IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 9, Issue.11, November 2021 E-ISSN: 2347-2693

Design and Implementation of Tree Topology in Software Defined Networking (SDN) using Mininet and OpenDaylight

S.M. Bayazid Khan[1*], M.M.H. Shojib[2]

[1]Dept. of ME, Dhaka University of Engineering & Technology, Gazipur, Bangladesh
[2]Dept. of ICT, Bangladesh University of Professionals, Dhaka, Bangladesh

[*]*Corresponding Author: alfadanga@gmail.com*

DOI: https://doi.org/10.26438/ijcse/v9i11.5167 | Available online at: www.ijcseonline.org

Received: 17/Nov/2021, Accepted: 20/Nov/2021, Published: 30/Nov/2021

***Abstract*—** Software-Defined Networking is an emerging network architecture approach that enables us to control or, program the network system intelligently and centrally using software applications. This allows the user to manage the entire network consistently and holistically, regardless of the underlying network technology. By installing and configuring OpenDaylight which is a modular open platform for customizing and automating networks of any size and scale, we then configured it by Mininet in Virtual Machine. Finally, packets are captured by Wireshark and help us to measure the scenario. In this paper, our purpose is to shed light on SDN related issues and give insight into the challenges facing the future of this revolutionary network model, from both protocol and architecture perspectives. Additionally, we aim to present different existing solutions and mitigation techniques that address SDN scalability, elasticity, dependability, reliability, high availability, resiliency, security, and performance concerns.

***Keywords*—** Software Defined Networking, OpenDaylight, Mininet, Virtual Machine, Protocol

1 INTRODUCTION

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.

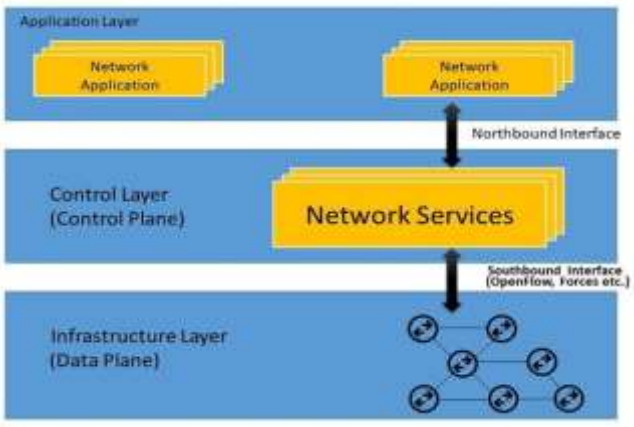

Fig 1.1 A three-layer software-defined networking (SDN) architecture.

Figure1.1 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 neighbourhood 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 History of the study

The records of SDN (Software Define Network) 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 stared 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.

1.2 Objectives

The objectives of this study are:

a. Implementation of Software Defined Network with Mininet.
b. Capturing open flow massage using Wireshark.
c. Incorporate open daylight user interface (DLUXUI) as a most advance tools of user friendliness.

1.3 Scope

SDN is an approach that introduces programmatic software defined network management to allow upper-layer

dynamic management of network resources which deals with network Virtualization related technologies, support for static and dynamic network provisioning technologies, support for tunnels – used to create overlay networks and Support for Network Functions Virtualization (NFV), L4-L7 Services, and data plane services:

a. Identifying relevant traffic and diverting it to / through individual services, service chains or Virtual Network Function forwarding graphs;
b. Service specific forwarding e.g., application aware QoS or service specific processing such as encryption.
c. Managing services and VNFs.
d. Enabling traditional and SDN controlled networks to coexist ("Hybrid" scenario).
e. Partitioning elements with some ports being traditionally controlled & others SDN controlled.
f. SDN control of an overlay network, with traditional control of the underlying network, and vice versa.
g. Partitioning of responsibilities, for example, traditional management/monitoring and SDN forwarding configuration or vice versa;
h. Switching traditionally and SDN Controlled networks to be interconnected (for both intra-domain and inter-domain scenarios).

Requirements and criteria to evaluate the SDN Architecture are defined. These can be grouped in the following categories:
a. Simplicity and Expressiveness
b. Applicability
c. Interworking
d. Interoperability
e. Scalability
f. Security
g. Resilience and Fault Tolerance
h. Management and Monitoring

1.4 Methodology

The objectives of this study are:

a) Implementation of Software Defined Network with Mininet.
b) Capturing open flow massage using Wireshark.
c) Incorporate open daylight user interface (DLUXUI) as a most advance tools of user friendliness Relevant details should be given including experimental design and the technique (s) used along with appropriate statistical methods used clearly along with the year of experimentation (field and laboratory).

Following 1.1 table describes difference between traditional and software defined networking types.

Table 1.1: Difference between the traditional Networking and SDN

Traditional Networking	Software Defined Networking
They are Static and inflexible networks and less useful for new business	Programmable networks in times of deployment and help new business venture
They are Hardware appliances	Configured using open software
They have distributed control plane	Logically centralized control plane
Use custom ASCIs and FPGAs	They use merchant silicon
They work using protocol	They use APIs to configure as per need.

1.5 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 that is including customizable 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 Software defined networking, 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 organize 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. The project is deals with several points:

a) OpenFlow: An OpenFlow Controller is a product application that oversees stream control in a SDN domain. 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 bestway for application traffic.

OpenFlow design comprises of three fundamental ideas:
a. The system is developed by OpenFlow-agreeable switches that create information plan
b. The control plane comprises of at least one OpenFlow controllers.
c. A secure control channel interfaces the switches with the control plane.

b) The essential bundle sending component with OpenFlow is outlined in Figure 1.2.

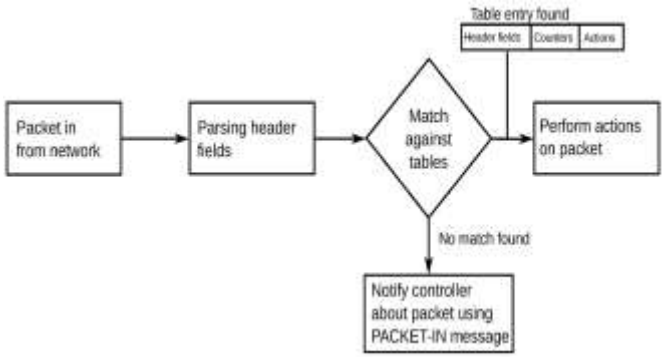

Fig 1.2 Basic packet forwarding with OpenFlow in a switch

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 buffered 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. 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.

c) 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. 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 that can be the set-in programming, utilizing programming dialects and basic working frameworks.
2. Applicability: Effectively executions done in models ought to be likewise which is usable in genuine several 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.

1.6 SDN Solution Providers Organizations

Some organizations that work with The SDN are –

Anuta Networks: Anuta Networks got its start in 2010 and has worked its way up the ranks and carved out a space for itself as a leader in SDN orchestration alongside the likes of Juniper Networks, HPE, and Huawei.

Pluribus Networks: Focusing on network analytics, data center interconnection and SDN solutions, Pluribus Networks offers open networking-based solutions which use a programmable SDN fabric for data centers and distributed cloud edge architecture. The company continues to be ranked as a leader in SDN and data center modernization by IT research firms.

Lumina Networks: San Jose, Calif.-based Lumina today offers its OpenDaylight-based Lumina SDN Controller, formerly the Brocade SDN Controller. The Lumina SDN Controller can be used to deliver 5G services.

Complete Communications: They provide SD-WAN (Software Defined Wide Area Networks) is a rapidly adopted technology trend providing cost-effective alternatives to MPLS networking solutions.

2 LITERATURE REVIEW

2.1 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 stared to be used in data networks. A PhD student from Stanford named Martin Casado developed something called Ethane back in 2006. This idea led to the first OpenFlow by 2011. Also in 2011, the Open Network Foundation was formed to standardize this future industry.
The Software Defined Networking (SDN) architecture was developed in the Open Networking Foundation (ONF) Architecture working group. It is intended to support standardization related to the broader area of software defined networking and affects not only the understanding and related work within ONF, but across a number of standards and/or open-source development organizations. This article provides a brief outline of the SDN architecture, its relation to Network Function Virtualization (NFV) and areas for further study [1].

2.2 Features of SDN

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.
The diverse nature of applications, devices, mobility of the endpoints has challenged IT to look beyond conventional applications for effective security policies, quality of service and performance. Software-Defined Networking (SDN) can be effectively used to address the challenges faced by service providers/IT in managing the dynamic nature of today's networks [2].

Network devices have always been considered as configurable black boxes until the emergence of software-defined networking (SDN). SDN enables the networks to be programmed according to the user requirements; furthermore, it allows the network to be easily modified to suit transient demands [3].

The SDN architecture enables the flexible support of a broad range of use cases and scenarios on a common infrastructure. Major components of SDN are resources and controllers. The SDN architecture defines the essence of an SDN controller as the continuing orchestration of all services against all resources according to a defined optimization policy.
Software Defined Network (SDN) is emerging technology having the centralized policy to design cloud network topology. The abstraction of control plane from data plane is isolating the control parameters from the flow of packets for making an efficient forwarding decision by programmable device [4].

5G promises faster and superior quality with better security guarantee in comparison to preceding technologies. The software defined networking (SDN) on the other hand is an enabling technology needed to actualise the huge promises of 5G network. With extensive network in 5G, a centralised controller approach in SDN has limitations related to the performance and scalability. Several studies have advocated the use and placement of multiple controllers to improve scalability in SDN [5].

3 COMPONENTS & ARCHITECTURE OF SDN

3.1 Basic Overview of SDN architecture

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 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. 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 another 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. Fig. 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”.

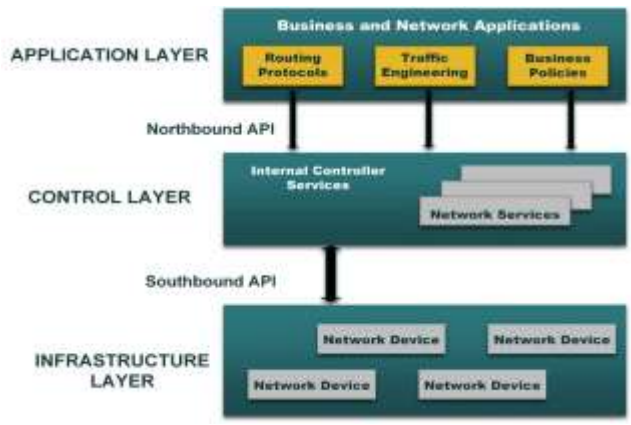

Fig 3.1 Basic SDN Components.

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.

3.2 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. In fig.3.2; the NE assets block consists of data sources, data sinks and forwarding and/or visitors processing engines, as properly as virtualize 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.

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.

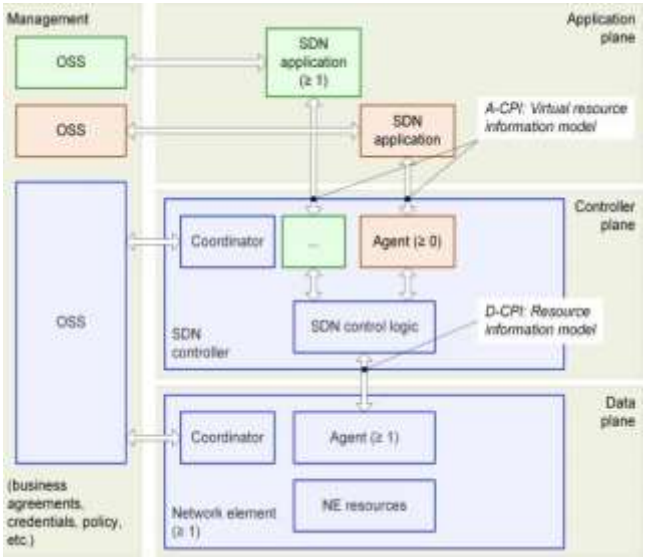

Fig 3.2 Network Element (NE) recourses details.

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 now not make self-sustaining forwarding decisions. 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-

a) Programmatic manipulate of all features exposed by way of the RDB.
b) Capabilities advertisement.
c) 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's. 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. Stack (the ''network brain'') jogging on a commodity hardware platform.

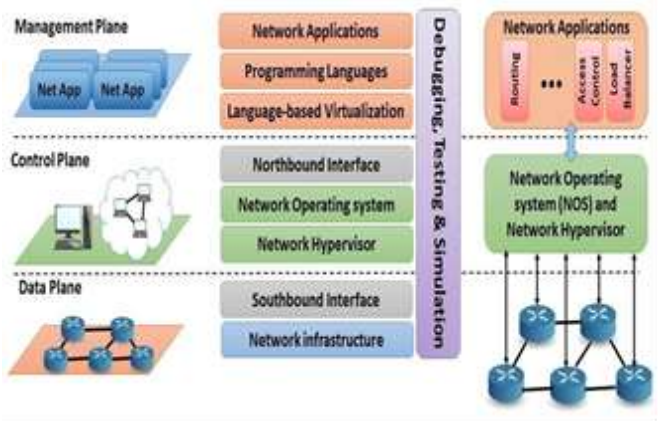

Fig 3.3 Architecture of plane, layers and architecture design.

In other words, these open interfaces enable controller entities to dynamically software heterogeneous forwarding devices. In an SDN/OpenFlow architecture, there are two

main elements, the controllers and the forwarding devices, as proven in Fig. 3.4. A facts plane system is a hardware or software element specialized in packet forwarding, whilst a controller is a software program

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 an 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.

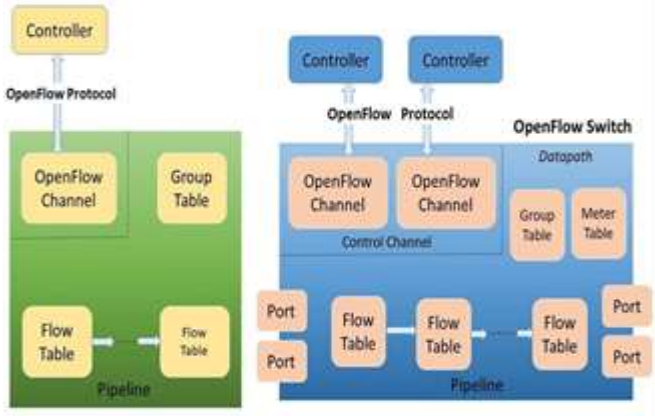

Fig 3.4 OpenFlow Qualify SDN.

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).

3.3 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 components of a Software Defined Network 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.

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

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 virtual machine, 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. 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. 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. 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 the network resources, but to comply with SDN principles, they need to either –

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

3.3.1 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 namely data plane manipulates characteristic (DPCF), coordinator, virtualizes, 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.

3.3.1.1 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.

3.3.1.2 Co-ordinate

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.

3.3.1.3 Virtualizes

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.

3.3.1.4 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.

3.3.1.5 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 gradually 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, and audit a launch administration cycle.

3.3.2 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, neighbour 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) A giant quantity of traffic that need to be processed.
b) Byte-or bit-oriented functions that do not simply lend themselves to capsulated, for example repetitive SDH multiplex area overhead
c) Survivability or continuity in case of controller failure or re-initialization.
d) Functionality oftentimes reachable in facts plane silicon, e.g., safety switching state machines, CCM counters and timer.
e) 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.

3.4 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:

a) Brocade Flow Optimize
b) Brocade Virtual router
c) Brocade Network advisor

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

a) HPE Network Optimizer
b) HPE Network protector
c) HPE Network visualizer
d) NEC UNC for HP SDN VAN Controller
e) Aricent SDN Load balancer
f) TechM server load balancer

As we quickly touched OpenFlow in preceding article, we would now cowl small print of southbound communication from control layer to infrastructure layer (network switches) thru 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. 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.

a) 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 3.5.
b) 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.
c) 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 visitor 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 behaviour, 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.

3.5 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.

4 OPENFLOW AND MININET 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).

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.

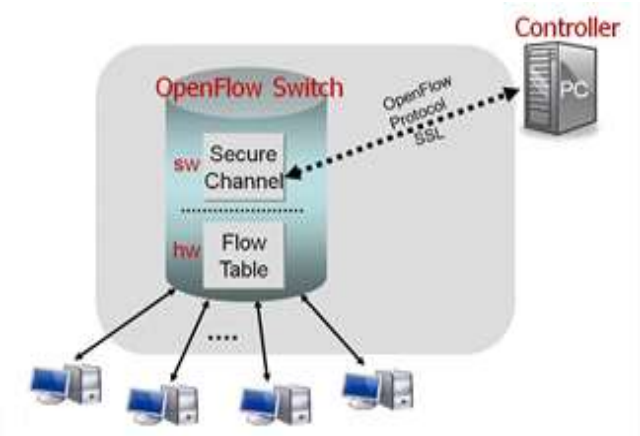

Fig 4.1 Open flow and open flow switch communicates over OpenFlow channel to an external controller

In fig. 3.1 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 cans 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 and 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:

a) With OpenFlow empowered switch, the SDN controller could course non basic/mass traffic on longer courses that are not completely used.
b) 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.
c) Traffic can be detached without the requirement for vlan's, the SDN controller of OpenFlow switch can simply decline certain associations.
d) 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.
e) It likewise takes into account the advancement of new administrations and thoughts all in programming on the SDN controller.

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.

4.7 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 limitations 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.Mininet supports the OpenFlow protocol, which provides an interface between the control plane and the data forwarding plane.
OpenFlow protocols are used to control the packet flow as per the API written on the controller. Mininet also has support for a variety of topologies in various pattern and ensures the availability of custom topologies. The CLI (command line interface) provided by Mininet is comfortable to use after a bit of practice.

Table 4.1 represents the requirements, versions and OS type for Mininet. 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.

Table 4.1 OS types, versions & other requirements

OS Type	OS Version	Virtual Machine	X Server	Terminal
Windows	7-10	VirtualBox	Xming	Putty
Windows	XP	VirtualBox	Xming	Putty
Mac	OS X 10.7-10.8	VirtualBox	Install XQartz	Terminal app
Mac	OS X 10.5-10.6 Leopard / Snow	VirtualBox	X11	Terminal app
Linux	Ubuntu 13.10-18.04	VirtualBox	X server	Gnome Terminal+SSH

Mininet is 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 has h1, h2 and switch s1.

5 PROPOSED METHODOLOGY

Enterprises, carriers, and service providers are being surrounded by a number of competing forces. The monumental growth in multimedia content, the explosion of cloud computing, the impact of increasing mobile usage, and continuing business pressures to reduce costs while revenues remain flat are all converging to wreak havoc on traditional business models. In recent global position Traditional Networking System is very difficult for managing and controlling which is not time efficient, cost efficient & has a cope of chance for vulnerabilities. In these circumstances we are going to propose a new network managing, designing and controlling technology which is called SDN.

5.1 Network Programmability

SDN enables network behaviour to be controlled by the software that resides beyond the networking devices that provide physical connectivity. As a result, network operators can tailor the behaviour of their networks to support new services, and even individual customers. By decoupling the hardware from the software, operators can introduce innovative, differentiated new services rapidly—free from the constraints of closed and proprietary platforms.

5.2 Logically Centralize Intelligence and Control Management

SDN is built on logically centralized network topologies, which enable intelligent control and management of network resources. Traditional network control methods are distributed. Devices function autonomously with limited awareness of the state of the network. With the kind of centralized control an SDN-based network provides, bandwidth management, restoration, security, and policies can be highly intelligent and optimized—and an organization gains a holistic view of the network.

5.3 Abstraction of the Network

Services and applications running on SDN technology are abstracted from the underlying technologies and hardware that provide physical connectivity from network control.

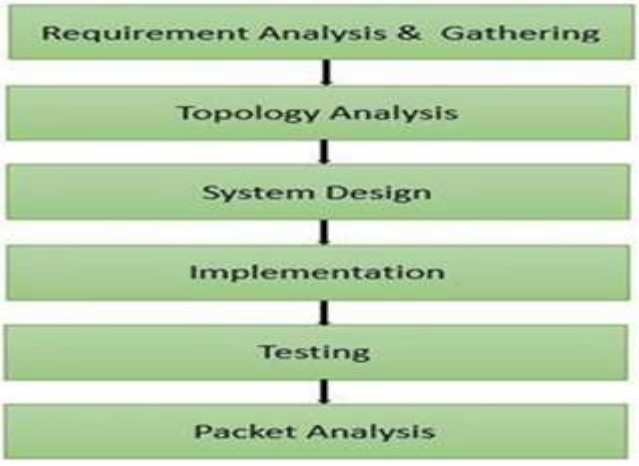

Fig 5.1 Proposed SDN methodology

Applications will interact with the network through APIs, instead of management interfaces tightly coupled to the hardware. In this purpose we have designed a proposed methodology in SDN which is belonging at fig. 5.1.

5.4 Proposed Methods over Traditional Networking

Typical networking devices main task is forwarding message like data-link frame such as Ethernet frames and we know that how switches and routers do that forwarding. Actually, those ideas are taken by programmable networking methods in our project. When we think about traditional network many ideas come to our mind. For instance, routers and switches physically connect to each other with cables, and with wireless to create networks. They forward messages: switches forward Ethernet frames, and routers forward packets. They use many different protocols to learn useful information such as routing protocols for learning network layer routes. Everything that devices do can be categorized as being in a particular plane. So, this section we are going to show the methods about how we do the network is a programmable, logically centralized and controlled and also about the abstraction of the network

5.4.1 Packet forwarding in Traditional Networking

We have already described about data plane in previous chapter. The term data plane refers to the tasks that a networking device does to forward a message. In other words, anything to do with receiving data, processing it, and forwarding that same data— whether we call the data a frame, a packet, or, more generically, a message—is part of the data plane. Here in fig. 5.2, it showed by a network diagram. If we focus on the Layer 3 logic for a moment, the host sends the packet (step 1) to its default router, R1. R1 does some processing on the received packet, makes a forwarding (routing) decision, and forwards the packet (step 2). Routers R2 and R3 also receive, process, and forward the packet (steps 3 and 4).

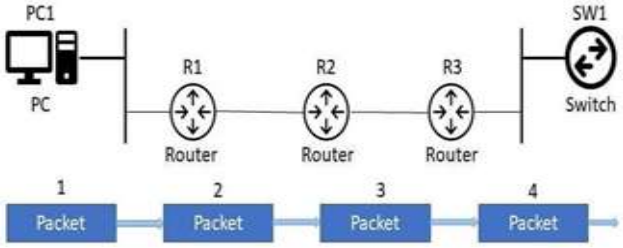

Fig 5.2 Data Plane Processing on Routers.

Next, take a moment to ponder the kinds of information that the data plane needs to know beforehand so that it can work properly. For instance, routers need IP routes in a routing table before the data plane can forward packets.

Layer 2 witches need entries in a MAC address table before they can forward Ethernet frames out the one best port to reach the destination. Switches must use Spanning Tree Protocol (STP) to limit which interfaces can be used for forwarding so that the data plane works well and does not loop frames forever. From one perspective, the

information supplied to the data plane controls what the data plane does. For instance, a router needs a route that matches a packet's destination address for the router to know how to route (forward) the packet. When a router's data plane tries to match the routing table and finds no matching route, the router discards the packet. And what controls the contents of the routing table? Various control plane processes. Traditional networks use both a distributed data plane and a distributed control plane. In other words, each device has a data plane and a control plane, and the network distributes those functions into each individual device, as shown in the example in fig. 5.3 Open Shortest Paths First (OSPF), control plane protocol runs on each router (R1, R2, and R3).

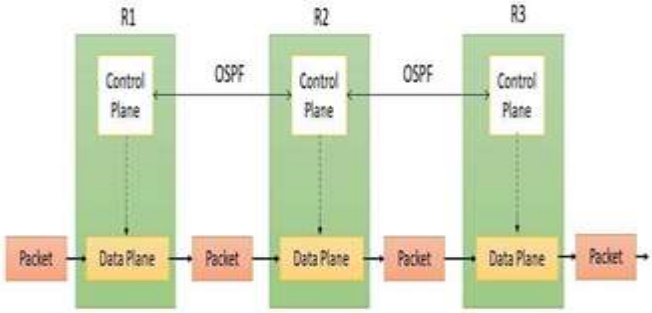

Fig 5.3 Control and Data Planes of Routers

OSPF on each router then adds to, removes from, and changes the IP routing table on each router. Once populated with useful routes, the data plane's IP routing table on each router can forward incoming packets, as shown from left to right across the bottom of the figure. Without the protocols and activities of the control plane, the data plane of traditional networking devices would not function well. Routers would be mostly useless without routes learned by a routing protocol. Without learning MAC table entries, a switch could still forward unicasts by flooding them, but doing that for all frames would create much more load on the local-area network (LAN) compared to normal switch operations. So, the data plane must rely on the control plane to provide useful information.

5.5 Ideas of Controllers and Centralized Control

So, what is the customization in this scenario? Here in fig.5.4 it is shown that the separation and centralization of control plane and data plane. A controller, or SDN controller, centralizes the control of the networking devices. The degree of control, and the type of control, varies widely. For instance, the controller can perform all control plane functions, replacing the devices' distributed control plane. Alternately, the controller can simply be aware of the ongoing work of the distributed data, control, and management planes on the devices, without changing how those operate. And the list goes on, with many variations. To better understand the idea of a controller, consider one specific case as shown in fig.5.4, in which one SDN controller centralizes all important control plane functions. First, the controller sits anywhere in the network that has IP reachability to the devices in the network. Each of the network devices still has a data plane; however, note that none of the devices has a control plane. In the variation of SDN as shown in figure 5.4, the controller directly programs the data plane entries into each device's tables.

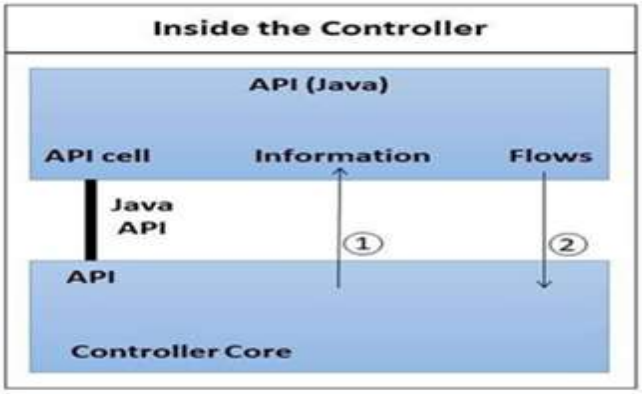

Fig 5.4 Centralized Control and Distributed Data Plane

The networking devices do not populate their forwarding tables with traditional distributed control plane processes. This particular controller happens to be written in Java and has a Java-based native API. Anyone—the same vendor as the controller vendor, another company, or even we—can write an app that runs on this same operating system that uses the controller's Java API. By using that API to exchange data with the controller, the application can learn information about the network. The application can also program flows in the network—that is, ask the controller to add the specific match/action logic (flows) into the forwarding tables of the networking devices. So, the benefits that actually help us to think that the replacement of traditional networking by software defined networking:

a) Centralized network provisioning.
b) Holistic enterprise management.
c) More granular security.
d) Lower operating cost.
e) Hardware savings and reduced capital expenditures.
f) Cloud abstraction.
g) Guaranteed content delivery.

6 SDN TREE TOPOLOGY IMPLEMENTATION USING MININET

6.1 Required Tools

In this part we broadly discussed about the implementation of SDN in Minnet along with several tools which needed to be interconnected to get the output, such as VirtualBox, Ubuntu server 16.04, OpenDaylight, Mininet, Wireshark, Xming, PuTTY.

6.1.1 Virtualbox

Users of virtual Box can load more than one guest OS below a single host-system (host OS). East guest can be started, paused and stopped independently inside its personal virtual computing device (VM). The user can independently configure each 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 OS 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 VM.

6.1.2 Ubuntu Server 16.04

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.

6.1.3 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.

6.1.4 Wireshark

Wireshark is a data capturing application that 'understands' the structure 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.

6.1.5 Xming

Xming provides an 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.

6.1.6 PuTTY

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

6.2 Simulation

6.2.1 Open Daylight

We have named the virtual machine ODL. To configure it uses two CPUs and 2 GB RAM. This is the minimum configuration to support OpenDaylight. Then add a host-only network adapter to the VM. The OpenDaylight Project arose out of the SDN movement, with a clear focus on network programmability. An SDN controller manages flow control to the switches/routers (via southbound APIs) and the applications and business logic (via northbound APIs) to deploy intelligent networks. They consolidate and mediate between different controller domains using common application interfaces.

IP commands:
sudo ifconfig
Now we can check the IP address assign into enp0s8 in fig.6.1.

Fig 6.1 IP address of OpenDaylight.

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-sr2.tar.gz
This command creates the folder named distribution-kara-f0.4.0-Beryllium-sr2 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.
To 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-sr2
$. /bin/karaf

6.2.2 Mininet

At first to set the Mininet we download the Mininet Virtual Box. After configuring both Mininet and OpenDaylight now our task is to create a SDN tree topology.
So, in Mininet we have found the IP address is 192.168.56.105 shown in fig. 6.2.

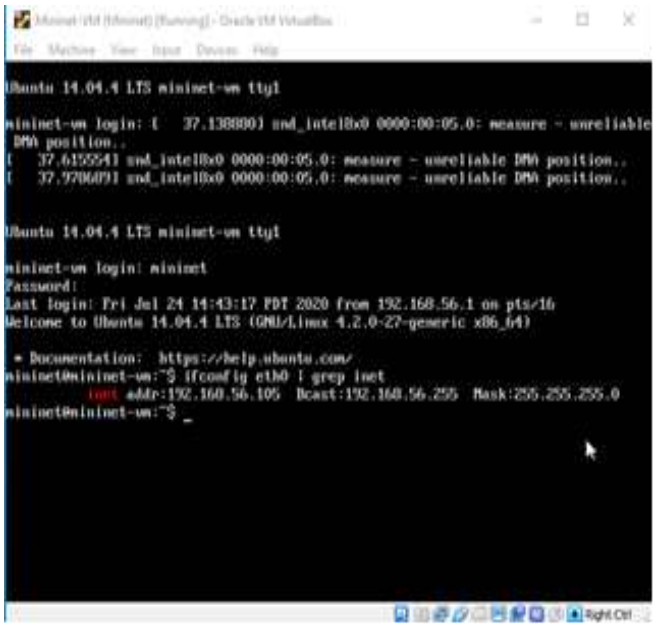

Fig 6.2 IP address of Mininet.

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 192.168.56.105 which is in the address range assigned by the Virtual Box host-only network in DHCP server.
So, we know that we need to use IP address 192.168.56.105 to access applications running on this virtual machine.

6.2.3 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 192.168.56.107 and the port defined by the application is 8181.

Here, the user name is "admin" and password also is "admin" by default.
Now we see our network topology in the OpenDaylight controller's topology tab in fig. 6.4.

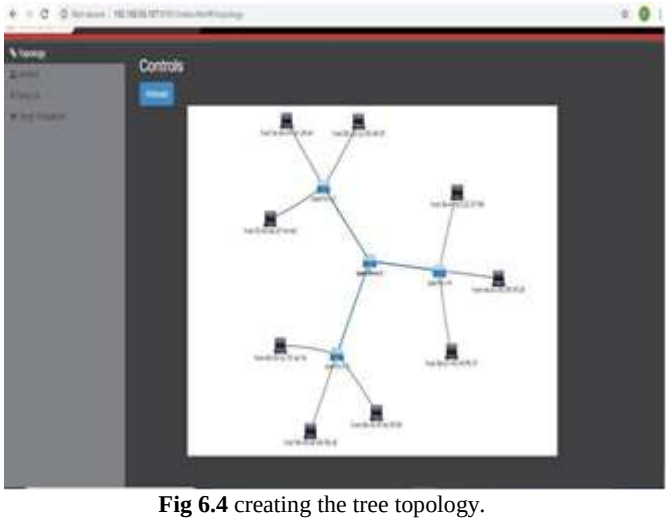

Fig 6.4 creating the tree topology.

6.2.4 Capturing Open Flow massages (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, now 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 in fig.6.5.

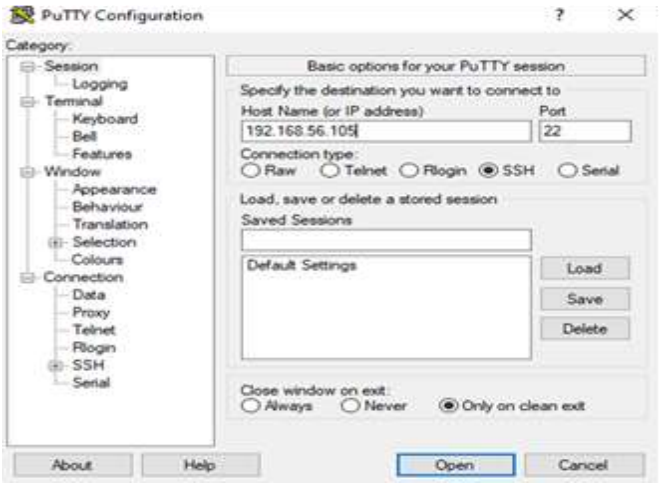

Fig 6.5 PuTTY configuration.

Then in the terminal for IP address 192.168.56.105 that we get, write the following command for running the Wireshark:

Wireshark starting command:
$ sudo wireshark &
A warning dialog is shown but we 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.

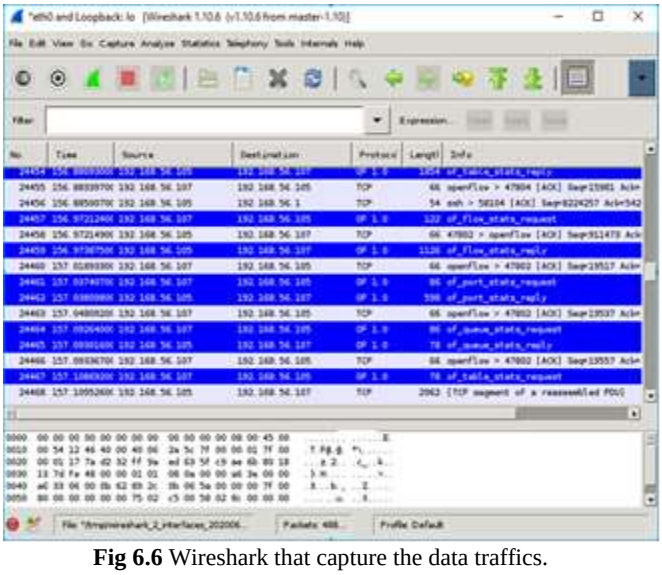

Fig 6.6 Wireshark that capture the data traffics.

Now we will see only OpenFlow messages in the Wireshark display, as shown below in fig. 6.6.
In Wireshark protocol hierarchy statistics that shown below in fig. 6.7.

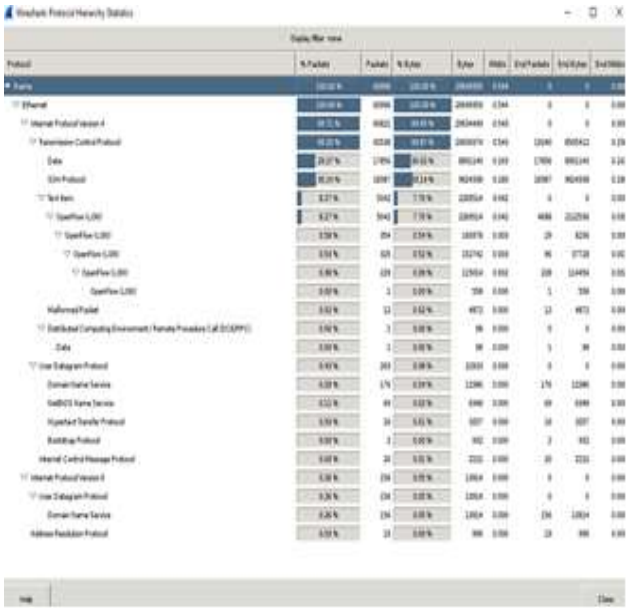

Fig 6.7 Wireshark analysis about SDN protocol hierarchy.

Here in fig 6.7, we can see –
Protocol: The protocol name.
Percent Packets: The percentage of protocol packets from the total captured packets.
Packets: The number of protocol packets from the total captured packets. Percent Bytes: The percentage of protocol bytes from the total captured packets.
Bytes: The number of protocol bytes from the total captured packets.
Bit/s: The bandwidth of this protocol, in relation to the capture time.
End Packets: The absolute number of packets of this protocol (for the highest protocol in the decode file).
End Bytes: The absolute number of bytes of this protocol (for the highest protocol in the decode file).
End Bit/s: The bandwidth of this protocol, relative to the capture packets and time (for the highest protocol in the decode file).

6.3 Other Simulators

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

a) Mininet-Wifi (developers added virtualized WiFi stations and access points based on standard Linux wireless drivers to extend Mininet's functionality).
b) KNet (KNet builds the topology of the virtual network with switches, hosts, routers and servers.
c) 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).

Other than OpenDaylight controller some other OpenFlow controllers also available as they are,

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

7 CONCLUSION AND FUTURE WORK

7.1 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 Datacentre technology. It has visionary approach to IT networking that easily and quickly is now the preferred style of network management. Software-Defined Networking (SDN) is a recently evolving networking architecture that focuses on the separation of control and data planes. Unlike traditional switches, SDN switches include flow tables that are remotely controlled by a separate software application, the controller. SDN is not completely new; it formulates an architecture on top of several good practices.

7.2 Future Work

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, etc., handle bundles by permitting unlimited oversight of the tenets set into system gadgets from a focal reassure. 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 centre. 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" levelled 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 it's fair an excessive amount to send. SDN need much greater advancement and evidence of being a protected and deployable innovation. 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 endeavours 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. SDN reception appears like an easy decision. It's still in the beginning times, be that as it may, of picking up section to datacentres. 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 datacentres. 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) endeavours, others are now sending and completely working those SDNs. The more IT experts think about SDN and how they can profit, the rate of organization will absolutely get. 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.

REFERENCES

[1] S. Schaller, D. Hood, "*Software defined networking architecture standardization*" Computer Standards & Interfaces Vol.**54**, Issue.**P4**, pp.**197–202**, **2017**.

[2] V. R. Tadinada, "Software Defined Networking: Redefining the Future of Internet in IoT and Cloud Era" in 2014 International Conference on Future Internet of Things and Cloud, Barcelona, **Spain**, **27-29** Aug, **2014**.

[3] Celio Trois, M. D. Del Fabro, L. C. E. de Bona and M. Martinello, "*A Survey on SDN Programming Languages: Toward a Taxonomy*," IEEE Communications Surveys & Tutorials, Vol.**18**, Issue.**4**, pp.**2687–2712**, **2016**.

[4] Ankita V. Mandekar, K. Chandramouli, "*Centralization of Network Using Openflow Protocol*" Indian journal of science and technology Vol.**8**, Issue.**S2**, pp.**165-170**, Jan **2015**.

[5] S. K. S. Yusof, P. E. Numan, K. M. Yusof, J. B. Din, M. N. B. Marsono, A. J Onumanyi, "*Software-Defined Networking (SDN) and 5G Network: The Role of Controller Placement for Scalable Control Plane*" in 2020 IEEE International RF and Microwave Conference (RFM), Kuala Lumpur, **Malaysia**, **14-16** Dec, **2020**.

AUTHORS PROFILE

Mr. S M Byazid Khan pursed his Bachelor of Science degree from Bangladesh University of Engineering and Technology in 2000 and Master's degree in Mechanical Engineering from Dhaka University of Engineering and Technology in 2019. He is currently pursuing his Ph.D. He is also serving as an officer in Bangladesh Army since 1990. His main research work focus in Mechanical Engineering, Information Technology, Network Security etc.

Mr. M. Mofazzel Hossain Shojib pursed his Bachelor of Science in Electrical and Electronics Engineering from Northern University Bangladesh in 2016 and Master of Science in Information and Communication Technology from Bangladesh University of Professionals (BUP) in 2019. He is experienced in Software development, Server Administration, Database & Cyber Security. His knowledge of expertise and research focus is in Information Technology, Cyber Security, Cloud Computing and Big Data Analytics.

