International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 13, Number 6 (2017), pp. 1509-1520

http://www.ripublication.com

Deployment of SDN Based Openflow System and Related Network Topologies

Varun S. Moruse[1] and A. A. Manjrekar[2]
[1](M. Tech Student, Computer Science and Technology, Department of Technology, Shivaji University, Kolhapur, Maharashtra,)
[2](Assistant Professor, Computer Science and Technology, Department of Technology, Shivaji University, Kolhapur, Maharashtra,)

Abstract

Today's network switches require reconfigurations from time to time, as they have data forwarding plane and control plane positioned together in same devices. This results in complex working of switches, with inefficient performance in terms of repeated functionality and delayed delivery. This system based on SDN gives an idea to separate packet forwarding functionality from control functionality from such devices which results in efficient network communication. OpenFlow is one of the SDN technique provides network architectural paradigm to networking. By decoupling the control plane from data plane, SDN can achieve repolicing. Using OpenFlow, a network administrator can modify the nature of network by writing simple software programs, which manipulate the logical map of network. This system elaborates use of an OpenFlow switch in network with various topologies having single, multiple switches.

Keywords: Central Controller, Datapath, Forwarding Element, OpenFlow, Software Defined Network

1. INTRODUCTION

Today, for all infrastructures in society, network has become a critical component. The problem is industry and designs of network components have not kept pace with ever growing requirements. The network components like switches, routers, and other devices need to implement an ever increasing number of distributed protocols standardized by IETF and use closed and proprietary interfaces within, which made the devices complex.

In existing environment, network operators, researchers, and even vendors face difficulties to innovate. As per the application set, customization and optimization cannot be carries out easily by operators relevant to their business.

General observations after analysing the existing systems are,- (a) networks continue to have serious known problems with security, robustness, manageability, mobility and evolvability that have not been successfully addressed; (b) their capital costs have not been reducing fast enough and operational costs have been growing, putting excessive pressures on network operators; and (c) network operators find it difficult to introduce new revenue generating services on their expensive infrastructures.

In a conventional network for packet forwarding, upon the reception of a packet by a routing device, uses a set of rules embedded in its firmware to find the destination device as well as the routing path for that packet. This operation takes place in expensive routing devices. More expensive routing devices can treat different packet types in different manners based on their nature and contents. Special algorithms are implemented on dedicated hardware devices to monitor and control the data flow in the conventional network systems. In general the routing algorithms and sets of rules are implemented in dedicated hardware components are designed for performing specific operations. OpenFlow gives us the opportunity to customize policy rules for various paths.

The SDN notion introduced OpenFlow, which advocates the idea of decoupling the control and data paths in separate planes. It focuses on exploitation of the common set of functions. A network operating system running on this control plane is anticipated to provide necessary measures for scalability and reliability in order to stand against the gigantic traffic pumped by the network.

2. RELATED WORK

The architecture of today's Internet is relatively stagnant due to the designing principle of "Keeping the simplicity of network while leaving the complex processing tasks to hosts" [1]. The functions of the application-layer have been greatly enriched because the applications on hosts can be flexibly modified and deployed but the network devices have become like opaque black-boxes because of the lack of openness in the network-layer. Apparently today's networks have become closed, inflexible and unmodifiable. [2].

Today, there is almost no practical way to experiment with new network protocols in sufficiently realistic settings to gain the confidence needed for their widespread deployment. The result is, most new concepts and intellection cannot be tried and tested by the networking research community. Having recognized the problem that the networking community is hard at work developing programmable networks, such as GENI [3] a proposed nationwide research facility for experimenting with new network architectures and distributed systems.

In the current routers, implementations of the control and forwarding functions are intertwined deeply in many ways. Communication between the control processors and the forwarding line cards is not based on any standard mechanism which makes it impossible to interchange control processors and forwarding elements. [4]

The existing Internet architecture must be checked, reviewed and several research groups are engaged already in this process. OpenFlow, a part of SDN, represents an extraordinary opportunity to rethink about existing computer networks, enabling the novel design and deployment of a future Internet. [5]

A few open software platforms already exist, but do not have the performance or port-density we need. The simplest example is a PC with several network interfaces and an operating system. [6] All well-known operating systems support routing of packets between interfaces, and open-source implementations of routing protocols exist (e.g., as part of the Linux distribution, or from XORP). The problem lies in performance: A PC can neither support the number of ports needed for a college wiring closet. [8]

There are lots of similarities between OpenFlow and previous attempts to provide an external interface for a control plane for locally controlled switches and routers. They're all bit different in some manner. There have also been attempts to separate the data plane from the control plane in the past, and, after all, there are many networks, like telephony networks, that already work that way. Here the difference is timeliness. Now days, every network service provider company pressing need to optimize the behaviour of their networks so they can differentiate their solution from others. [9]

Table 1 shows the difference between traditional networking and SDN based networking based on various factors.

Table I: Comparisons between SDN and Conventional Networking

	Traditional Networking	Software Defined Networking
Nature	Static	Programmable
Basis	Hardware	Software.
Control Plane	Distributed	logically centralized
Material used	Custom ASICs and FPGAs	Merchant silicon.
Work using	Protocols	APIs
Time consuming	High	Less
Error prone	High	Less
Expertise required	High level	Low level
Flexibility and agility	Little	More

As the time chart below (Fig 1) illustrates, we are early in this technology flow. The SDN started around 2009 and will probably last for another decade or more. Networking technology needs to change to cope with the explosion of IP traffic.

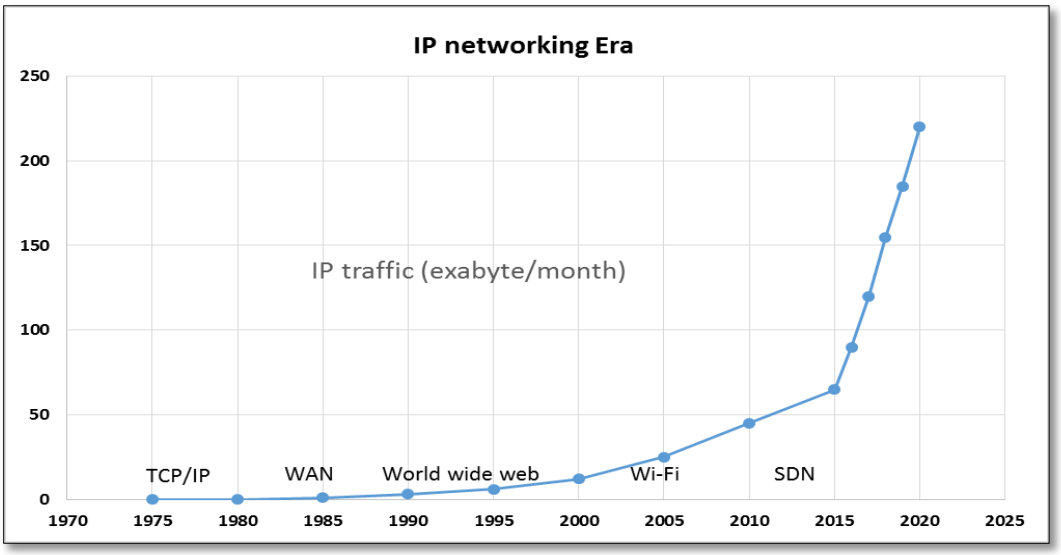


Fig 1: Time chart of IP traffic in Exabyte/month versus years, as the technology beneath changed

3. SDN BASED OPENFLOW SYSTEM

This software system is varied on the basis of number of switches and hosts, with two modules as follows: [7]

A. Forwarding Element (FE)

B. Central Controller (CC)

3.1 Single switch based system

The architecture of the system is shown in Fig 1.

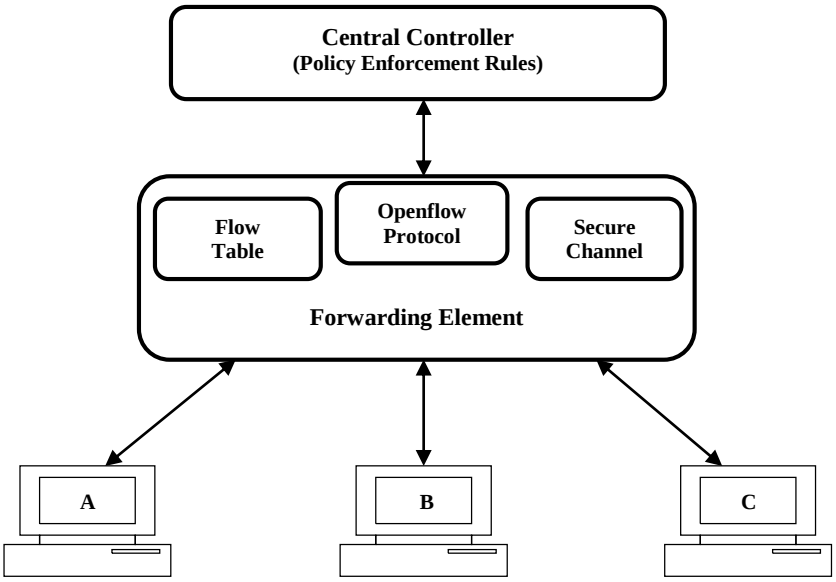


Fig 2: Architecture of single switch based system

A. Forwarding Element (FE)/ Switch

The forwarding element is the OpenFlow Switch itself, its main function is to forward the packets to particular destination. The data path portion resides on the FE. It will only forward the packet to the controller. Network administrator can control packet flow by selecting the routes for the packets and then process it.

The main functions of FE are as follows:

- Address lookup and mapping
- Policy enforcement
- Tunneling

The Forwarding element has following important entities:

- Flow Table
- A Secure Channel
- The OpenFlow Protocol

3.1.1 Flow Table

Flow means combination of rules, actions and statistics. Flow table describes the components of flow table entries and the process by which incoming packets are matched against flow table entries.

Each flow table entry contains:

- Header fields to match against packets
- Counters to update for matching packet
- Actions to apply to matching packets

In Flow Table an action is associated with each flow entry, which decides how to process the flow. The FE can perform three different actions associated to the flow entries.

- Forward the flow's packets to a given port (or ports).
- Encapsulate and forward the flow's packets to a controller
- Drop the flow's packets.

3.1.2 A Secure Channel

The secure channel is OpenFlow channel, which is the interface that connects each FE to CC. Through this interface, the CC configures and manages the FE, receives events from the FE, and sends packets out to the FE. All channel messages between FE and CC must be formatted according to the OpenFlow protocol. The channel is usually encrypted using TLS, but may be run directly over TCP.

3.1.3 The OpenFlow protocol

It is the protocol which governs rules to communicate between the FE and the CC. It has the header and the data fields as per the type of message. It supports three message types, CC-to-FE, asynchronous, and symmetric, each with multiple sub-types. CC-to-

FE messages are initiated by the controller and used to directly manage or inspect the state of the FE. Asynchronous messages are initiated by the FE and used to update the controller of network events and changes to the FE state. Symmetric messages are initiated by either the FE or the controller and sent without solicitation.

3.1.3.1 Controller-to-Switch Messages:
- Features: Controller requests the identity and capabilities of a switch
- Configuration: Controller asks configuration parameters in the switch.
- Modify-State: Controller asks to modify flow entries in the OpenFlow table.
- Read-State: Reads the current configuration and collect statistics from the switch.
- Packet-out: Controller sends packets out a specific port.

3.1.3.2 Asynchronous Messages
- Packet-in: Switch generates this message and sends to the controller for processing.
- Flow-Removed: Switch notifies the controller about the removal of a flow entry.
- Port-status: Switch notifies the controller when a change to the port occurs.
- Error: Switch notifies the controller about a problem.

3.1.3.3 Symmetric
- Hello: Exchanged between controller and switch during the connection setup phase.
- Echo request/ reply: Used as keep-alive messages between the controller and switch.

B. Central Controller (CC)

The Controller is responsible for managing the Forwarding Elements. It takes high level routing decisions regarding data received from FE, it is also called as server. It makes control function independent of the hardware it controls. It speeds up the forwarding and routing process.

The FEs are connected to the CC before the actual communication takes place between the different hosts. CC decides whether the communication between hosts should be allowed or not. The controller adds and removes flow-entries from the Flow Table.

Policy rules are maintained by CC. CC stores control information such as addresses, location and policy. Such information is distributed to different FEs as needed. CC provides control and configuration function. It also provides security at lower layers, which is more promising than providing security at higher levels.

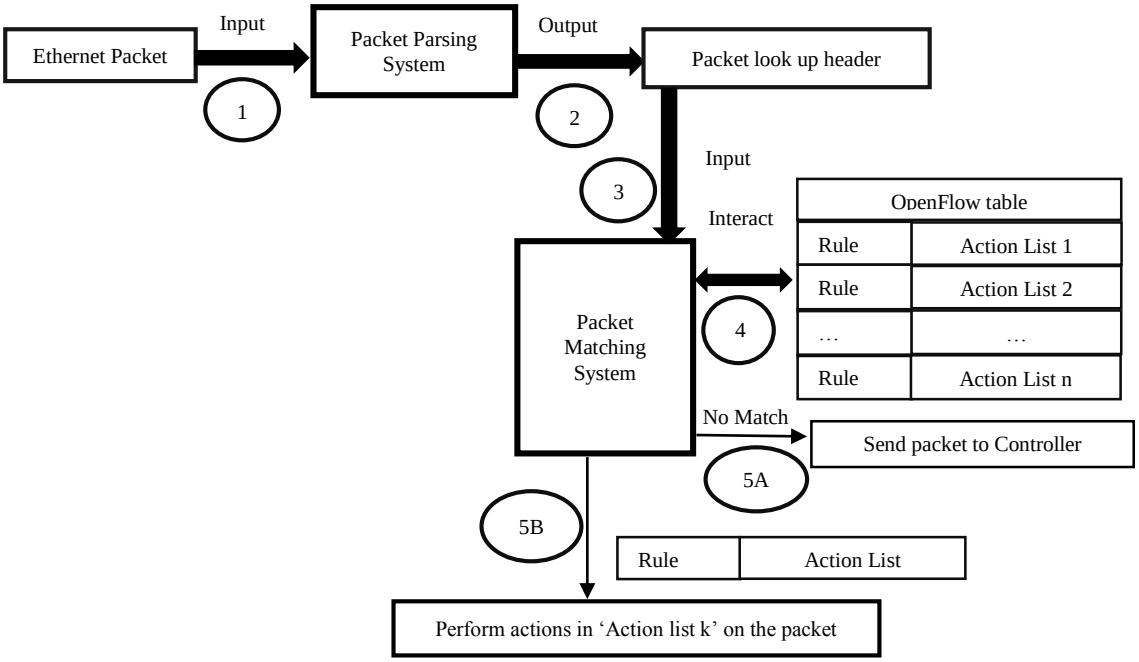


Fig 3: The processing and forwarding of packet

The processing of packet from entering to the switch, which may or may not be forwarded as per rule deployed in it, is shown in fig.2, with following steps.
1- The Ethernet packet entering the switch goes to a packet parsing system.
2- For matching purposes, header fields are extracted and placed in packet lookup header.
3- Generated packet lookup header is sent to the packet matching system.
4- The packet lookup header is compared to the rules defined in the OpenFlow flow table.
5A- The packet is sent to the controller for processing.
5B- The actions in the matched flow entry are performed on the packet.

3.2 Multiple switches based system

Here we consider multiple switches which support different number of hosts. As switches and hosts are increased, the system is organized in various patterns.
Definition Topology - A network topology is the arrangement of a network, including nodes and connecting lines.
Here two topologies are considered which are as follows

3.2.1 Linear topology

The Fig 4 shows the linear topology. It has one controller, n number of switches with n number of hosts, provided that every switch has single host connected to it. There are links between all switches, and they are connected to the controller.

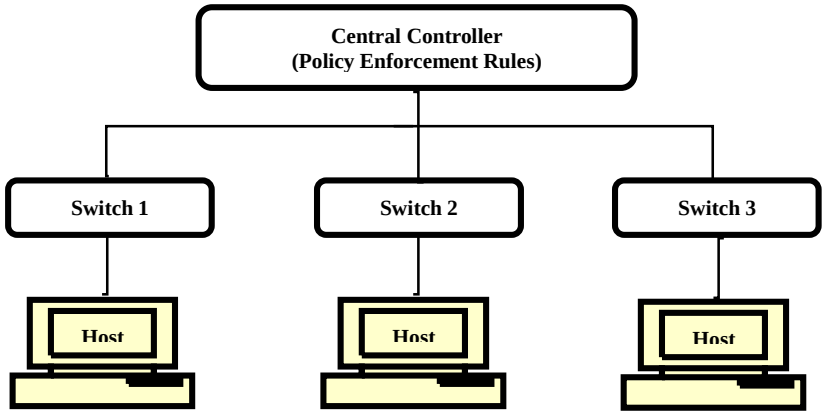


Fig 4: Linear topology, with 3 switches and 3 hosts

3.2.2 Tree Topology

The Fig 5 shows the tree topology. It has one controller, with multiple switches and multiple hosts. The tree topology has two more parameters to describe, the depth and fanout. This topology will have fanout +1 switches and fanout^depth hosts. Here every switch can have number of connected hosts specified by fanout. There are links between all switches, and they are connected to the controller

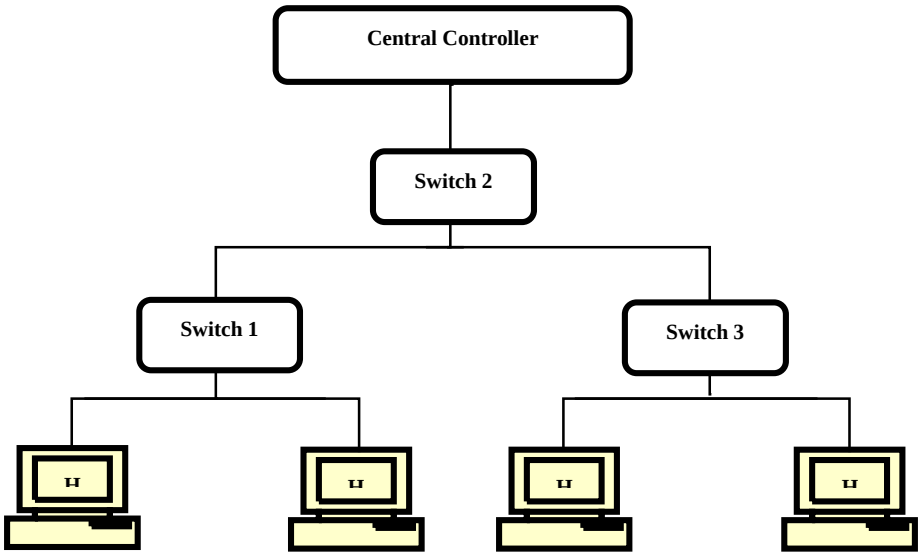
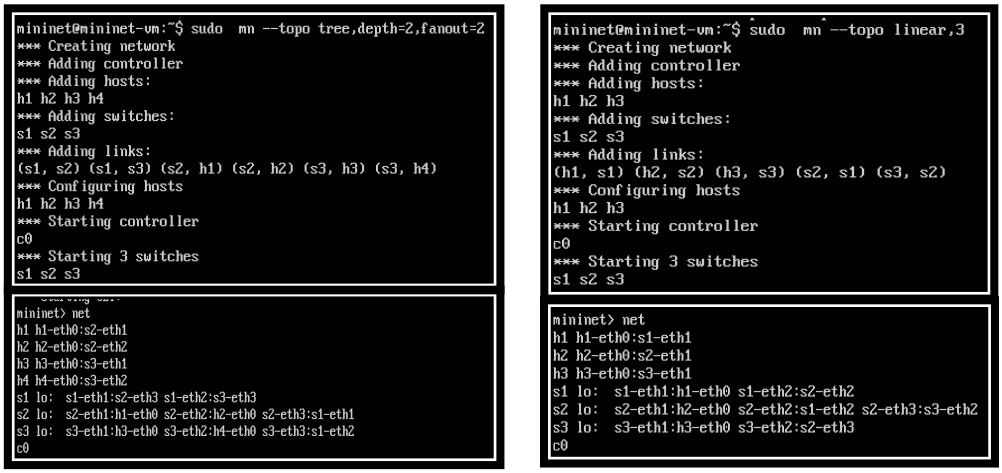


Fig 5: Tree topology, with depth=2 and fanout=2

4. RESULTS

After deploying the Linear and Tree topologies, we get following results. Fig.(6.a) shows Linear topology with 3 switches and 3 hosts as Fig.4 is deployed. There are links between all switches and controller. Fig.(6.b) shows Tree topology with depth 2 i.e. level 2 of switches and every switch with fanout 2has been shown as Fig 5 is deployed. The tree is created with mentioned depth and every level follows the number of

switches/ hosts as mentioned in fanout.



(6.a) (6.b)

Fig 6: (a) shows the deployment of linear topology, (b) shows the deployment of Tree topology using Mininet

For the linear topology a test is carried out, the bandwidth is calculated with various number of switches and it is recorded. Fig. 7 shows that as number of switches are increased, bandwidth descended for linear topology. For the tree topology, with various depths (i.e. levels) and fanouts (number of outgoing nodes), the network time, stop time, memory taken by network in Mininet, with varying number of hosts and switches is recorded. Fig. 8 shows details of startup time, memory taken for various number of hosts and switches using tree topology.

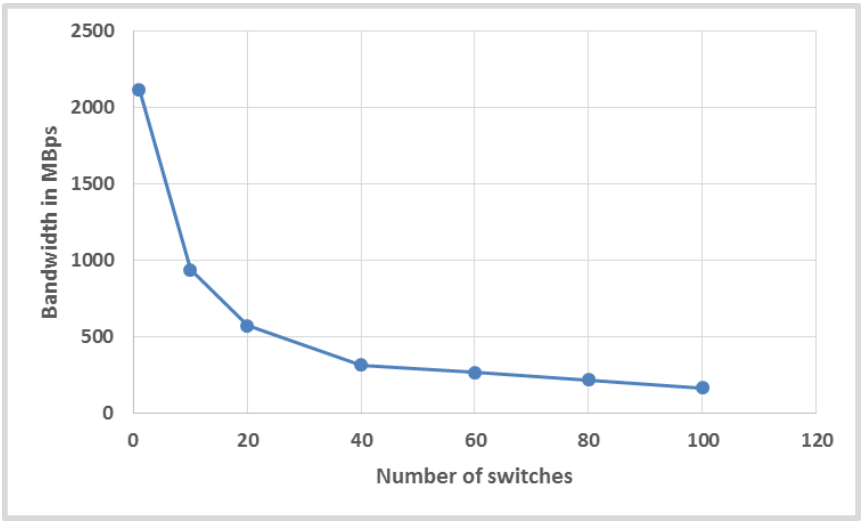


Fig 7: A chart shows trade-off between number of switches vs Bandwidth for linear topology using Mininet.

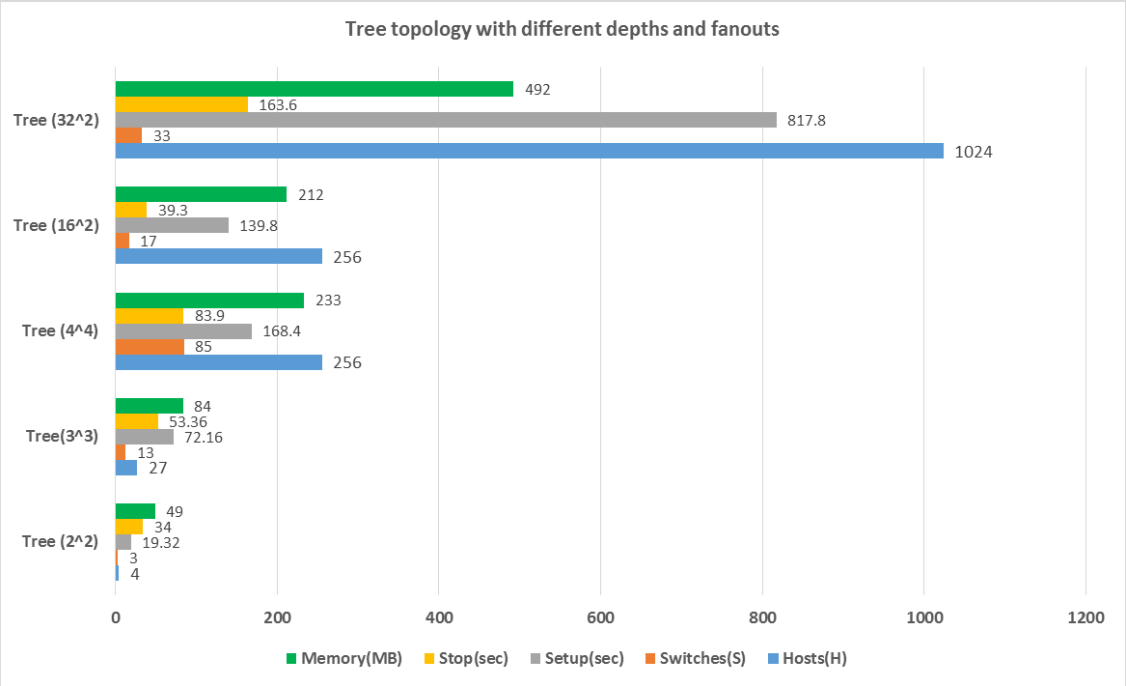


Fig 8: A chart represents various tests on tree topology with different number of hosts and switches, including setup and stop time, memory consumed by network in Mininet

5. CONCLUSION

The system decouples data forwarding plane from control plane. The policy rules are installed in controller, after which the process of packet forwarding through FEs done. Like existing on site networks, users can manage and specify the policy settings. Using enhanced FE switches makes, the architecture takes advantage of network virtualization and centralized control. Users can customize existing infrastructure using OpenFlow, so the new investment on network devices can be avoided. Without disturbing existing infrastructure, user can perform real life experiments in network using this system. This system makes innovation easier and makes deployed networks programmable not just configurable. The various topologies can extend the capabilities of OpenFlow technology.

REFERENCES

[1] D.D. Clark "The Design Philosophy of the DARPA Internet Protocols", Proc. ACM SIGCOMM '88, pp. 102-111.

[2] Global Environment for Network Innovations. http://www.geni.net, 2006

[3] T. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo,"The SoftRouter Architecture" ACM HOTNETS, 2004.

[4] de Oliveira Silva, de Souza Pereira, J. H, Rosa, P.F, Kofuji, S.T. "Enabling Future Internet Architecture Research and Experimentation by Using Software Defined Networking", IEEE 2012

[5] Greg Goth, "Software-Defined Networking Could Shake Up More than Packets", IEEE Internet Computing, 2011

[6] Mark Handley Orion Hodson Eddie Kohler. "XORP: An Open Platform for Network Research", ACM SIGCOMM Hot Topics in Networking, 2002

[7] Mr. Varun S. Moruse, Miss. A. A. Manjrekar," SOFTWARE DEFINED NETWORK BASED FIREWALL TECHNIQUE", IAEME, ISSN 0976 – 6367(Print),ISSN 0976 – 6375(Online)Volume 4, Issue 2, March – April (2013), pp. 598-606

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, "OpenFlow: Enabling innovation in campus networks", www.openflowswitch.org, 2008

[9] Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy. "Network Innovation using OpenFlow: A Survey", IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 16, NO. 1,2014

[10] Fei Hu, "Network Innovation through Openflow and SDN Principles and Design", CRC Press, 2014

[11] ONF White Paper, "Software-Defined Networking: The New Norm for Networks", April 2012

[12] Open Networking Foundation (ONF), "SDN Architecture Overview", Version 1.0, December 2013

[13] L. Yang, R. Dantu, T. Anderson, and R. Gopal, Forwarding and Control Element Separation (ForCES) Framework, Apr. 2004, RFC 3746.

[14] A. Doria et al., Forwarding and Control Element Separation (ForCES) Protocol Specification, Mar. 2010, RFC 5810.

[15] Yazici,V. Sunay, M.O, Ercan A.O, "Architecture for a distributed openflow controller", IEEE 2012

[16] Rob Sherwood, Glen Gibby, Kok-Kiong Yapy, Guido Appenzellery, Martin Casado, Nick McKeowny, Guru Parulkar "FlowVisor: A Network Virtualization Layer", OPENFLOW-TR-2009-1

[17] S. Turner, P. Crowley, J. DeHart, A. Freestone, B. Heller, F. Kuhns, S. Kumar, J. Lockwood, J. Lu, M. Wilson, C. Wiseman, and D. Zar."Supercharging planet lab:a high performance, multi-application, overlay network platform." J SIGCOMM '07: conference on Applications, technologies, architectures, and protocols for computer communications, pages 85–96, New York, NY, USA, 2007. ACM.

[18] K. Salah, J. M. Alcaraz-Calero, S. Zeadally, S. Almulla, M. Alzaabi, "Using Cloud Computing to Implement a Security Overlay Network", IEEE, 2011

[19] Bob Lantz, Brandon Heller, Nick McKeown," A Network in a Laptop: Rapid

Prototyping for Software-Defined Networks", Hotnets '10, October 20–21, 2010, Monterey, CA, USA.ACM 978-1-4503-0409-2/2010

[20] Wolfgang Braun, Michael Menth "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices", Future Internet 2014, 6, 302-336; doi:10.3390/fi6020302

[21] Heena, Japinder Singh "Development of Top Layer Web Based Filtering Firewall using Software Defined Networking" IJARCSSE, Volume 6, Issue 6, June 2016 ISSN: 2277 128X

[22] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie, "A Survey on Software-Defined Networking", IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 17, NO. 1, FIRST QUARTER 2015

[23] CS 154: Introduction to Mininet, Department of Information Systems and Computer Science Ateneo de Manila University, August 2014

[24] Cyrus Mewawalla, "Software Defined Network, Who's who and who's impacted", www.researchcm.com, September 2013