Future works

- Packet visualization is under development
- Simulations of advance SDN concepts.
- Combining physical and virtual network

- Deploy the SDN testbed and testing

References

[1] H. Kim, J. Kim, and Y.-B. Ko, “Developing a Cost-Effective OpenFlow Testbed for Small-Scale Software Defined Networking,” tech. rep.

[2] W. Jlmn and C. Njaw, “SDN Testbed for Undergraduate Education,” tech. rep.

[3] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, “Implementing an OpenFlow switch on the NetFPGA platform,” in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems - ANCS '08*, 2008.

[4] F. Keti and S. Askar, “Emulation of Software Defined Networks Using Mininet in Different Simulation Environments,” in *Proceedings - International Conference on Intelligent Systems, Modelling and Simulation, ISMS*, 2015.

[5] C. Decusatis, A. Carranza, and J. Delgado-Caceres, “Modeling Software Defined Networks using Mininet,” Tech. Rep. 133.

[6] V. R. Sudarsana Raju, “SDN CONTROLLERS COMPARISON,” tech. rep., 2018.