

Software-defined Networking and Its Implementation By using Mininet

- | | |
|------------------------|-------------------|
| 1. Kaniz Shormin Kushi | ID- 2014-3-55-024 |
| 2. Tasnia Enayet | ID- 2015-1-55-002 |
| 3. Oishe Ahmed Dipa | ID- 2015-1-55-029 |

Under Supervision of

Dr. Mohammad Arifuzzaman

Assistant Professor

Department of Electronics and Communications Engineering

East West University

**A thesis submitted in partial fulfillment of the requirements for the degree
of Bachelor of Science in Electronics and Telecommunications Engineering**



Department of Electronics and Communications Engineering

EAST WEST UNIVERSITY

Dhaka-1212, Bangladesh

Fall Semester 2018-2019

April 2019

Declaration

I, hereby, declare that the work presented in this thesis is the outcome of the investigation performed by me under the supervision of Dr. Mohammad Arifuzzaman, Assistant Professor, Department of Electronics and Communications Engineering, East West University. I also declare that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

Signature

.....
(Dr. Mohammad Arifuzzaman)
Supervisor

.....
(Kaniz Shormin Kushi)
(2014-3-55-024)

Signature

.....
(Tasnia Enayet)
(2015-1-55-002)

Signature

.....
(Oishe Ahmed Dipa)
(2015-1-55-029)

Letter of Acceptance

This thesis report entitled “*Software-defined Networking & its Implementation using Mininet*” submitted by Kaniz Shormin Kushi (ID:2014-3-55-024), Tasnia Enayet (ID: 2015-1-55-002) and Oishe Ahmed Dipa (ID: 2015-1-55-029) to the Department of Electronics and Communications Engineering, East West University is Accepted by the department in partial fulfillment of requirements for the award of the Degree of Bachelor of Science in Electronics and Telecommunications Engineering on April, 2019.

Supervisor

.....
(Dr. Mohammad Arifuzzaman)
Assistant Professor,
Department of Electronics and Communications Engineering, East West University

Chairperson

.....
(Dr. Mohammed Moseur Rahman)
Assistant Professor and Chairperson,
Department of Electronics and Communications Engineering, East West University

Abstract

Software Defined Networking (SDN) is an evolving networking technology concept that enables advancement about how to maintain and design network systems. Due to the separate data plane and control plane, consolidated management is assisted in a Software defined Network. In this thesis we did a compulsive study on SDN.

Another important term is OpenFlow which is an SDN approach, presenting the inherent OpenFlow switches with such a centralized controller. OpenFlow switches are designed with flow tables that the controller can handle to use the OpenFlow protocol interface. An evaluation of an SDN emulation tool called Mininet is performed in this thesis. In Software Defined Networking (SDN) and OpenFlow, Mininet had been created to enable research. In this thesis we have used Mininet network emulator to create a virtual SDN network for OpenDaylight to control. A tree topology is created over OpenDaylight using mininet, which is a simple demonstration of SDN and understandable to everyone.

Acknowledgments

As it is true for everyone, we have also arrived at this point of achieving a goal in our life through various interactions with and help from other people. However, written words are often elusive and harbor diverse interpretations even in one's mother language. Therefore, we would not like to make efforts to find best words to express my thankfulness other than simply listing those people who have contributed to this thesis its elfin an essential way. This work was carried out in the Department of Electronics & Communication Engineering at East West University, Bangladesh.

First of all, we would like to express my deepest gratitude to the almighty for His blessings on us. Next, our special thanks go to our supervisor, Dr. Mohammad Arifuzzaman, who gave us this opportunity, initiated us into the field of networking, and without him this work would not have been possible. His encouragements, visionaries and thoughtful comments and suggestions, unforgettable support at every stage of our B.Sc. study were simply appreciating and essential. His ability to muddle us enough to finally answer our own question correctly is something valuable what we have learned and we would try to emulate, if ever I/We get the opportunity.

Last but not the least; we would like to thank our parents for their unending support, encouragement and prayers.

There are numerous other people to who have shown us their constant support and friendship in various ways, directly or indirectly related to our academic life. One of them is Mohaiminul Islam Shovon. We will remember them in our heart and hope to find a more appropriate place to acknowledge them in the future.

Kaniz Shormin Kushi

April, 2019

Tasnia Enayet

April, 2019

Oishe Ahmed Dipa

April, 2019

Table of Content

Chapter 1 Introduction	1
1.1: Software-defined networking(SDN).....	1
1.1.2: Differences between Traditional Networking and SDN.....	2
1.1.3: Importance of SDN	2
1.2: OpenFlow	3
1.2.1: OpenFlow working operation.....	4
1.2.2: Necessary of OpenFlow.....	4
1.3: The Minnet.....	5
Chapter 2 Literature Review	6
2.1 Introduction.....	6
2.2 History of SDN	6
2.3 Existing Works On Software-defined Networking.....	6
Chapter 3 Components and Architecture of SDN	8
3.1: Infrastructure Layer (Data plane).....	9
3.2: Controller Plane	12
3.2.1: SDN controller functional components	14
3.2.1.1: Data plane control function.....	15
3.2.1.2: Co-ordinator.....	15
3.2.1.3: Virtualizer.....	15
3.2.1.4: Agent.....	16
3.2.1.5: Other controller component.....	16

3.2.2: Legation of control.....	17
3.3: Application layer.....	17
3.4: Management.....	19
Chapter 4 Openflow Basics	20
4.1: Openflow.....	20
4.2: A Brief of OpenFlow SDN	20
4.3: OpenFlow and OpenFlow Switch	20
4.4: OpenFlow Switch Process System.....	21
4.5: Differences of OpenFlow Switch v Conventional Switch.....	21
4.6: State of the Art of Open Switch.....	21
Chapter 5 Mininet	23
5.1: Mininet	24
5.2: Future Work.....	34
Chapter 6 SDN Tree Topology Implementation using Mininet	25
6.1: Introduction.....	25
6.2: Simulation.....	27
6.3: Other Simulators.....	37
Chapter 7 Future plan of SDN	38
7.1: SDN Migration plan.....	38
7.2: SDN versus Conventional Systems Administration (Traditional Networking).....	39

7.3: SDN Adoption.....	39
7.4: Virtual-Network peering.....	40
7.5: Virtual-Network encryption.....	40
Conclusion	41

List of Figure

Figure 1: three-layer software-defined networking (SDN) architecture	1
Figure 2: Basic packet forwarding with OpenFlow in a switch	4
Figure 3: Basic SDN Components	8
Figure 4: Network Element (NE) recourse details	9
Figure 5: Architecture of plane, layers and architecture design	10
Figure 6: Openflow Qualify SDN	11
Figure 7: SDN Control logic	12
Figure 8: Open flow and open flow switch communicates over OpenFlow channel to an external Controller.....	21
Figure 9: Screenshot of configuring IP address of OpenDaylight	27
Figure 10: Screenshot of Ip address of OpenDaylight	28
Figure 11: Screenshot of starting the OpenDaylight	30
Figure 12: Screenshot of IP address of Mininet	31
Figure 13: Screenshot of configuring the Tree topology command	32
Figure 14: Screenshot of testing the all connections	33
Figure 15: Screenshot of page layout of OpenDaylight	34
Figure 16: Screenshot of creating the tree topology	34
Figure 17: Screenshot of node connections of the tree	35
Figure 18: Screen shot of PuTTY configuration	36
Figure 19: Screenshot wireshark that capture the data traffics	37

Chapter 1

Introduction

1.1 Software-defined networking (SDN)

Software-defined networking (SDN) raised a great deal in present-day since it tends to the insufficiency of programmability in existing systems administration structures and empowers less demanding and quicker system development.

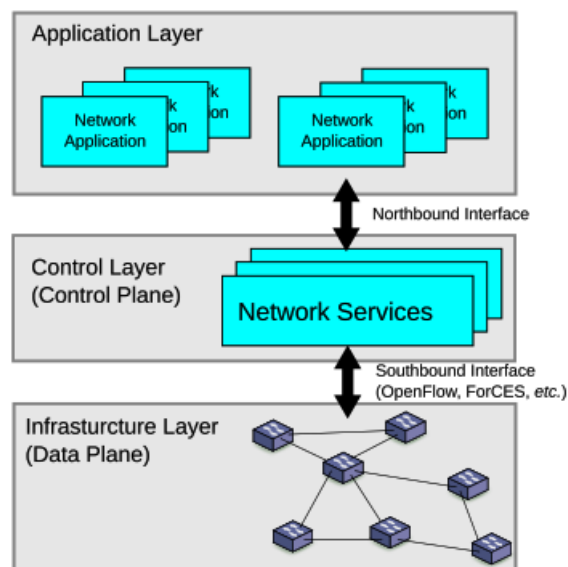


Figure 1.1: A three-layer software-defined networking (SDN) architecture. [1].

Figure 1 delineates the SDN structure, which comprises of three layers. The least layer is the foundation layer, additionally called the information plane. It includes the sending system components. The obligations of the sending plane are mostly information sending, just as observing neighborhood data and social event measurements. One layer above, we find the control layer, additionally called the control plane. It is in charge of programming and dealing with the sending plane. The application layer contains arrange applications that can present new system highlights, for example, security and sensibility, sending plans or help the control layer in the system configuration. [1]

1.1.2 Differences between Traditional Networking and SDN:

Following table describes difference between traditional and software defined networking types.

Table 1: Difference between Traditional Networking and SDN

Traditional Networking	Software Defined Networking
They are Static and inflexible networks. They are not useful for new business ventures. They possess little agility and flexibility	They are programmable networks during deployment time as well as a later stage based on a change in the requirements. They help new business ventures through flexibility, agility and virtualization.
They are Hardware appliances.	They are configured using open software.
They have distributed control plane.	They have a logically centralized control plane.
They use custom ASICs and FPGAs.	They use merchant silicon.
They work using protocols.	They use APIs to configure as per need.

1.1.3 Importance of SDN:

With the end goal for SDN to convey on its full guarantee, it must be empowered by open systems administration guidelines that can be effectively coordinated with current frameworks. Receiving a SDN approach has a bunch of advantages including adaptability, versatility, repetition, and execution. In a customary system, there may be sure restricted equipment and programming pieces. At the point when a system requires extra assets, there will be significant expense in purchasing new equipment and authorizing.

With SDN, the system is disconnected onto programming, leaving progressively decision and adaptability in obtaining equipment. Also, a developing system can be all the more effectively upheld by SDN on the grounds that a system overseer or designer can basically include more virtual switches or switches as opposed to buy expensive hardware and authorizing.

A product characterized organization is likewise convenient, which permits the adaptability in picking and moving to distributed storage, open or private. Abstracting your system onto a cloud could exhibit numerous advantages too: less equipment to oversee nearby, lower vitality bills, and more noteworthy uptime.

1.2 OpenFlow

An OpenFlow Controller is a product application that oversees stream control in a SDN domain. As a rule, numerous SDN controllers depend on the OpenFlow convention. All correspondences among applications and gadgets legitimately experience the controller. The OpenFlow convention associates the controller programming to organize gadgets with the goal that server programming can advise changes where to send parcels for the sending table. Thusly, the controller utilizes the OpenFlow convention to arrange organize gadgets to pick the best way for application traffic. [2]

OpenFlow design comprises of three fundamental ideas.

1. The system is developed by OpenFlow-agreeable switches that create the information plan
2. The control plane comprises of at least one OpenFlow controllers
3. A secure control channel interfaces the switches with the control plane.

Three classes of correspondence exist in the OpenFlow convention: controller-to-switch, non-concurrent and symmetric correspondence. The controller-to-switch correspondence is in charge of highlight identification, configuration, programming the switch and data recovery. Offbeat correspondence is started by the OpenFlow-consistent switch with no requesting from the controller. It is utilized to advise the controller about bundle landings, state changes at the switch and mistakes. At last, symmetric messages are sent without sales from either side, i.e., the switch or the controller is allowed to start the correspondence without sales from the opposite side. Instances of symmetric correspondence are hi or reverberation messages that can be utilized to distinguish whether the control channel is still live and accessible.

1.2.1 OpenFlow working operation

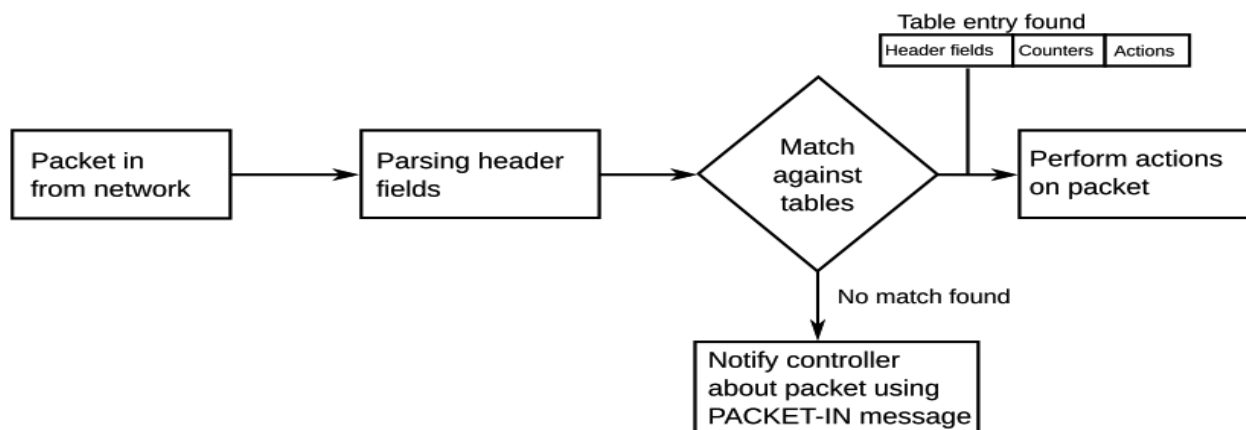


Figure1.2: Basic packet forwarding with OpenFlow in a switch [1].

The essential bundle sending component with OpenFlow is outlined in Figure3. At the point when a switch gets a bundle, it parses the parcel header, which is coordinated against the flow table. On the off chance that a flow table passage is discovered where the header field special case coordinates the header, the section is considered. On the off chance that few such passages are discovered, bundles are coordinated dependent on prioritization, i.e., the most specific section or the trump card with the most astounding need is chosen. At that point, the switch refreshes the counters of that flow table passage. At last, the switch plays out the activities specified by the flow table section on the parcel, e.g., the switch advances the bundle to a port. Something else, if no flow table section coordinates the parcel header, switch by and large notifies its controller about the bundle, which is cradled when the switch is fit for buffering. Keeping that in mind, it embodies either the unbuffered bundle or the first bytes of the cushioned parcel utilizing a PACKET-IN message and sends it to the controller; usually to exemplify the bundle header and the quantity of bytes defaults to 128. The controller that gets the PACKET-IN notification identifies the right activity for the parcel and introduces at least one fitting sections in the mentioning switch. Supported bundles are then sent by the guidelines; this is activated by setting the cradle ID in the flow inclusion message or in unequivocal PACKET-OUT messages. Most generally, the controller sets up the entire way for the bundle in the system by changing the flow tables of all switches on the way

1.2.2 Necessity of OpenFlow

Merchants offer differing degrees of client programmability on their switches and switches. This can prompt constrained usefulness for traffic building and the executives or conflicting traffic the board between hardware from different merchants. OpenFlow is intended to give consistency in rush hour gridlock the executives and building by making this control work free of the equipment it's proposed to control. [3]

1.3 The Mininet

The worldview SDN is as yet later in this way the system explores have concentrated their own investigations of the subject. At the point when those scientists need to test the new SDN includes in the controllers, switches or even in the OpenFlow convention, they have a few troubles. Those challenges happen extraordinarily in light of the fact that there are so couple of modest gadgets accessible that can execute in SDN standard. In addition, in increasingly explicit cases, when it is important to reproduce substantial systems with extensive quantities of hosts, switches and SDN controllers, utilizing the Internet may not be a smart thought, in light of the fact that ill-advised setups can cause undesirable issues. One of the answers for this issue is making models and reproducing them in virtual mode. To do this, a few apparatuses have been made and one of them is the Mininet programming [8]. The Mininet is a framework that permits quickly prototyping expansive systems on a solitary PC. It makes versatile Programming characterized systems utilizing lightweight virtualization instruments, for example, procedures and system namespaces. These highlights license the Mininet make, communicate with, redo and share the models rapidly.

A few attributes guided the formation of Mininet are

- 1) Flexibility, that is, new topologies and new highlights can be set in programming, utilizing programming dialects and basic working frameworks;
- 2) Applicability, effectively executions done in models ought to be likewise usable in genuine systems dependent on equipment with no adjustments in source codes;
- 3) Interactivity, the executives and running the reproduced system must happen progressively as though it occurs in genuine systems;
- 4) Scalability, the prototyping condition must scale huge systems with hundreds or thousands of switches on just a PC;
- 5) Realistic, the model conduct ought to speak to constant conduct with a high level of certainty, so applications and conventions stacks ought to be usable with no code alteration; lastly
- 6) share-capable, the made models ought to be effectively imparted to different teammates, which would then be able to run and change the trials.

Chapter 2

Literature Review

2.1 Introduction

Though SDN is a new topic but a good number work already done on SDN. This section additionally talked about existing works on Software-defined Networking.

2.2 History of SDN

The records of SDN concepts can be traced lower back to separation of the control and data plane first used in the public switched telephone networks as a way to simplify provisioning and management properly earlier than this structure started to be used in data networks. The open networking Foundation was founded in 2011 to promote SDN and OpenFlow. At 2014 Interop and Tech field Day, software-defined networking was demonstrated by Avaya using shortest path bridging (IEEE 802.1aq) and OpenStack as an automated campus, extending automation from the data center to the end device, removing manual provisioning from service delivery. [10, 11]

2.3 Existing Works On Software-defined Networking

Many researchers and scholars have done works using different tools and methods of software-defined networking. In this section of the paper, we discussed about some existing works already done by various researchers.

In [12] authors have focused on to make researches capable to do experiments and to take a look at novel features of this new paradigm in practice at a low financial cost, and to use virtual community emulators. This paper focuses on find out about and contrast of SDN emulation tool called Mininet. Also its elements with working principles, some net prototypes are created to better understand the Mininet toll and a distinguished its advantages and disadvantages

In [13] authors have proposed an innovative SDN architecture and their aim is to execute SDN applications written for different controller in a unique network where the structure presents an important trouble to debug and analyze the SDN network. Some set of tool is layout and developed with the motive of solving this issue and assurance the appropriate operation of the network.

In [14] authors describe a way to bring the Software-defined Network paradigm into Wireless Local Area Networks. The control and management functions that need to be implemented in the wireless scenario are described. The thesis provides a quantitative measure of the gain attained in various network parameters such as throughput and wireless station setup time by bringing in the SDN paradigm, via simulations.

In [15] authors compared different mininet topologies and used one of the topologies to find out how host communicates in the mininet environment, as well as using the wget linux feature to make this possible.

In [16] authors evaluate Mininet's scalability in terms of creating many topologies is tested with varying number of nodes and two different environment scenarios where results show simulation environment creates impressive effect on required time to construct a topology.

Chapter-3

Components and Architecture of SDN

A software-defined networking (SDN) architecture (or SDN architecture) defines how a networking and computing machine can be constructed the use of an aggregate of open, software-based applied sciences and commodity networking hardware that separate the SDN manipulate plane and the SDN statistics plane of the networking stack.

Traditionally, each the SDN manage data plane and facts data plane factors of a networking architecture had been packaged in proprietary, built-in code disbursed by means of one or a mixture of proprietary vendors. The OpenFlow standard, created in 2008, used to be identified as the first SDN architecture that defined how the manage and facts plane elements would be separated and communicate with every other the usage of the OpenFlow protocol. The Open Network Foundation (ONF) is the body in cost of managing OpenFlow standards, which are open source. However, there are other requirements and open-source agencies with SDN resources, so OpenFlow is now not the only protocol that makes up SDN. Each layer has its own precise functions. While some of them are always present in an SDN deployment, such as the southbound API, NOSs, northbound API, and network applications, others might also be present only in precise deployments, such as hypervisor- or language-based virtualization. The following sections introduce each layer, following a bottom-up approach. For every layer, the core properties and ideas are defined primarily based on the special technologies and solutions. Additionally, debugging and troubleshooting strategies and equipment are discussed. [17]

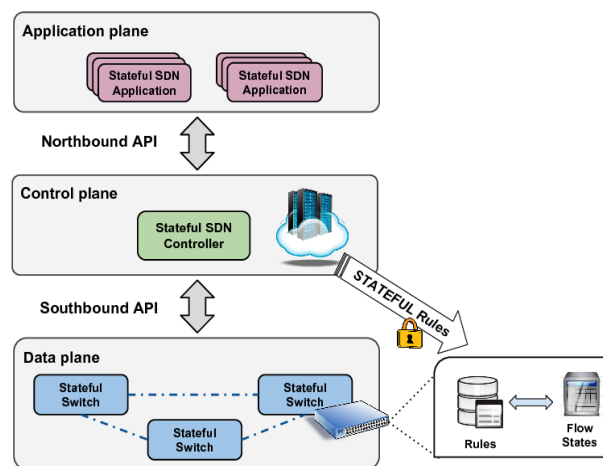


Figure3.1: Basic SDN Components [17].

Figure 3.1 introduces the simple SDN components, with terminology similar to that from the unique ONF white paper, “Software-Defined Networking: The New Norm for Networks”. The initial view comprised infrastructure, control and utility layers (red text), which are certain in this architecture report as data, controller, and utility planes (black text). The infrastructure layer (data

plane, note) includes community elements, which expose their skills towards the manipulate layer (controller plane) by using interfaces southbound from the controller. This is known as a control-data plane interface.) The SDN applications exist in the utility layer (plane), and talk their network necessities toward the controller airplane by northbound interfaces, frequently referred to as NBIs. In the middle, the SDN controller translates the applications' requirements and exerts low-level manipulate over the network elements, while presenting applicable records up to the SDN applications. An SDN controller may orchestrate competing software demands for restricted network sources in accordance to policy. [17]

3.1 Infrastructure Layer (Data plane)

The data plane incorporates the resources that deal immediately with customer traffic, alongside with the quintessential help in gussets to make certain suitable virtualization, connectivity, security, availability, and quality. The NE assets block consists of data sources, data sinks and forwarding and/or visitors processing engines, as properly as a virtualizer whose function is to abstract the sources to the SDN controller and put in force policy. This enlargement of element additionally introduces a master aid facts base (RDB), the conceptual repository of all resource facts recognized to the community element.

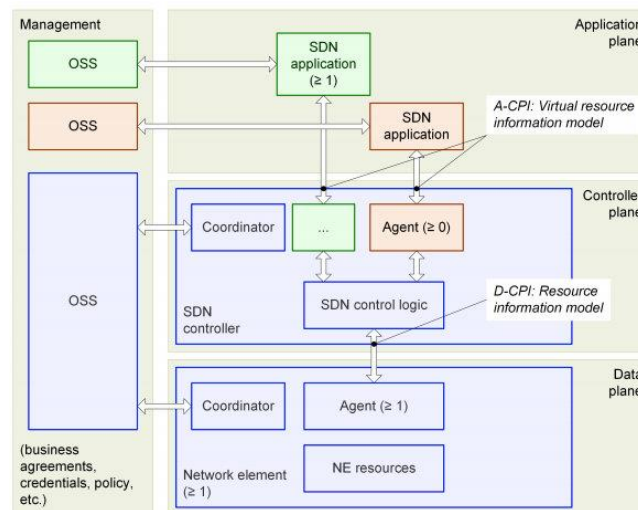


Figure 3.2: Network Element (NE) resources details [18].

Software-defined networking concerns itself with visitors forwarding and visitors processing function such as QoS, filtering, monitoring, or tapping. Traffic may additionally enter or leave the SDN data plane by using physical or logical ports, and might also be directed into or out of forwarding or processing functions. Traffic processing would possibly be exemplified by using an

OAM engine, an encryption function, or a virtualized network function. Control of traffic forwarding or processing features can also be carried out by an SDN controller or with the aid of separate mechanisms, maybe orchestrated in conjunction with the given SDN controller. The statistics data plane implements forwarding choices made in the controller plane. In principle, it does not make self-sustaining forwarding decisions. [18] However, the controller data plane may configure the facts data plane to reply autonomously to activities such as network screw-ups or to guide functions delivered by, for example, LLDP, STP, BFD, or ICMP. The interface between facts and controller planes (D-CPI) consists of features such as

- Programmatic manipulate of all features exposed by way of the RDB
- Capabilities advertisement
- Event notification

The data plane agent is the entity that executes the SDN controller's directions in the facts plane. The information plane coordinator is the entity through which management allocates information data plane resources to a range of client marketers and establishes policy to govern their use. Agents and coordinators serve the equal cause in every plane of the architecture. An SDN infrastructure, in a similar fashion to an ordinary network, is composed of a set of networking equipment (switches, routers, and middle box appliances). The predominant difference resides in the fact that these ordinary bodily gadgets are now easy forwarding elements besides embedded manage or software to take self-reliant decisions. The network Genius is removed from the data plane devices to a logically centralized manipulate system, i.e., the NOS and applications, as shown in Fig. 3.3. More importantly, these new networks are constructed (conceptually) on top of open and standard interfaces (e.g., OpenFlow), a crucial approach for ensuring configuration and communication compatibility and interoperability amongst unique data and manage plane devices.

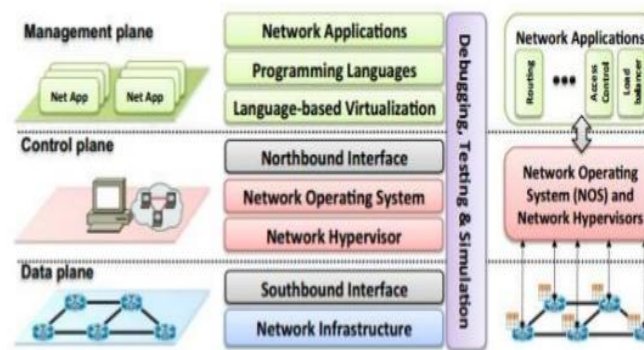


Figure 3.3: Architecture of plane, layers and architecture design [19].

In different words, these open interfaces enable controller entities to dynamically software heterogeneous forwarding devices, something hard in standard networks, due to the large range of proprietary and closed interfaces and the disbursed nature of the control plane. In an SDN/OpenFlow architecture, there are two main elements, the controllers and the forwarding devices, as proven in Fig. 2.4. A facts plane system is a hardware or software element specialized

in packet forwarding, whilst a controller is a software program stack (the “network brain”) jogging on a commodity hardware platform. An OpenFlow enabled forwarding system is based on a pipeline of drift tables the place every entry of a float desk has three parts: 1) a matching rule; 2) moves to be carried out on matching packets; and 3) counters that preserve records of matching packets. This excessive stage and simplified mannequin derived from OpenFlow is currently the vastest layout of SDN records airplane devices. Nevertheless, different specs of SDN-enabled forwarding units are being pursued, including POF and the negotiable data path models (NDMs) from the ONF Forwarding Abstractions Working Group (FAWG). Inside an OpenFlow device, a course thru a sequence of glide tables defines how packets ought to be handled. When a new packet arrives, the look up process begins in the first table and ends both with a in shape in one of the tables of the pipeline or with a leave out (when no rule is observed for that packet). A waft rule can be defined via combining one of a kind matching fields, as illustrated in Fig. 3.4. If there is no default rule, the packet will be discarded. However, the common case is to installation a default rule which tells the change to send the packet to the controller (or to the ordinary non-OpenFlow pipeline of the switch). The precedence of the rules follows the herbal sequence quantity of the tables and the row order in a waft table. Possible moves include: forward the packet to outgoing port(s)

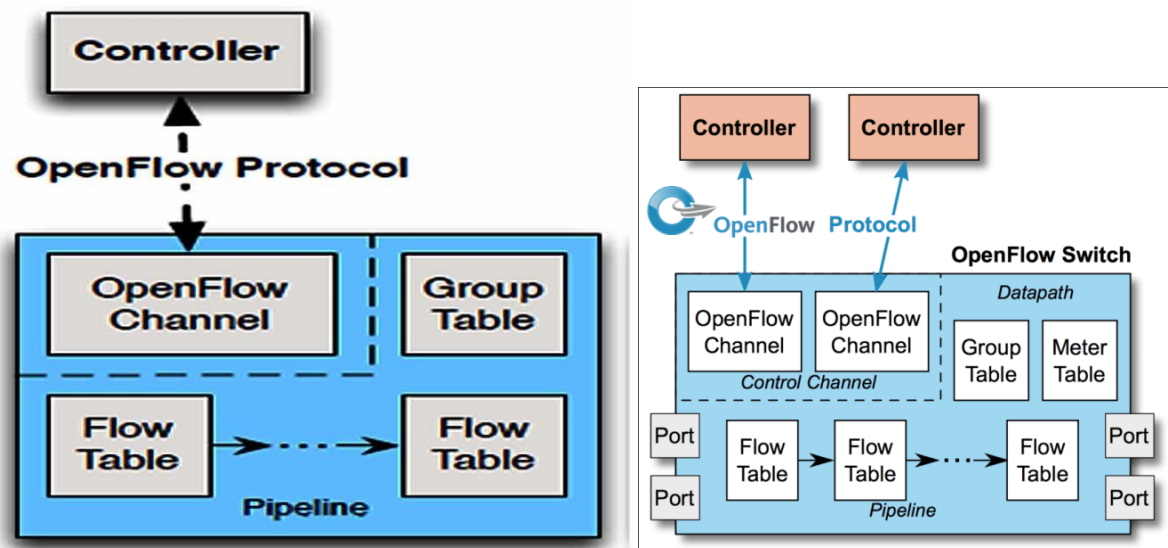


Figure 3.4: Openflow Qualify SDN [19].

3.2 Controller Plane

Although control is exercised to various degrees in other planes (note), the SDN controller plane is modeled as the home of one or greater SDN controllers. This clause describes the functional

components of an SDN controller and its relation to other controllers and other administrative domains. As will as a result emerge, now not all duties of the SDN controller can be allocated to precise practical components; the structure sees no price in proliferating blocks beyond the modern-day level.

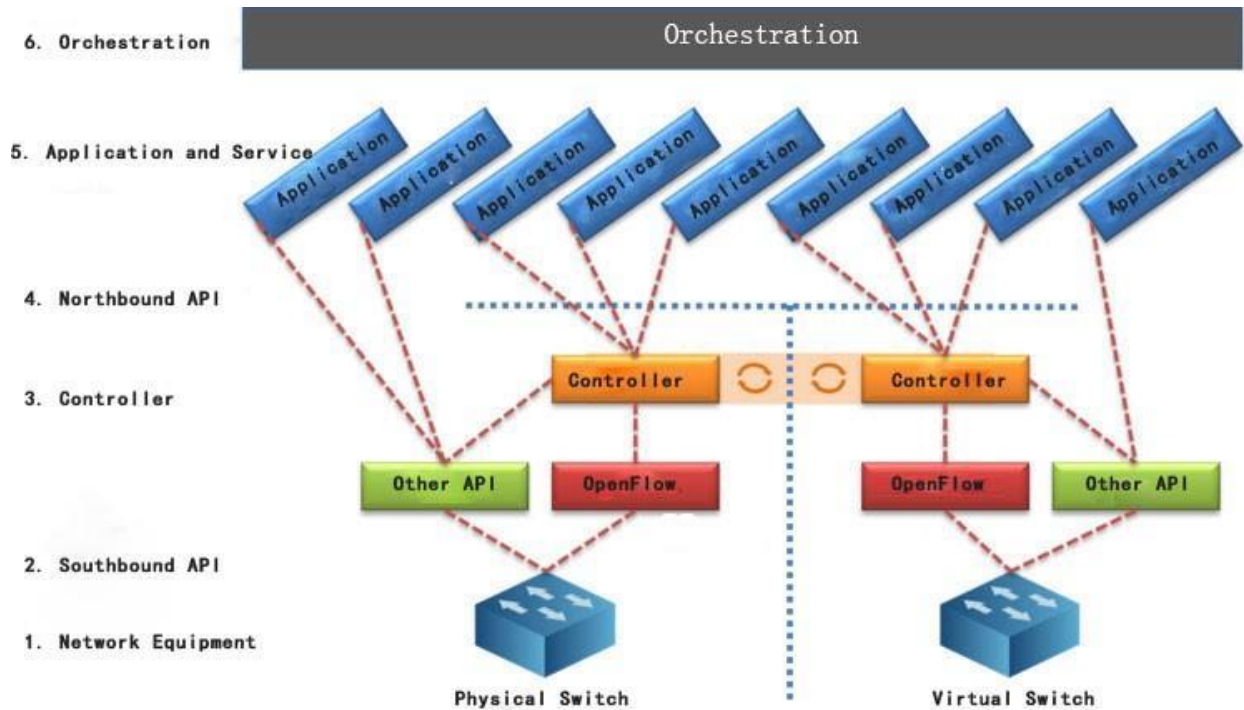


Figure 3.5: SDN Control logic [21].

The control layer consists of the number of orchestrators and controllers. Orchestrators provide end-to-end lifecycle management skills to the upper Application Layer. The orchestrators get hold of requests from the application layer (such as trade requests or new deployment request), and fulfill these requests through direct conversation underlying controllers. These controllers, in turn, screen parts of the Infrastructure layer. Orchestrators are consequently provider aware, whether or not the provider request includes network, compute, storage, security, or a mixture of all functions. Orchestrators may be unique to a service; however inter-orchestrator competencies are required to hand off requests in a disbursed or area specific model. For example, one orchestrator may handle data core service requests while another may additionally handle WAN networking requests. Management area boundaries may additionally also exist, whether within a single business enterprise, or across corporation boundaries. [21]

The controllers supply resource abstraction to the orchestrations. An Orchestration does not need to understand the underlying infrastructure; however, the controllers provide this mapping from a service view to an aid view. If an orchestrator receives a request to installation a workload on a VM, the controllers virtually make that occur in the Infrastructure Layer. The orchestrator has the understanding of which controllers to make the request of, whilst the controllers have the know-how of which sources to make the requests to. For example, a software request may additionally be obtained by using an orchestrator to set up the application on a new VM, with certain compute, storage and networking attributes. The orchestrator parses the request into how it receives realized via the controllers it communicates with, across the visible environment. One controller may also be called to provision a new VM in Data Center X, while every other might also be known as to provision available storage ability in Data Center Y. Yet another may be known as to establish a secure network throughout the two Data Centers to make certain the new VM has storage resources. [21]

Controllers take care of infrastructure demands, in response to requested services. The SDN structure does now not specify the inner graph or implementation of an SDN controller. It ought to be a single monolithic process; it should be a confederation of identical processes arranged to share load or shield one another from failures; it should be a set of distinct functional components in a collaborative arrangement; it should subscribe to exterior offerings for some of its functions, for example course computation. Any combination of these selections is allowed: the SDN controller is considered as a black box, defined by means of its externally-observable behavior. Controller factors are free to execute on arbitrary compute platforms, including compute assets local to a bodily NE. [17]

They might also execute on dispensed and possibly migratory resources such as on virtual machines (VMs) in statistics centers. The principle of logically centralized control is explored in element below; here, it suffices to say that the SDN controller is understood to have international scope, for some fee of globe, and that its components are understood to share facts and state, such that no external block need concern itself with conflicting or contradictory instructions from the controller. To the extent that the OSS influences resources or states, it is concern to the same coordination requirement with any SDN controllers that can also be involved.

Multiple supervisor or controller components may additionally have joint write get entry to to network resources, but to comply with SDN principles, they need to either

- Be configured to manipulate disjoint sets of resources or actions, or
- Be synchronized with every other so that they by no means trouble inconsistent or conflicting commands.

3.2.1 SDN controller functional components

Having just referred to that the SDN controller is a black box, it is nonetheless useful to conceptualize a minimum set of useful elements within the SDN controller (figure 3.5), namely data plane manipulates characteristic (DPCF), coordinator, virtualizer, and agent. Subject to the logical centralization requirement, an SDN controller can also encompass arbitrary additional functions. An aid facts base (RDB) models the modern-day facts model instance and the necessary supporting competencies [20]

3.2.1.1 Data plane control function

The DPCF component efficaciously owns the subordinate assets reach able to it, and makes use of them as recommended by using the OSS/coordinator or virtualized(s) that controls them. These sources take the form of a data model instance accessed through the agent in the subordinate level. Because the scope of an SDN controller is predicted to span multiple(virtual) NEs or even multiple digital networks (with a wonderful D-CPI instance to each), the DPCF need to include a function that operates on the aggregate. This feature is typically known as orchestration. This architecture does not specify orchestration as a wonderful practical component.

3.2.1.2 Co-ordinator

To set up each patron and server environments, management functionality is required. The coordinator is the useful factor of the SDN controller that acts on behalf of the manager. Clients and servers require management, for the duration of all perspectives on data, manipulate and application airplane models, so coordinator functional blocks are ubiquitous.

3.2.1.3 Virtualizer

An SDN controller affords offerings to applications by using way of a statistics mannequin occasion that is derived from the underlying resources, management-installed policy, and nearby or externally available support functions. The useful entity that supports the records mannequin instance and coverage at an A-CPI (application-controller aircraft interface) is called a virtualizer. It presents the local have confidence area boundary to the corresponding agent, which represents the client's view of the records model instance. A virtualizer is instantiated by using the OSS/coordinator for every patron software or organization. The OSS/coordinator allocates resources used via the virtualizer for the A-CPI view that it exposes to its application client, and it installs policy to be enforced by the virtualizer. The effect of these operations is the introduction of an agent for the given client. The virtualizer might also be notion of as the process that receives client-specific requests across the A-CPI, validates the requests against the policy and assets

assigned to the client, translates the request into phrases of the underlying resources, and passes the results on to the DPCF and the D-CPI. Virtualizer and DPCF and per chance different SDN controller features need to collaborate to provide features such as notification interpretation, useful resource sharing, implicit provider services, and transactional integrity. [22]

3.2.1.4 Agent

Any protocol must terminate in some form of useful entity. A controller-agent model is appropriate for the relation between a controlled and a controlling entity, and applies recursively to the SDN architecture. The controlled entity is special the agent, a practical component that represents the client's resources and competencies in the server's environment. An agent in a given SDN controller at degree N represents the sources and actions on hand to a client or application of the SDN controller, at level N+1. An agent in the level N-1 statistics plane represents the resources and movements handy to the given level N SDN controller. Even though the agent's bodily place is inside the servers have faith domain (i.e., on a server SDN controller platform), the agent notionally resides in the client's believe area.

3.2.1.5 Other controller component

To keep away from over specification, the structure solely describes functions that are required of an SDN controller, however does not avoid additional functions. These can also take the shape of applications or features supported with the aid of the controller. These elements may be exported to some or all of the server's external applications clients, or used internally by way of the issuer administration for its own purposes. As factors of the SDN controller, such applications or features are difficulty to the same synchronization expectation as other controller components. To facilitate integration with third party software, the interfaces to such purposes or aspects may additionally be the same as these of others at the A-CPI. The security components of such embedded functions are necessary to understand. Because they execute in the server's believe domain, they will be challenge to the server's test, verification, audit and launch administration cycle.

3.2.2 Legation of control

Although a key principle of SDN is noted as the decoupling of control and data planes, it is clear that an agent in the data plane is itself exercising control, albeit on behalf of the SDN controller. Further, a quantity of features with control components are widely viewed as candidates to execute

on community elements, for instance OAM, ICMP processing, MAC learning, neighbor discovery, defect consciousness and integration, safety switching. A greater nuanced reading of the decoupling precept approves an SDN controller to delegate control features to the data plane, concern to a requirement that these features behave in ways acceptable to the controller; that is, the controller have to in no way be surprised. This interpretation is vital as a way to follow SDN concepts to the real world.

Criteria that motivate the controller to delegate a feature to the data plane include: Rapid real-time response required to network events

- A giant quantity of traffic that need to be processed
- Byte- or bit-oriented functions that do not simply lend themselves to packetization, for example repetitive SDH multiplex area overhead
- Low-value, possibly repetitive, predictable, well-understood, totally standardized behavior, for example encryption, BIP, AIS insertion, MAC learning, CCM exchanges
- Survivability or continuity in case of controller failure or re-initialization
- Functionality oftentimes reachable in facts plane silicon, e.g., safety switching state machines, CCM counters and timer
- No perceived probability to add cost through separating the function.

Assuming the uncooked data can be made available, an SDN controller continually has the option now not to delegate a manipulate function, but to habits the indispensable operations itself. The standards listed above have an effect on whether or not such a choice is practical. [23]

3.3 Application layer

Application layer is open place to strengthen as tons revolutionary application as feasible by using leveraging all the community data about community topology, network state, network statistics, etc. There can be numerous types of purposes which can be developed like these associated to community automation, community configuration and management, network monitoring, network troubleshooting, community insurance policies and security. Such SDN functions can provide a range of end-to-end solutions for real world organization and records center networks. Network vendors are coming up with their set of SDN applications. (Shown in figure 3.6) For example, Brocade has following very beneficial applications:

1. Brocade Flow Optimize
2. Brocade Virtual router
3. Brocade Network advisor

HPE is additionally one supplier having SDN App keep which contains many SDN apps from distinct groups as well. For example:

- HPE Network Optimizer
- HPE Network protector
- HPE Network visualizer

- NEC UNC for HP SDN VAN Controller
- Aricent SDN Load balancer
- TechM clever glide steering
- TechM server load balancer

As we quickly touched Openflow in preceding article, we would now cover small print of southbound communication from control layer to infrastructure layer (network switches) through Openflow protocol. Openflow has been instrumental in the revolution of SDN in the experience that it has been key to show-case separation of manipulate aircraft from facts plane. Openflow is the trendy specification provided through Open Networking Foundation (ONF), and is evolving over the time with assist for various requirements of current world networking. Current model of the Openflow protocol is 1.5.1. [24]

An SDN application may also invoke different external services, and can also orchestrate any variety of SDN controllers to reap its objectives. The OSS hyperlink and the coordinator function recognize that, like the other most important blocks of the architecture, SDN functions require at least a certain amount of a priori information of their environments and roles.

- An application plane entity might also act as an information model server, in which case, it exposes a records mannequin occasion for use by way of other applications. Formally, the different applications are clients, which speak to the SDN utility server agent shown in figure 3.6
- An application plane entity may act as an information model client, in which case it operates on a facts model occasion exposed through a server entity. The server entity may also be an SDN controller or a subordinate application.
- A software airplane entity may additionally act in both roles simultaneously. For example, a path computation engine (PCE) may also count number on an SDN controller for virtual network topology information (maintained in a site visitors engineering database), while supplying the SDN controller a course computation service.

Activity across the A-CPI normally includes queries or notifications about the kingdom of the virtual network, and commands to alter its state, for example to create or adjust network connectivity or visitors processing functions between network client layer (data plane) handoff points, with some distinct bandwidth and QoS. The A-CPI can also be used for extra functions, for example as an get entry to point to configure a carrier chain through one or extra layer 4-7 services or as an input to manipulate virtualized community functions. Note – In terms of network behavior, service chaining is just the steering of traffic through a terrific set of components. The brought cost at an A-CPI may also be the ability to specify a sequence of aspect functions, watching for that the SDN controller will choose the most appropriate cases of these features and observe the pertinent traffic forwarding rules. The utility could additionally aid programming of component attributes, or even instantiate new virtualized network functions at most desirable factors in the topology.

North Bound API: Northbound interface: is supposed for communication with upper, Application layer and would be in frequent realized via REST APIs of SDN controllers.

South Bound API: Southbound interface, is meant for communication with lower, Infrastructure layer of community factors and would be in typical realized through southbound protocols – Openflow, Netconf, Ovsdb, etc. [24]

3.4 Management

Management covers infrastructure assist tasks that are no longer to be achieved with the aid of the application, controller and information planes themselves. Management might also a perform operations that the application, controller- and data planes are restrained from doing by coverage or for different reasons. Perhaps the single most necessary motive to forestall a project from being accomplished through SDN components is that the SDN controller may also dwell in a consumer trust domain, while business reasons mandate that core administration and help functions be achieved inside the issuer trust domain. Although an agent policy may want to be devised that totally depended on its controller, the transparency policy and coverage enforcement software would on the other hand have to be hooked up by the provider's manager. For security reasons, the default conduct is advocated to be to expose nothing, alternatively than everything.

The SDN structure recognizes classical administration features such as equipment inventory, fault isolation, software improves and the like, but regards them as mostly out of scope of SDN. One of the perceived benefits of SDN is allowing clients (in foreign trust domains) to perform many of the actions that are today performed by using management systems. The ordinary OSS interface is anticipated to play a smaller position over the path of time, as client applications take on greater accountability via SDN controllers. Within the scope of SDN are the SDN-specific management functions, particularly recording and expressing enterprise relationships (policies) between provider and client, and configuring SDN entity environment and initialization parameters. This consists of coordinating facts aircraft handoff points, identification conventions, reachability and credentials among logical and physical entities.

The SDN architecture requires that this information be configured into the relevant SDN NEs, controllers, and applications, but does now not specify the nature or shape of the OSSs. In the everyday case, each client-server pair of information plane, controller and application level entities lies in a separate believe area. Where a trust boundary exists in the SDN hierarchy, a corresponding believes boundary additionally exists in the management domain. Managers called OSSs in this record – in special trust domains may additionally want to alternate information, but this change is past the scope of the SDN architecture. Two administration roles are recognized: server supervisor and purchaser manager. The responsibilities of the server manager are now not the identical as those of the consumer manager. [21]

CHAPTER-4

Openflow Basics

4.1 Openflow

OpenFlow empowers arrange controllers to decide the way of system bundles over a system of switches. The controllers are unmistakable from the switches. This detachment of the control from the sending takes into account more refined traffic the executives than is plausible utilizing access control records (ACLs) and steering conventions. Additionally, OpenFlow permits changes from various sellers frequently each with their own exclusive interfaces and scripting dialects to be overseen remotely utilizing a solitary, open convention. The convention's creators consider OpenFlow an empowering agent of programming characterized organize (SDN).

4.2 A Brief of OpenFlow SDN

ONF characterizes OpenFlow as the main standard correspondences interface characterized between the controls and sending layers of a SDN design. OpenFlow enables direct access to and control of the sending plane of system gadgets, for example, switches and switches, both physical and virtual (hypervisor-based). [1]

4.3 OpenFlow and OpenFlow Switch

OpenFlow is a programmable system convention for SDN condition, which is utilized for correspondence between OpenFlow switches and controllers. OpenFlow isolates the programming of system gadget from hidden equipment, and offers an institutionalized method for conveying a brought together, programmable system that can rapidly adjust to changing system necessities.

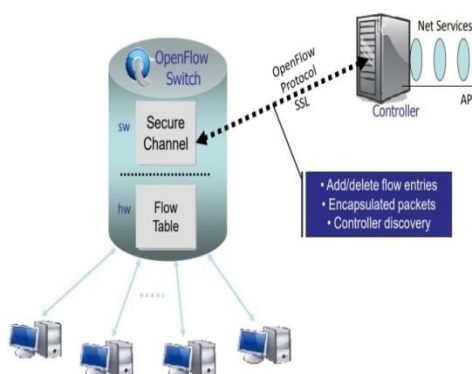


Figure 4.1: Open flow and open flow switch communicates over OpenFlow channel to an external controller [1].

An OpenFlow switch is an OpenFlow-empowered information switch that imparts over OpenFlow channel to an outside controller. It performs parcel query and sending as indicated by at least one stream tables and a gathering table. The OpenFlow switch speaks with the controller and the

controller deals with the switch by means of the OpenFlow switch convention. They are either founded on the OpenFlow convention or good with it.

4.4 OpenFlow Switch Process Systems

An OpenFlow switch can just capacity with the team up work of three basic components: stream tables introduced on switches, a controller and a restrictive OpenFlow convention for the controller to talk safely with switches. Stream tables are set up on switches. Controllers converse with the switches through the OpenFlow convention and force arrangements on streams. The controller could set up ways through the system improved for explicit attributes, for example, speed, and least number of bounces or decreased inactivity.

4.5 Differences of OpenFlow Switch vs Conventional Switch

In an ordinary switch, parcel sending (the information plane) and abnormal state directing (the control plane) happen on a similar gadget. While for an OpenFlow switch, the information plane is decoupled from the control plane: with the information plane executed in the switch itself yet the control plane in programming and a different SDN controller settles on abnormal state steering choices. The switch and controller convey by methods for the OpenFlow convention. OpenFlow switch consequently helps the accompanying points of interest:

- With OpenFlow empowered switch, the SDN controller could course non basic/mass traffic on longer courses that are not completely used.
- The SDN controller can without much of a stretch execute load-adjusting at high information rates by simply guiding distinctive streams to various hosts, just doing the set-up of the underlying streams.
- Traffic can be detached without the requirement for vlan's, the SDN controller of OpenFlow switch can simply decline certain associations.
- Setup a system TAP/Sniffer effectively for any port or even explicit traffic by programming the system to send a copy stream to a system checking gadget.
- It likewise takes into account the advancement of new administrations and thoughts all in programming on the SDN controller, also to quicken new highlights and administrations.

4.6 State of the Art of Open Switch

OpenFlow change is intended to give consistency in rush hour gridlock the executives and designing, by making control work autonomous of the equipment it's planned to control. This blend of open source programming and product equipment holds the potential for phenomenal

productivity and operational dexterity, which fitted well on the planet where arrange turns out to be progressively assorted and requesting. Empowering OpenFlow on physical changes and move to OpenFlow switch is something that most customers have been progressing in the direction of. FS.COM switch product offering comprises of 10GbE switch, 40GbE switch and 100GbE switch that bolsters OpenFlow, which can be utilized as OpenFlow switches in open systems administration environment. [2]

CHAPTER-5

Mininet Basics

5.1 Mininet

We have different accessible options of controller for SDN like POX, NOX, Onix, reference point and Floodlight. The systems OS controls the information plane gadgets through restricted interface called as OpenFlow which characterizes the sending of low dimension information plane gadgets. SDN handles a convoluted assignment as well as it does as such in distributive path running in a quickly evolving condition. Present day server farm incorporates a large number of switches and hundred a large number of hosts. In this manner SDN must be recreated over numerous servers. To execute SDN is expensive undertaking in light of the fact that SDN usage requires SDN steady

equipment assets. There is different open source organize emulators and test system accessible on the web.

A portion of the accessible emulators are Mininet, Die Cast and Model Net and so forth. Most utilized system test system is NS-2. The one of the limitation of the NS-2 is it doesn't give us the actual networking condition. We have utilized Mininet emulator since it gives practical condition to the client. We can say that Mininet goes about as a testing stage for the SDN. It encourages us in quickly prototyping the vast systems with constrained resources on a solitary PC.

Table 2: OS Types and versions Other Requirements.

OS Type	OS Version	Virtualization Software	X Server	Terminal
windows	7.8	Virtualbox	Xming	Putty
windows	XP	Virtualbox	Xming	Putty
Mac	OS X 10.7-10.8 Lion/Mountain	Virtualbox	Download and Install XQartz	Terminal.app (built.in)
Mac	OS X 10.5-10.6 Leopard/Snow Leopard	Virtualbox	X11(Install from OS X main system	Terminal.app (built.in)

			DVD, preferred), or download XQuartz	
Linux	Ubuntu 13.10	Virtualbox	X server already installed	Gnome terminal + SSH built in

Mininet is advantageous in light of the fact that it gives precision in execution. It is simple being used and also provides versatility. Mininet underpins different topologies that assistance us in production of thousands of hubs and can perform testing on them. We can adjust the conduct of these topologies as indicated by our need. Mininet bolster light weight virtualization that gives us the virtual system which is like the genuine network, running genuine piece, switch and application code and so on. Mininet depends on Command Line Inter face (CLI). Mininet switches support OpenFlow controller for programming characterized systems administration and custom topology. To begin Mininet in Linux plat structure we enter order in terminal as: Sudo mn. So as to run Mininet as root we should utilize the Sudo watchword to run the Mininet. It begins the Mininet and makes then it works by including controller, has, and switches and includes connects between them. This direction at first make have sh1, h2 and switch s1.

Chapter 6

SDN Tree Topology Implementation using Mininet

6.1 Introduction:

In this part of the paper we broadly discussed about the implementation of SDN in Mininet along with several tools which needed to be interconnected to get the output. Such as virtualbox, Ubuntu server 14.04.5, OpenDaylight, Mininet, Wireshark, Xming, PuTTY.

Virtualbox

Users of virtualBox can load more than one guest OS below a single host-system(host,OS).Each guest can be started, paused and stopped independently inside its personal virtual computing device(VM). The user can independently configure each VM and run it under a preference of software-based virtualization or hardware assisted virtualization if the underlying host hardware supports this. The host OS and visitor Oss and functions can communicate with every different through a number of mechanisms which include a frequent clipboard and a virtualized community facility. Guest VMs can also immediately speak with each different if configured independently inside its personal virtual computing device (VM).[25]

Ubuntu Server 14.04.5

To construct the OpenDaylight virtual machine, we have downloaded the Ubuntu Server ISO image from the ubuntu.com web site. Then we mounted it in a new VM in VirtualBox.

OpenDaylight

OpenDaylight is especially available, modular, extensible, scalable and multi-protocol controller infrastructure built for SDN deployments on current heterogeneous multi-vendor networks. OpenDaylight offers a model-driven service abstraction platform that permits users to write apps that without difficulty work throughout a broad range of hardware and south-bound protocols.[26]

Wireshark

Wireshark is a data capturing application that ‘understands’ the structure(encapsulation) of distinct networking protocols. It can parse & show the fields, along with their meanings as detailed through unique networking protocols. Wireshark uses pcap to capture packets to capture packets, so it can only capture packets on the sorts of networks that pcap supports. [27]

Xming

Xming provides a X Window System display server, a set of traditional X sample applications and tools, and a set of layouts. Xming can be used with Secure Shell (SSH) implementations to securely forward X11 sessions from other computers. It supports PuTTY and ssh.exe, and comes with a version of PuTTY’s plink.exe. The Xming project also offers a portable version Putty. When SSh forwarding is not used, the local file Xn hosts must be updated with host name or IP address of the remote machine where GUI application is started. [28]

PuTTY

PuTTY is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection.

6.2 Simulation

Open Daylight

We have named the virtual machine *OpenDaylight*. To Configure it uses two CPUs and 2 GB or RAM. This is the minimum configuration to support OpenDaylight. Then add a host-only network adapter to the VM.

The first network adapter of the VM is attached to the VirtualBox NAT interface by default and already configured when the VM boots up. We need to configure the second network adapter that is attached to the VirtualBox host - only interface.

Ip Commands: <i>ip addr show</i>

```

sword:
t login: Tue Apr  2 23:43:47 +06 2019 on tty1
come to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic i686)

Documentation:  https://help.ubuntu.com/

system information as of Tue Apr  2 23:43:47 +06 2019

system load:  0.54                Processes:           105
Usage of /:   27.7% of 6.77GB     Users logged in:    0
Memory usage: 5%                 IP address for eth0: 10.0.2.15
Swap usage:   0%

Graph this data and manage this system at:
https://landscape.canonical.com/

A new release '16.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

von@server1:~$ ip addr show
lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:13:9a:2d brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe13:9a2d/64 scope link
        valid_lft forever preferred_lft forever
eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:85:b3:67 brd ff:ff:ff:ff:ff:ff
von@server1:~$

```

Fig6.1: Screenshot of configuring IP address of OpenDaylight.

Here, we see interface in eth1, that it has no ip address. Because this the second network adapter connected to *vboxnet0*. On this interface using the DHCP, if DHCP client request then virtualBox can assign an ip address. No we run the following command or that it becomes set the eth1.

Ip commands:

```
sudo dhclient eth1
```

Now we can check the the ip address assign into eth1

```

ovon@server1:~$ ip addr show eth1
eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:85:b3:67 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.102/24 brd 192.168.56.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe85:b367/64 scope link
        valid_lft forever preferred_lft forever
ovon@server1:~$

```

Fig 6.2: Screenshot of Ip address of OpenDaylight.

Now, here we can see that in virtualBOX DHCP server that connected to the host only network give us an ip address which is *192.168.56.102*. So, this ip address now we can use to connect any application that running to the VM.

Now, we need to configure this interface eth1. By using following command.

Ip Command: *sudo nano /etc/network/interfaces*

For the OpenDaylight we need to install JAVA because OpenDaylight SDN controller is Java program. To run Java we are using following some commands-

```

$ sudo apt-get update
$ sudo apt-get install default-jre-headless

```

Now we the next step we done in set the java environment variable by using following command-

```

$ sudo nano ~/.bashrc

```

After creating bashrc file we add some following commands in that file-

```

export JAVA_HOME=/usr/lib/jvm/default-java

```

Then we run the Java successfully.

After that now the main and last step for running the OpenDaylight is downloading the OpenDaylight software from the **OpenDaylight web site**.

For extracting the file we use the following command-

```
$ tar -xvf distribution-karaf-0.4.0-Beryllium.tar.gz
```

This command creates the folder named *distribution-kara-f0.4.0-Beryllium* which contains the OpenDaylight software and plugins. OpenDaylight is packaged in a **karaf container**. Karaf is a container technology that allows the developers to put all required software in a single **distribution** folder.

Start OpenDaylight

We need to run the karaf command inside the package distribution folder for run the OpenDaylight. We use the following commands-

```
$ cd distribution-karaf-0.4.0-Beryllium
```

```
$ ./bin/karaf
```

NOW, finally we can see that our OpenDaylight starts running. Here the snap of this-

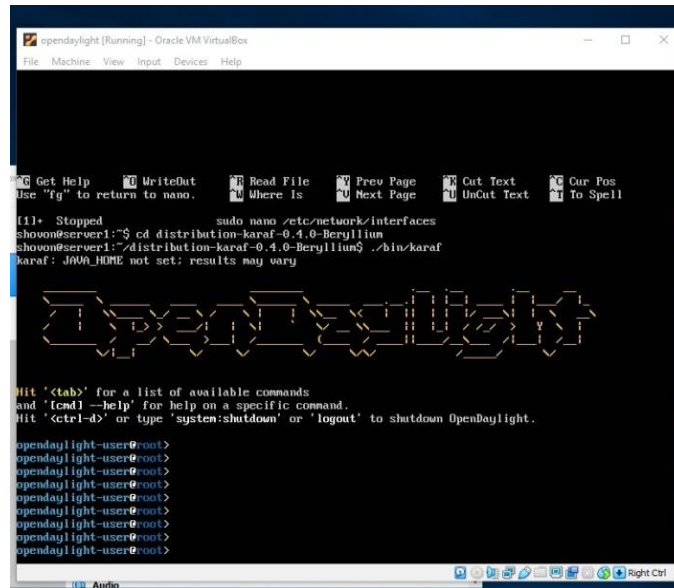


Fig 6.3: Screenshot of starting the OpenDaylight.

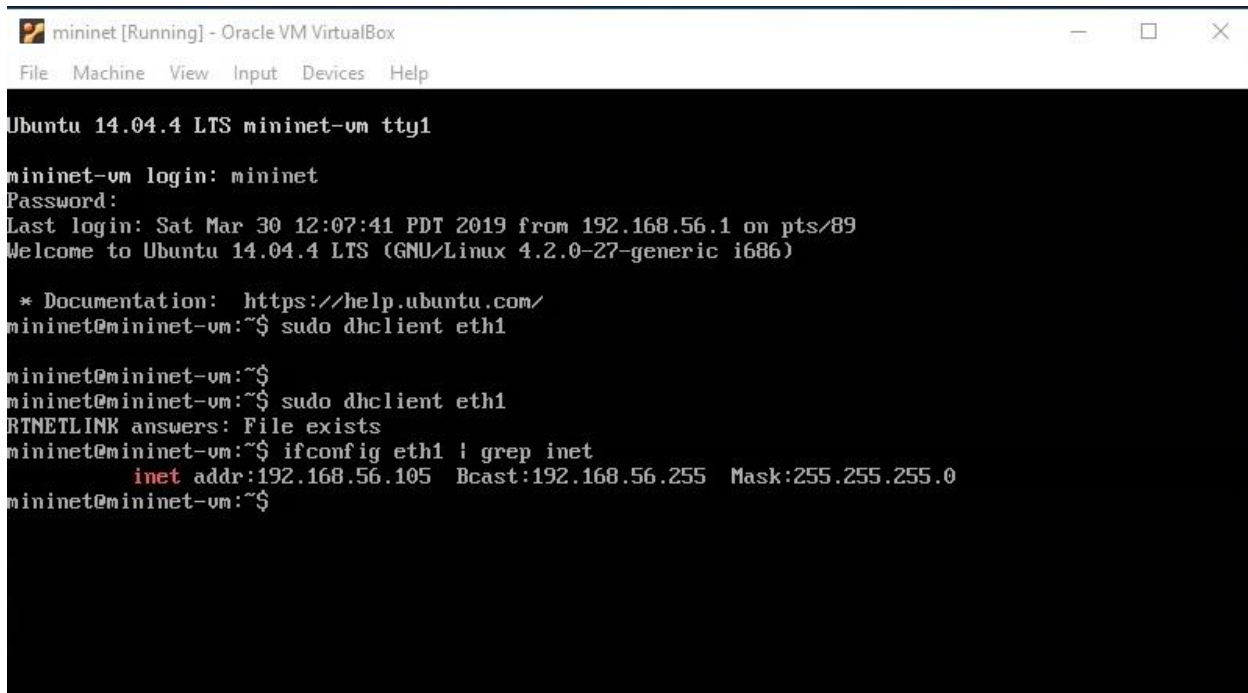
Mininet

Firstly to set the mininet we need to download the mininet virtualbox from <http://mininet.org/download>. I chose the latest version available, which was *mininet-2.1.0-130919-ubuntu-13.04-server-amd64-ovf.zip*. This file is a compressed ZIP archive containing two files so, after downloading it, decompress it and save the files to my hard drive.

Then we go the Vm and click on setting and named the Vm as *Mininet*. Then go to the adapter 2 and choose host only network. And adding the mininet downloaded file in Vm. Then we click on start and our mininet start running. Here, user name is mininet and the password also mininet. Here the snap of the mininet that we run.

After configure both Mininet and OpenDaylight now our task is to create a SDN topology. Here we create a *tree topology*. So, in Opendaylight we found the ip address is *192.168.56.102*. To check the mininet ip address we use the following command-

```
$ sudo dhclient eth1
$ ifconfig eth1 : grep inet
```

```

mininet [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Ubuntu 14.04.4 LTS mininet-vm tty1
mininet-vm login: mininet
Password:
Last login: Sat Mar 30 12:07:41 PDT 2019 from 192.168.56.1 on pts/89
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic i686)

 * Documentation:  https://help.ubuntu.com/
mininet@mininet-vm:~$ sudo dhclient eth1

mininet@mininet-vm:~$
mininet@mininet-vm:~$ sudo dhclient eth1
RTNETLINK answers: File exists
mininet@mininet-vm:~$ ifconfig eth1 | grep inet
    inet addr:192.168.56.105 Bcast:192.168.56.255 Mask:255.255.255.0
mininet@mininet-vm:~$

```

Fig 6.4: Screenshot of IP address of Mininet.

After this we get the mininet ip address which is 192.168.56.105. We see *eth0* is connected to the host-only interface because it has IP address 192.168.56.1025 which is in the address range assigned by the VirtualBox host-only network DHCP server. So we know we need to use IP address 192.168.56.105 to access applications running on this virtual machine.

Connect to the mininet Vm using SSH

Now in mininet Vm we open a new terminal and turn X forwarding on. (If you are using Windows, use Xming for an X Window System Server and Putty as an SSH client). By using following command-

```
$ ssh -X 192.168.56.105
```

After connecting this, we have to create a tree topology. For this we write the following command-

```
$ sudo mn --controller=remote,ip=192.168.56.102 --topo=tree,3,4
```

```
mininet [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
*** Adding controller
Connecting to remote controller at 192.168.56.102:6653
*** 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 h
28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h
53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s21
*** Adding links:
(s1, s2) (s1, s7) (s1, s12) (s1, s17) (s2, s3) (s2, s4) (s2, s5) (s2, s6) (s3, h1) (s3, h2) (s3, h3)
(s3, h4) (s3, h5) (s4, h6) (s4, h7) (s4, h8) (s5, h9) (s5, h10) (s5, h11) (s5, h12) (s6, h13) (s6,
h14) (s6, h15) (s6, h16) (s7, s8) (s7, s9) (s7, s10) (s7, s11) (s8, h17) (s8, h18) (s8, h19) (s8, h2
0) (s9, h21) (s9, h22) (s9, h23) (s9, h24) (s10, h25) (s10, h26) (s10, h27) (s10, h28) (s11, h29) (s
11, h30) (s11, h31) (s11, h32) (s12, s13) (s12, s14) (s12, s15) (s12, s16) (s13, h33) (s13, h34) (s1
3, h35) (s13, h36) (s14, h37) (s14, h38) (s14, h39) (s14, h40) (s15, h41) (s15, h42) (s15, h43) (s15
, h44) (s16, h45) (s16, h46) (s16, h47) (s16, h48) (s17, s18) (s17, s19) (s17, s20) (s17, s21) (s18,
h49) (s18, h50) (s18, h51) (s18, h52) (s19, h53) (s19, h54) (s19, h55) (s19, h56) (s20, h57) (s20,
h58) (s20, h59) (s20, h60) (s21, h61) (s21, h62) (s21, h63) (s21, h64)
*** Configuring 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 h
28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h
53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
*** Starting controller
c0
*** Starting 21 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s21 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
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 h2
7 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h5
2 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 X
h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52
h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h2
7 h28 h29 h30 h31 h32 h33 h34 _
```

Fig 6.5: Screenshot of configuring the Tree topology command.

For testing the network we use the following command-

```
$ pingall
```

This test that the OpenDaylight controller is working by pinging all nodes. Every host should be able to reach every other host.

```

File  Machine  View  Input  Devices  Help
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
h54 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
h55 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h56 h57 h58 h59 h60 h61 h62 h63 h64
h56 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h57 h58 h59 h60 h61 h62 h63 h64
h57 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h56 h58 h59 h60 h61 h62 h63 h64
h58 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h56 h57 h59 h60 h61 h62 h63 h64
h59 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h56 h57 h58 h60 h61 h62 h63 h64
h60 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h56 h57 h58 h59 h61 h62 h63 h64
h61 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h56 h57 h58 h59 h60 h62 h63 h64
h62 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h63 h64
h63 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h64
h64 -> 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 h2
6 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h5
1 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63
*** Results: 0% dropped (4032/4032 received)
mininet> ~

```

Fig 6.6: Screenshot of testing the all connections.

Graphical user interface of the OpenDaylight

In this stage we need to open a browser on your host system and enter the URL of the OpenDaylight User Interface (DLUX UI). It is running on the OpenDaylight VM so the IP address is 192.168.56.102 and the port, defined by the application, is 8181. So the URL is-

<http://192.168.56.101:8181/index.html>

Here, the user name is “admin” and password also is “admin” by default.

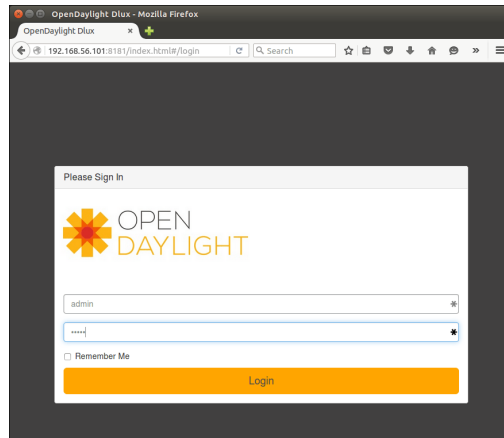


Fig 6.7: Screenshot of page layout of OpenDaylight .

Now we see our network topology in the OpenDaylight controller's topology tab.

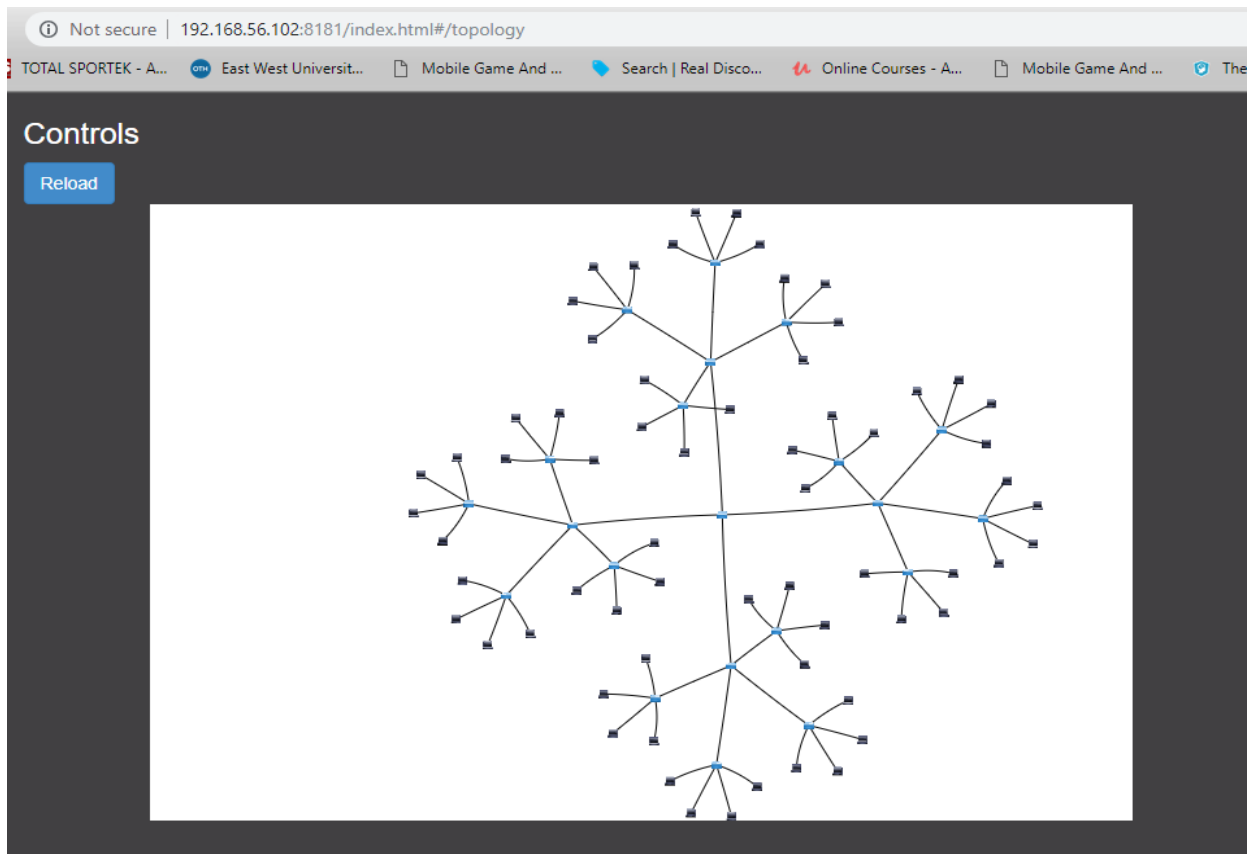
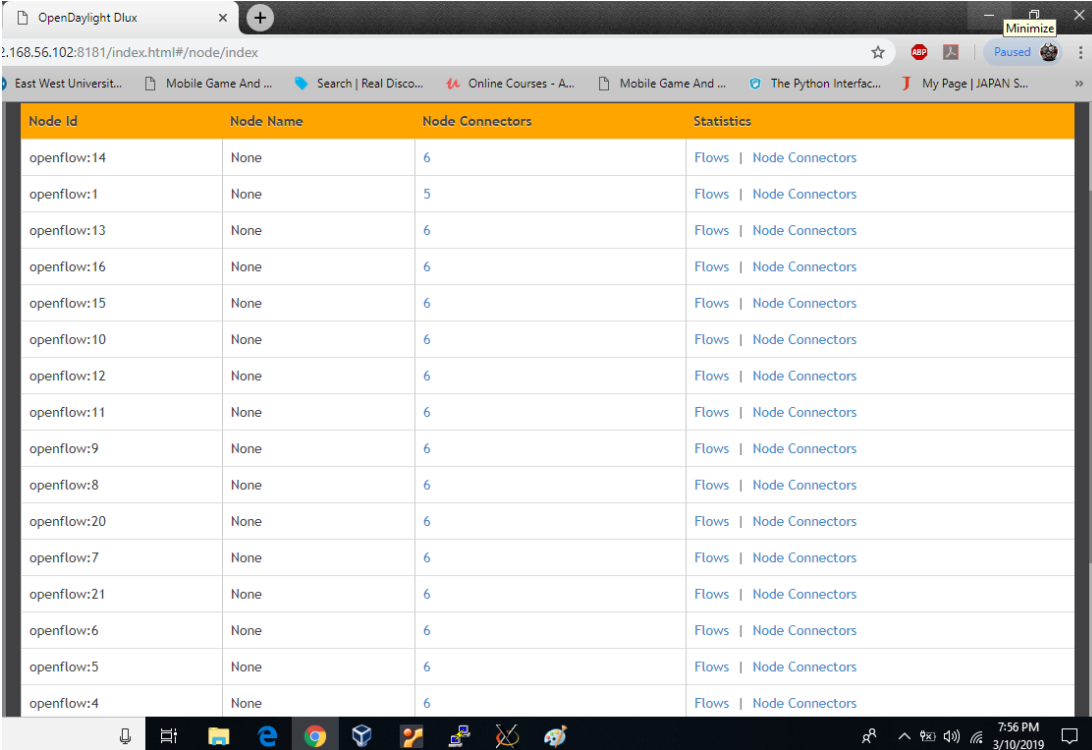


Fig 6.8: Screenshot of creating the tree topology.

we can see the network that is emulated by the Mininet network emulator. now any one can test OpenDaylight functionality by building different network topologies in Mininet with different attributes, and by using OpenDaylight to run experiments on the emulated network.

Nodes-

If we click on the nodes then we can see the information of each switch in the network.



Node Id	Node Name	Node Connectors	Statistics
openflow:14	None	6	Flows Node Connectors
openflow:1	None	5	Flows Node Connectors
openflow:13	None	6	Flows Node Connectors
openflow:16	None	6	Flows Node Connectors
openflow:15	None	6	Flows Node Connectors
openflow:10	None	6	Flows Node Connectors
openflow:12	None	6	Flows Node Connectors
openflow:11	None	6	Flows Node Connectors
openflow:9	None	6	Flows Node Connectors
openflow:8	None	6	Flows Node Connectors
openflow:20	None	6	Flows Node Connectors
openflow:7	None	6	Flows Node Connectors
openflow:21	None	6	Flows Node Connectors
openflow:6	None	6	Flows Node Connectors
openflow:5	None	6	Flows Node Connectors
openflow:4	None	6	Flows Node Connectors

Fig 6.9: Screenshot of node connections of the tree.

Capturing Open Flow messages (Wireshark)

To dive deeper into how SDN controllers and switches operate, you can also desire to view the OpenFlow messages exchanged between the controller and switches in the network. The Mininet VM comes with Wireshark installed, with a custom version of the OpenFlow dissector already set up.

So, we can start Wireshark on the mininet vm to capture the data on the interface to connect the host only network, which is eth1 in our case.

Now, we go to putty and enable the SSH and X11 for Xming. Then in the terminal for ip address 192.168.56.105 that we get, write the following command for running the Wireshark-

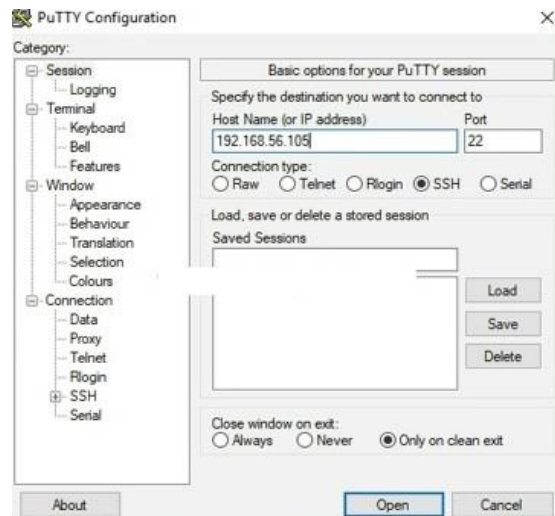


Fig 6.10: Screen shot of PuTTY configuration.

```
$ sudo wireshark &
```

You will see a warning dialog but you can ignore it. Starting Wireshark with root privileges is a security risk but, for our simple testing. In the display filter we write of and select loopback then click on apply. Now we will see only OpenFlow messages in the Wireshark display, as shown below.

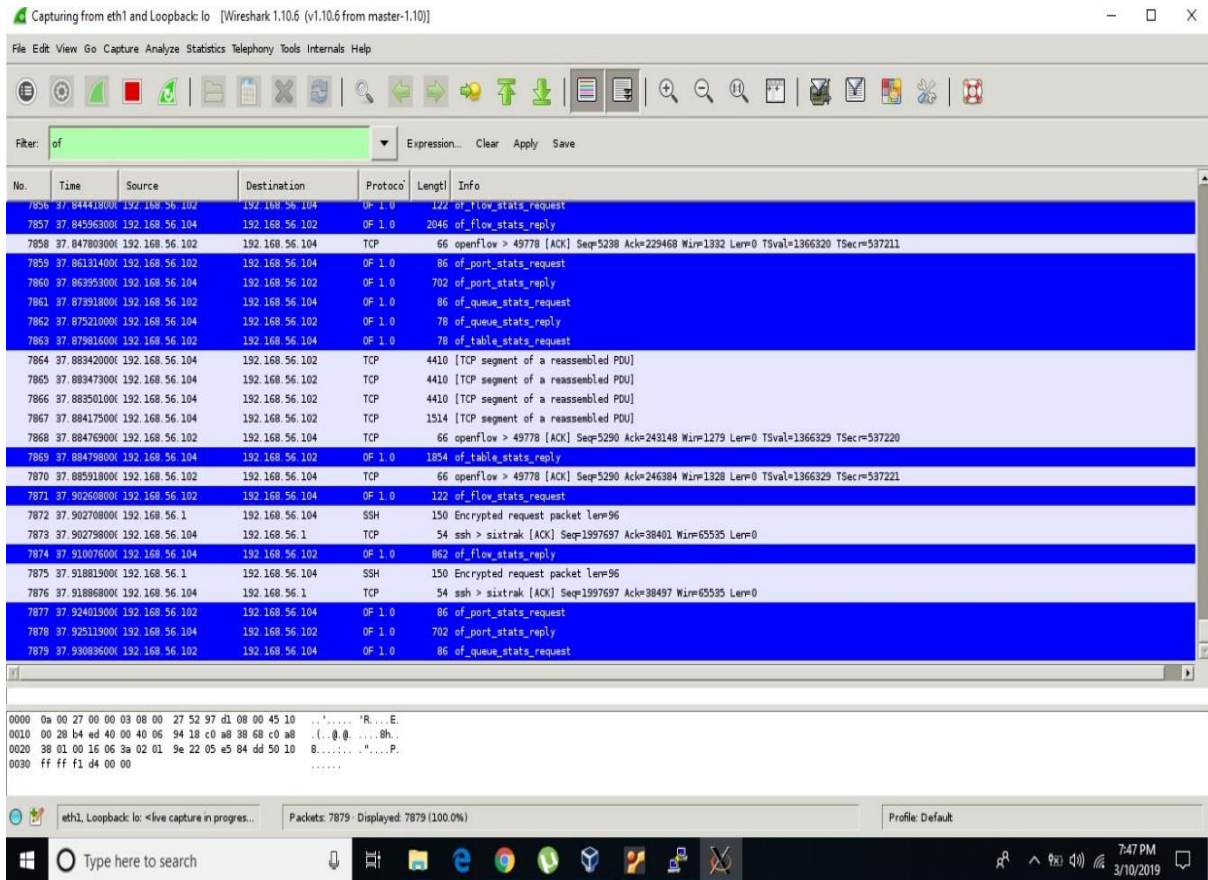


Fig 6.11: Screenshot wireshark that capture the data traffics.

6.3 Other Simulators:

Mininet is widely used, simplest and flexible simulator of SDN. There are also some other simulator such as,

- Mininet-Wifi (developers added virtualized WiFi stations and access points based on standard Linux wireless drivers to extend Mininet's functionality), [29]
- KNet (KNet builds the topology of the virtual network with switches, hosts, routers and servers).[30]
- NS3 (Discrete network of event simulators, ns-1, ns-2, ns-3 and ns-4 in particular. All are computer network simulators for discrete events, mainly used in research).[31]

Other than Opendaylight controller some other Openflow controllers also available as they are,

- Floodlight Controller (It is an open developer community - led Apache - licensed, Java - based OpenFlow controller).[32]
- MiniEdit (is an experimental tool created to show how to extend Mininet. To show how to create and execute network simulations using MiniEdit).[33]

CHAPTER-7

Future plan of SDN

7.1 SDN Migration plan

At the point when Windows Server 2019 is discharged this fall, the updates will incorporate highlights that undertakings can use to use programming characterized organizing (SDN).

SDN for Windows Server 2019 has various parts that have pulled in the consideration of early adopters including security and consistence, calamity recuperation and business congruity, and multi-cloud and half and half cloud.

SDN is one of the most recent innovations intended for Network Control. SDN can enable a Network Manager to change the way the system gadgets, for example Switches, Switches.etc, handle bundles by permitting unlimited oversight of the tenets set into system gadgets from a focal reassurance. SDN takes into consideration the whole control of as system to enable brisk reaction to changing system or business needs. SDN additionally has a great deal of analytic ability.

SDN has a hazardous future as it has issues to survive. The main issue is the support/remote capacity with the present security issues many won't have any desire to open their system to a potential programmer takeover. It is an Open Source Technology.

Another issue is the present province of Network Management approaches and practices with single gadget or single way center. At the point when a Network Manager takes a gander at SDN he just perceives how it can help or damage his system yet SDN is a lot greater and a ton of training still stays to be done to make SDN or comparable advances attractive.

SDN is a Human Centric Technology where the present innovation is Device Centric which is and dependably will be a test to get supervisors to embrace particularly when one individual can totally change your system, stockpiling, WAN, etc texture.

Many discussions about SDN's capacity to help with the "Cloud" however before we get SDN included we have to get the "Cloud" leveled out.

There are additionally genuine budgetary, preparing and essential sending issues.

SDN opens bunches of inquiries for the fate of development systems administration and figuring advances. We need progressively responsive innovation yet not at the expense of security and control. Over the long haul SDN might be conveyed in supplier organizes yet singular companies may think that its fair an excessive amount to send. SDN need much greater advancement and evidence of being a protected and deployable innovation.

7.2 SDN versus Conventional Systems Administration (Traditional Networking)

Conventional systems administration gadgets, for example, routers and switches are autonomous. Every gadget chooses how to send its traffic as system administrators arranging approaches that control traffic move through every gadget. Every individual gadget has two separate segments that cooperate to transport traffic through the system: A control plane, the cerebrums of the gadget that chooses where traffic ought to be sent, and an information plane which is in charge of sending information.

This is the customary system structure that we've utilized for quite a long time. Numerous preliminaries delivered programming and systems administration devices that, best case scenario, upgraded arrangement the board on numerous gadgets in the meantime. In any case, be that as it may, every gadget would decipher these arrangements by means of its cerebrum and course the choice to the basic sending plane. Virtualization endeavors concentrated on making sub-occasions of a similar gadget instead of structure a virtual system over the physical foundation.

SDN is an alternate story. With SDN, organizing gadgets are overseen and arranged from a focal framework called a SDN controller. The controller decouples the basic leadership segment (control plane) in systems administration gadgets from the information sending segment (information plane). It at that point brings together the control plane outside of the system gadget and empowers it to wind up programmable by outer administrations, predominantly the applications.

This creates a dynamic and flexible networking infrastructure that allows for a more efficient automation of different networking services.

SDN targets are for making higher virtual system layer over the physical one which takes into account making separate system areas for various applications or potentially clients. All new consistent system gadgets and administrations, for example, routers, switches, firewalls and burden balancers keep running over the physical system. For a considerable length of time, the system was the significant piece of framework lacking virtualization. With SDN comes the guarantee of virtualization the system to help progressively virtual conditions.

7.3 SDN Adoption

At first look, SDN reception appears like an easy decision. It's still in the beginning times, be that as it may, of picking up section to datacenters. This is typical with any new innovation as the greater part will in general be distrustful of early adopters. As examples of overcoming adversity come about more organizations will be keen on conveying SDN in their datacenters.

Presently, there are continuous dynamic SDN extends in different stages. While a few organizations are as yet assessing the advancements in Proof of Concepts (PoC) endeavors, others are now sending and completely working them SDNs. The more IT experts think about SDN and how they can profit, the rate of organization will absolutely get.

7.4 Virtual-Network peering

The new virtual systems administration peering usefulness in Windows Server 2019 enables undertakings to peer their very own virtual systems in a similar cloud area through the spine arrange. This gives the capacity to virtual systems to show up as a solitary system.

Central extended systems have been around for a considerable length of time and have given associations the capacity to put server, application and database hubs in various locales. Be that as it may, the test has dependably been the IP tending to of the hubs in restricting destinations. At the point when there are just two static locales in a conventional wide region organize, the IP conspire was generally static. You knew the subnet and tending to of Site A and Site B.

With Vnet Peering, while the outer area and texture that the host and applications frameworks are running in may radically change, the virtual system stays predictable. No compelling reason to change source and target addresses inside the application, no requirement for Web and Database set to change settings.

7.5 Virtual-Network encryption

Another huge improvement in Windows Server 2019 is the capacity for virtual-arrange traffic to be encoded between virtual machines. Traffic encryption isn't new to the business, anyway having the encryption worked into the working framework as the premise of hypervisor correspondences, server interchanges and application correspondences give both adaptabilities and that in the past was often done at the application layer.

Presently with Vnet encryption, whole subnet interchanges between host servers can be secured, and all system traffic inside that arrange is naturally encoded. For associations hoping to guarantee interchanges between a Web server and a database server is scrambled, Vnet encryption in Windows Server 2019 can be empowered. Since the correspondences are at the system/subnet level, if extra Web frontends and backend databases should have been included, every one of those servers joins the equivalent scrambled correspondence stream, offloading the verified interchanges from the application itself, improving execution and effectiveness.

Chapter 8

Conclusion

Traditional network systems have become very complicated to manage for network operators, the network systems are rapidly changing and it is still difficult to configure network devices across a broad network system. Operators can combine network devices from different vendors into a single, large network system in a traditional distributed network. SDN not only addresses these issues, but also simplifies them by separating control planes and network data planes and centralizing the control and management of the overall network system.

Networks are growing, bandwidth specifications are increasing along with the number of connected devices, and our data networks will need to change and adapt to that growth and change rate in order to keep up with the rest of Datacenter technology. It has visionary approach to IT networking that easily and quickly is now the preferred style of network management.

In this paper we have broadly discussed about Software-defined networking & simulated in mininet.

References :

- [1] Available at:
https://ieeexplore.ieee.org/document/6984209?fbclid=IwAR2_PIBJcz_028v94RSUZFKvaKE5eSNdRY72qPvg4RxGT8s6ROyVr9GFwkw [accessed at Jan 15th 2019]
- [2] Available at: Software-Defined Networking for Internet of Things: A Survey
- [3] Available at:
https://ieeexplore.ieee.org/document/6984209?fbclid=IwAR2_PIBJcz_028v94RSUZFKvaKE5eSNdRY72qPvg4RxGT8s6ROyVr9GFwkw
 [accessed at Jan 15th 2019]
- [4] Available at: <https://ieeexplore.ieee.org/abstract/document/7452335/authors#author>
 [accessed at Jan 15th 2019]
- [5] Available at: <http://mininet.org/overview/> [accessed at Jan 15th 2019]
- [6] Available at: <https://en.wikipedia.org/wiki/OpenFlow> [accessed at Jan 20th 2019]
- [7] **Software Defined Networks (S.D.N): Experimentation with Mininet Topologies Deepak Kumar* and Manu Sood** Software Defined Networks (S.D.N): Experimentation with Mininet Topologies Deepak Kumar* and Manu Sood
- [8] Available at: <http://www.gocertify.com/articles/sdn-is-the-future-of-it-networking?fbclid=IwAR0ueb486bvPVpxLTTHrxHJKbUQihhxaIVFtGlmK8Ckl6MCvYmOKAoO-n9o>
- [9] Available at: https://plvision.eu/rd-lab/blog/sdn/p4-programming-future-sdn?fbclid=IwAR0XhWjppuuOwor4xbG1jfilesjmJai4Z66vLQzOsGpvpbEENc6Pn1Tr-ov_o
 [accessed at Feb 5th 2019]
- [10] Yang (Intel Corp.), R. Dantu (Univ. of North Texas), T. Anderson (Intel Corp.) & R. Gopal (Nokia.) (April 2004). "Forwarding and Control Element Separation (ForCES) Framework".
- [11] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo (Nov 2004). "The SoftRouter Architecture"
- [12] Using Mininet for emulation and prototyping Software-Defined Networks
<https://ieeexplore.ieee.org/document/6860404> by RLS de Oliveira - 2014
- [13] Master Thesis "Tools for a Multi-Controller SDN Architecture"
eprints.networks.imdea.org/.../Tools_for_a_Multi_Controller_SDN_Architecture_2015 by SN Tamurejo Moreno - 2015

- [14] Design and Implementation of a Software Defined Network Based
...https://www.ee.iitb.ac.in/~karandi/thesis/ojas_thesis.pdf by O Kanhere - 2017
- [15] . Software Defined Networks (S.D.N): Experimentation with Mininet
...www.indjst.org/index.php/indjst/article/viewFile/100195/72639 Software Defined Networks (S.D.N): Experimentation with Mininet Topologies. Deepak Kumar*
- [16] Mininet as Software Defined Networking Testing Platform Karamjeet Kaur¹, Japinder Singh² and Navtej Singh Ghumman Conference Paper • August 2014
<https://www.researchgate.net/publication/287216738>
- [17] Available at: <https://www.sdxcentral.com/networking/sdn/definitions/inside-sdn-architecture/>
[Accessed at 20th jan 2019]
- [18] Available at: <https://www.sdxcentral.com/networking/sdn/definitions/open-sdn/> [Accessed at 25th jan]
- [19] Availavle at: https://www.opennetworking.org/wp-content/uploads/2014/11/TR_SDN-ARCH-1.0-Overview-12012016.04.pdf[Accessed at 25th jan 2019]
- [20] Available at : <https://www.cozlink.com/modules-a272-275-273/article-73747.html> [Accessed at 20th feb 2019]
- [21] Available at: <https://www.opennetworking.org/sdn-definition/>[Accessed at 11th jan 2019]
- [22] Available at: <https://www.sdxcentral.com/networking/sdn/definitions/inside-sdn-architecture/>[Accessed at 11th jan 2019]
- [23] Available at: https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf[Accessed at 20th jan 2019]
- [24] Available at: <https://www.howtoforge.com/tutorial/software-defined-networking-sdn-architecture-and-role-of-openflow/> [Accessed at 20th jan 2019]
- [25] <https://en.wikipedia.org/wiki/VirtualBox> [accessed at Feb 10th 2019]
- [26] https://en.wikipedia.org/wiki/OpenDaylight_Project
- [27] <https://en.wikipedia.org/wiki/Wireshark>
- [28] <https://en.wikipedia.org/wiki/Xming>
- [29] <https://www.quora.com/What-kind-of-monitoring-tools-are-used-for-SDN-deployments>
- [30] <https://searchsdn.techtarget.com/news/.../Five-must-know-open-source-SDN-controllers>

[31] Henderson, Tom (2012-06-09). "upcoming ns-3.1 release" (Mailing list). ns-3 GSoC 2015 students. Retrieved 2013-05-31.

[32] <https://www.sdxcentral.com> › Networking › SDN › SDN Definitions

[33] <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet>

List of Acronym

SDN: Software Defined Networking.

VM: Virtual Machine.

ONF: Open network function.

NE: Network element.

QoS: Quality of service

LLDP: Link layers discovery protocol.

STP: Spanning tree protocol.

BFD: Bidirectional forwarding detection.

ICMP: Internet control message protocol.

NOS: Network operating system.

FAWG: Forwarding abstraction working group.

NDMs: Nevertheless different space of models.

DPCF: Distributed point coordination function.

HPE: Hewlett Packard Enterprise Company.

VAN- Value added network.

PCE: Path computation engine.

CLI: Command line interface.

OS: Operating system.

Pcap: Packet capture.