International Journal of Advance Research in Science and Engineering
Vol. No.6, Issue No. 05, May 2017
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

PROGRAMMABLE NETWORKS USING SOFTWARE DEFINED NETWORKING

Harkirat Kaur[1], Novjot Jyoti[2]

[1,2] *Computer Science and Engg, North West Group of Institutions, Moga (INDIA)*

ABSTRACT

Control plane and Data plane are two elements of network devices. Both the elements, software (control plane) and hardware (data plane) are tightly integrated into same device. It means if we have a firewall device it perform the functionality of only firewall. Any type of modification in the behavior is very error prone task. In Software Defined Networking, the software part of the device is shifted to central place. This shifted portion is called SDN controller. Now the device is simply forwarding device. It is a simple merchant silicon box that contains no functionality. The behavior of device depends upon the controller. According to application running on controller, the dumb device act as firewall, load balancer and switch. SDN enables innovation in the network. The two planes or elements communicate with each other using OpenFlow protocol. In this paper, we discuss about components of SDN, experiment tools, applications and use cases.

Keywords*: *OpenFlow Protocol, Software Defined Networking, ForCES, POX, Mininet

I. INTRODUCTION

There are number of devices available in the traditional network such as firewall, routers, switches, load balancer, intrusion detection system, and intrusion prevention system. In these devices data plane that actually forward the packet and control plane that control the behavior of data plane are tightly coupled into same device. So modification in existing behavior of such devices is very complex and error prone task. Secondly vendors write the code and there is long delay to introducing new features in these devices. So network administrator configures each device separately as specified by vendor. It is also very difficult because different configuration is required even for the products are from same vendor. This means innovation in traditional networks is very complex, error prone, time consuming process. The operational and capital cost is also very high.

To remove the problems of traditional networks, Software Defined Networking (SDN) gained very much popularity in the field of networking. Basically Software Defined Networking is a new networking approach in which the data plane is decoupled from control plane as shown in Fig. 1. Software Defined Networking simplifies the network management and also enables innovation in the network [1]. In software defined networking the shifted control plane is called controller. This centralized controller manages the behavior of the data plane. Now the data plane is a simple packet forwarding device that does not perform any function itself. The data plane performs the action on the packet specified by the control plane. OpenFlow and ForCES are protocols that are used for communication between data plane and centralized control plane. The Open Network Foundation (ONF) [2] is an organization that promotes SDN. The SDN is relatively new field and growing at fast speed. There are number of challenges are to be addressed [3]. Due to separation between two planes it is

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

very easy to deployed new applications. We can write an application in any programming language according to controller and run that application on controller. According to the logic written in that application, the same data plane act as a firewall, load balancer, switch, and intrusion detection system. So by writing applications we can convert dumb device into powerful networking device.

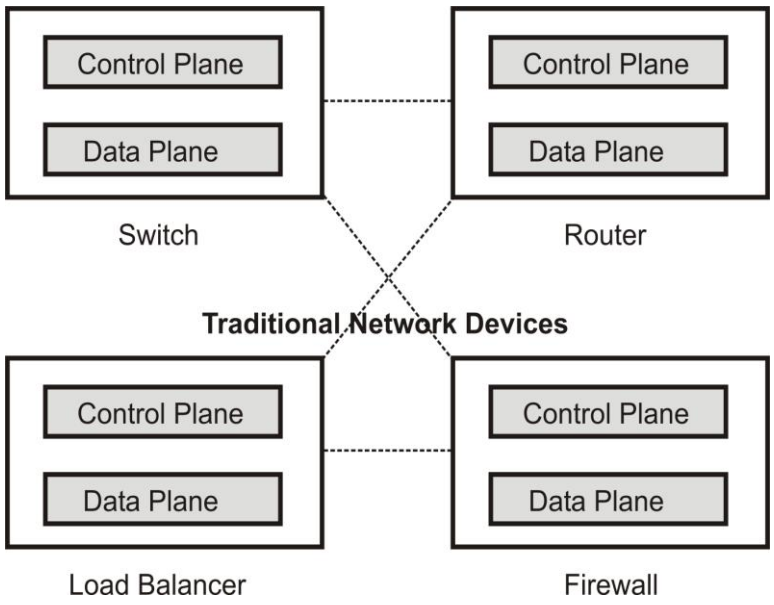

Fig. 1 traditional network

II. SOFTWARE DEFINED NETWORKING ARCHITECTURE

There are number of components such as forwarding device (data plane), controller (control plane), southbound interface (interface between data plane and control plane), northbound interface (interface between control plane and applications) in software defined network architecture [4]. By writing applications at management plane and run that applications on control plane, we can convert the dumb device or forwarding plane into powerful and low cost firewall, load balancer, switch.

2.1 Forwarding Device

The main task of forwarding device is to forward the incoming packet to particular destination. Traditional network consist of different types of these network devices such as load balancer, firewall, switch. If we have a firewall device then it acts only as a firewall and if we have load balancer device it perform the functionality of only load balancing. The software and hardware part of such devices are tightly coupled. But in SDN the forwarding devices are simply dumb devices that do not perform any task [5]. According to instructions given by the control plane these same devices act as a firewall, load balancer or switch as shown in Fig. 2. OpenFlow protocol is used for communication between the two planes. These devices are also called openflow switches.

There are basically two types of openflow switches available: hybrid switches and pure switches. Pure switch completely support the SDN architecture. But in Hybrid switches traditional network features plus openflow features are supported simultaneously. Because SDN is very new concept in networking so till now only hybrid switches are available in the market.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

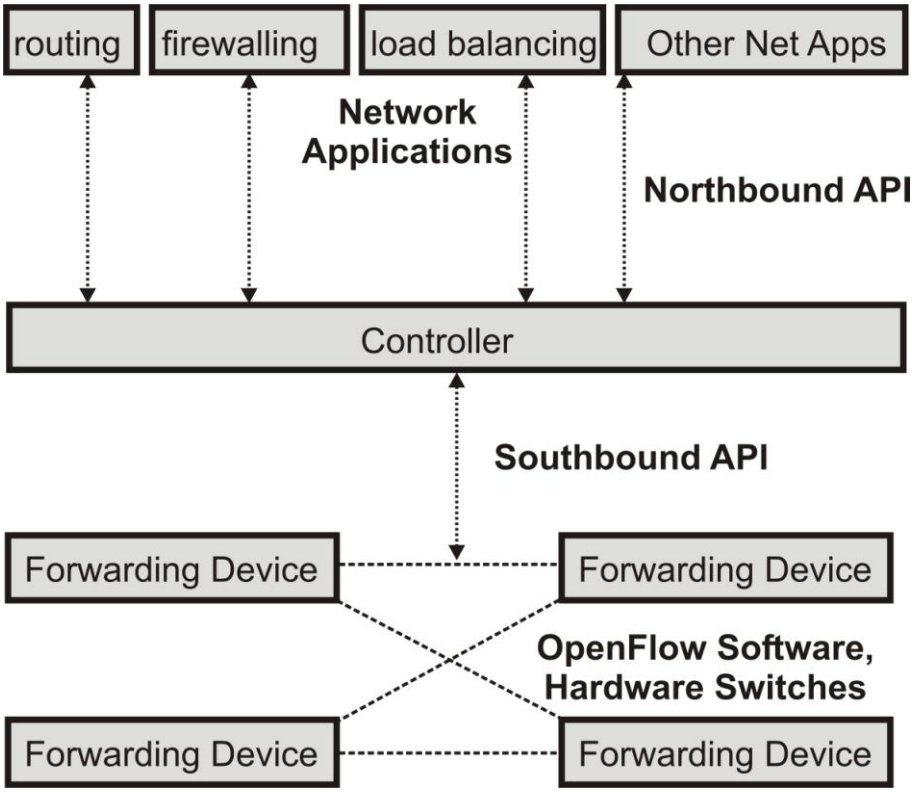

Fig. 2 SDN architecture

2.2 Southbound Interface

ForCES and OpenFlow are two southbound interfaces in SDN architecture that are used for communication between openflow switches and controller.

2.2.1 ForCES

Forwarding and Control Element Separation (ForCES) consists of two elements. One is forwarding element (FE) and the other is control element (CE). The forwarding element is same as data plane in Openflow that only forward the packet. The control element is same as control plane that instructs to the FE what action is to be performed. ForCES protocol use master slave model in which control element is master and forwarding element is slave. Logical Function Block is main part of ForCES architecture.

2.2.2 OpenFlow

SDN devices use openflow protocol for communication between separated data plane and control plane. An OpenFlow switch consists of flow table which contains number of rules [6]. There are number of fields in every flow rule. The first field in flow table is matching fields that are used to identify the packet src mac, destination mac, src ip, destination ip, src port and destination port as shown in Table I. The second field is actions that specify what to do with the packet like forward, drop the packet as shown in Table II.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

TABLE I MATCH FIELDS

Ingress Port
Source MAC
Destination MAC
Ethernet type
VLAN id
VLAN priority
Source IP
Destination IP
IP Protocol
IP TOS bits
Source Port
Destination Port

TABLE II ACTIONS

Action	**Description**
Forward	Forward packets to a specified port.
Flood	Forward packets to all ports except incoming port.
Modify Field	Modify the header fields of packet.
Drop	Drop the packet.
Controller	Send the packet to the controller.
Local	Send packet to local switch.

Last field is counter that is used for counting the number of packets and flows. When the packet reaches at the switch, switch checks its flow entries in flow table. If the packet matches then action is to be performed on that packet according to the action specified otherwise drop the packet or forward the packet to the controller. Secure channel is used by the openflow switch for communication with the controller.

2.3 Controller

In software defined network the controller is the brain of the network because it manages the openflow switches using openflow protocol. It is the main part of the SDN architecture because all the logic of the openflow device is placed on the controller. Controller inserts flow rules into flow table of switch according to the application written on top of the controller. Controller is placed between openflow switch and applications [7]. There are various types of controller available such as POX [8], NOX [9], Floodlight [10], Beacon [11], Trema [12] and

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

all these controllers are written in different languages as shown in Table III. All type of traffic between openflow devices and applications passes through the controller.

TABLE III SDN CONTROLLERS

Name	Language	Description
Ovs	C	A reference controller
Nox	C++	The first OpenFlow controller
Pox	Python	Open source SDN controller
Ryu	Python	Ryu is a component-based software defined networking framework
Trema	Ruby, C	A framework for developing OpenFlow controller
Floodlight	Java	OpenFlow controller that work with physical and virtual OpenFlow switches.
Beacon	Java	A cross platform, modular OpenFlow controller

2.3.1 Distributed controller

When the size of network increases more requests are send to the controller. In this case single controller may fail to handle all incoming requests. To remove this problem distributed control plane is used in which control plane elements is physically distributed into number of places. Onix, Kandoo, HyperFlow are the examples of distributed control planes.

2.3.2 Reactive vs Proactive

There are two methods available by which controller sends the instructions to openflow switch about how to handle particular packet. In Reactive approach, packet is matched against flow table entry, if first time packet does not match against flow rules so that packet is send to controller. Then controller inserts flow entries into flow table of switch. In Proactive approach, controller proactively install the rules into flow table of switch so that packets are handled by openflow switch itself according to flow rules.

2.4 Northbound Interface

It is an interface between controller and applications. By using this interface we can develop number of applications such as router, switch, firewall, load balancer. On the other hand, a southbound interface provides a communication between the forwarding device and controller.

III. SDN EXPERIMENTAL TOOLS

For testing applications developed in software defined networking we need some tools such as emulators and simulators. Mininet is an emulator tool by which we can build a large network. The large network can contain very large number of hosts, switches and controller [13]. The choice is very good if our network should contain hundred of hosts, virtual openflow switches and a controller. The disadvantage of mininet is that it is limited with the resources of single PC. It has many advantages over other tools like simulators and testbeds. It is very cost effective, easily configurable tool. It is very close to real environment. The application that runs in mininet can also run in real environment. There are number of simulator such as EstiNet and NS-3. We can also use

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

testbeds such as GENI [14], VENI [15] but are not very flexible.

Mininet emulator is a tool that is widely used for SDN experiments. Mininet is installed on single PC and we can create an experiment setup consisting of hosts, switches and controller with the limited resources of single PC. Basically there are testbeds also available but configuration of testbeds is very complex task. Mininet is easily configurable, quickly and easily available tool as compared to testbeds. Numbers of topologies are already available in mininet but we can also create topology according to our needs by writing a code in python language. If we run a code in simulators, it cannot run in real lab setup. But mininet code can run without modification in both environments.

IV. SDN APPLICATIONS

In this section we will provide some application examples on using SDN and OpenFlow.

4.1 SDN in Research

It is very difficult to test new strategies and innovative ideas on existing networks for solving problems. Internet is bringing new challenges with every passing day. SDN helps us in testing future internet applications without disturbing existing network. SDN separates hardware from the software. This enables us in experimenting new architecture on internet.

4.2 Data Center and SDN

There Computing Virtualization has made it extremely easy to automatic provision of servers in data centers. Storage virtualization such as SAN has made automatic provisioning of storage very easy. But one main component that is networking has been the bottleneck. SDN has changed this by allowing automatic provisioning of networking thus enabling the concept of SDDC (Software Defined Data Center). SDN is going to be the integral part of SDDC.

4.4 SDN in the WAN and LAN

With the increased usage of BYOD and SAAS, it is getting very difficult to manage WAN. But SDN is helping in solving these issues by allowing dynamic provisioning of network resources. SDN is also enabling easy combined wired and wireless network management, BYOD control and better security.

4.5 SDN in security

It allows greater access control mechanism and increased security. SDN offers more visibility thus allowing better network monitoring, traffic engineering and deep packet analysis.

4.6 SDN in cloud

A new type of cloud service known as NAAS (Network as a service) is emerging. It will allow dynamic provisioning of Routers, Switches, Firewalls and Load Balancers.

V. CONCLUSION

Traditional Network devices are complex and difficult to manage. Basically, main problem with traditional network devices is that software is bundled with hardware and interfaces are vendor specific like in the case of Cisco devices where IOS operating system and hardware is tightly coupled into the same device. Therefore vendor write the code and there are long delay in introducing new features in these devices. Software defined

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

networking solve these problems by decoupling the control plane from data plane. Openflow is the protocol that is used for communication between control plane and data plane. According to the application written on the control plane, we can convert simple openflow device into firewall device, load balancer device or any other device. SDN is gaining lot of attention from academia and industry. It is the biggest that has happen in last thirty years in networking field.

REFERENCES

[1] Kim, Hyojoon, and Nick Feamster. "Improving network management with software defined networking." *Communications Magazine, IEEE* 51, no. 2 (2013): 114-119.

[2] Schneider, Fabian, Takashi Egawa, Sibylle Schaller, Shin-ichiro Hayano, Marcus Schöller, and Frank Zdarsky. "Standardizations of SDN and ITs practical implementation." *NEC Technical Journal, Special Issue on SDN and Its Impact on Advanced ICT Systems* 8, no. 2 (2014).

[3] Yeganeh, Soheil Hassas, Amin Tootoonchian, and Yashar Ganjali. "On scalability of software-defined networking." *Communications magazine, IEEE* 51, no. 2 (2013): 136-141.

[4] Jarraya, Yosr, Taous Madi, and Mourad Debbabi. "A survey and a layered taxonomy of software-defined networking." *Communications Surveys & Tutorials, IEEE* 16, no. 4 (2014): 1955-1980.

[5] Rowshanrad, Shiva, Sahar Namvarasl, Vajihe Abdi, Maryam Hajizadeh, and Manijeh Keshtgary. "A survey on SDN, the future of networking." *Journal of Advanced Computer Science & Technology* 3, no. 2 (2014): 232.

[6] Naous, Jad, David Erickson, G. Adam Covington, Guido Appenzeller, and Nick McKeown. "Implementing an openflow switch on the NetFPGA platform." In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp. 1-9. ACM, 2008.

[7] Shalimov, Alexander, Dmitry Zuikov, Daria Zimarina, Vasily Pashkov, and Ruslan Smeliansky. "Advanced study of SDN/OpenFlow controllers." In *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, p. 1. ACM, 2013.

[8] Khondoker, Rahamatullah, Adel Zaalouk, Ronald Marx, and Kpatcha Bayarou. "Feature-based comparison and selection of Software Defined Networking (SDN) controllers." In *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, pp. 1-7. IEEE, 2014.

[9] Shah, Syed Ahmar, Jawad Faiz, Maham Farooq, Aamir Shafi, and Syed Atif Mehdi. "An architectural evaluation of SDN controllers." In *Communications (ICC), 2013 IEEE International Conference on*, pp. 3504-3508. IEEE, 2013.

[10] Wallner, Ryan, and Robert Cannistra. "An SDN approach: quality of service using big switch's floodlight open-source controller." *Proceedings of the Asia-Pacific Advanced Network* 35 (2013): 14-19.

[11] Monaco, Matthew, Oliver Michel, and Eric Keller. "Applying operating system principles to SDN controller design." In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, p. 2. ACM, 2013.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

[12] Ivashchenko, Pavel, Alexander Shalimov, and Ruslan Smeliansky. "High performance in-kernel SDN/OpenFlow controller." *Proceedings of the 2014 Open Networking Summit Research Track, USENIX* (2014).

[13] Wang, Shie-Yuan. "Comparison of SDN openflow network simulator and emulators: EstiNet vs. Mininet." In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pp. 1-6. IEEE, 2014.

[14] Berman, Mark, Jeffrey S. Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. "GENI: A federated testbed for innovative network experiments." *Computer Networks* 61 (2014): 5-23.

[15] Sonkoly, Balázs, Felicián Németh, Levente Csikor, László Gulyás, and András Gulyás. "SDN based testbeds for evaluating and promoting multipath TCP." In *Communications (ICC), 2014 IEEE International Conference on*, pp. 3044-3050. IEEE, 2014.