International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 18 (2015) pp 39030-39033

Construction of a Simplified Software Defined Networking (SDN) Test-Bed

Sanjeev Rao Palamand
Project Assistant Department of Electronics Systems Engineering
Indian Institute of Science, Bangalore Karnataka, India
sanjeev_palamand@rediffmail.com

Shakthipriya P
MTech Embedded Systems Technology Dept. of Electronics and Communication
SRM University, Kattankulathur Chennai, Tamil Nadu, India
shakthipriya@outlook.com

Abstract
Software Defined Networking (SDN) is an approach to enhance the computer network performance by employing a centralized controller that orchestrates the control of traffic in the network by decoupling the network plane and the control plane. The mechanism that governs this method of network operations is called OpenFlow Protocol. This paper proposes the construction of a simplified Test-Bed with a virtual SDN switch using OpenFlow Protocol. It further explains one of the preliminary applications tested on the Test-Bed, namely "Frame level separation" of data load.

Keywords: Centralized controller, Frame level separation, OpenFlow protocol, Software Defined Networking (SDN), Virtual SDN switch.

Introduction

The term Software defined networking (SDN) has been the outcome of the ever existing need for better network operability. The concept behind SDN has been evolving since a long time, driven by the desire to provide user controlled management of forwarding in network nodes. SDN, as described by the Open Networking Foundation is presented as follows:
"In the SDN architecture the control and data planes are decoupled, intelligence and state of the network are logically centralized and the underlying network infrastructure is abstracted from the applications." [1].
SDN establishes the separation of control and data plane and the node intelligence in the node is now available with a centralized controller and this controller can be programmed by external applications. Open interfaces exist between the controller and the network elements.

Existing System

In the existing system of traditional networking, the control and the data plane are combined in the network node. The data plane is responsible for the processing and delivering of packets based on the state of the routers and end points. The control plane is responsible for configuration of the node and programming the paths used for the data flows. Once these paths have been determined, they are sent down to the data plane. Forwarding data at the hardware level is based on this control information.
In this approach of traditional networking, once the flow management has been defined, the only way to make an adjustment to the policy is via reconfiguration of the devices.
With increasing use of mobile devices, the scaling of networks as per changing traffic demands is restricted by this approach [1].

Software Defined Networking System

In Software Defined Networks (SDN), the control is moved out of the individual network nodes and into the separate centralized controller. The controller can hence, exploit complete knowledge of the network to optimize flow management and support user requirements. SDN is emerging as an efficient way to support the dynamic nature of future network functions and intelligent applications while lowering operation costs through simplified hardware and software [1][2][3][4].

Related Work

Many network Test-Bed providers and communications Test-Beds for future internet analysis have introduced the support for SDN paradigms and programmable network controllers.
PlanetLab Europe is a key Test-Bed with OneLab that performs experiments on Futuristic Internet technologies. OneLab is extending PlanetLab Europe into new environments, beyond the classic wired internet. OneLab is evolving PlanetLab Europe deeply by incorporating new monitoring tools. OneLab is federating PlanetLab Europe, both with other PlanetLabs worldwide and with other types of Test-Beds.
The Software Defined Telecommunication Laboratory at Monash University, Australia is experimenting using Open Flow to simulate various network situations on physical network hardware [5].
OpenFlow is a network protocol that provides users with means to control the routing of data through a network switch. This is accomplished by providing access to the user through an API on the network switch which can be controlled by an OpenFlow controller. The Open Flow controller can then manage the traffic through each switch in the network based

on data and by adding and removing "flows" from the switch [6].

The OpenFlow Test-Bed developed by theMonash University consists of 32 Raspberry Pi's, 1 Server and 2 HP 3500-24 Network Switches. The Raspberry Pi's will be used to generate network traffic as well as collect network statistics while the Server will act as an OpenFlow Controller and control the Raspberry Pi's. The Server is a standard desktop PC. The HP 3500-24 is a 24 port Layer 3 capable networking switch. OpenDaylight, which is a Software Defined Networking (SDN) platform developed with major support from industry, is installed on the Test-Bed.

Construction of the SDN Test-Bed

SDN enables the rapid innovation and optimization of routing and switching equipment by decoupling the control and data planes in network switches and routers. SDN greatly simplifies network management by offering administrators network-wide visibility and direct control over the underlying switches from a centralized controller.

Routing Service is an intelligent application based on

Open Flow architecture. It takes an Open Flow-Based SDN approach to create a logically centralized control plane that is separated from the forwarding switches in order to focus on the required routing decision process and routing control from **a** large service provider's perspective.

Following are the components of the SDN Test-Bed architecture:

***Hardware*:**

- Two Linux machines
- Multiple USB to NIC card convertors
- Multiple Ethernet cables

***Software*:**

- Open vSwitch
- POX (python based controller)
- Ostinato (packet crafting software)
- Wireshark (packet tracing software)

Figure1 shows the linear topology according to which the network is connected.

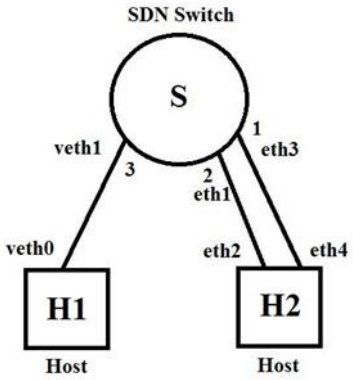

Fig.1. Linear topology arrangement of the network

The arrangement is chosen for its simplicity to demonstrate the working of SDN. The same can be extended to a bigger network and other topologies as well.

In this setup, 2 hosts are connected by an SDN switch. While host H1is connected to the switch by the link veth0-veth1, host H2 is connected by2 separate links namely eth1-eth2 and eth3-eth4, as seen in Fig.1. Having 2 separate links to a host eases demonstration of frame level separation of data packets using SDN which is the application developed on the Test-Bed.

The topology is realized by making the switch S, a virtual SDN switch. The two hosts H1 and H2 are laptops that have Linux (Ubuntu) as the operating system. The physical set up of the Test-Bed is as shown in Fig.2.

Setting up of the Test-Bed

Following are the steps for the setting up of the Test-Bed:

- The USB to NIC convertors are configured on each of the two Linux machines.
- The links of the NIC cards identified by the Linux machine are noted down to set up the port based forwarding.
- The buildbridge.sh file that is exclusively shell scripted for the Test-Bed is run to set up the virtual switch with a virtual link called the virtual eth link (veth).
- The controller (POX) is invoked in another new terminal.
- Host H1 is configured to run Ostinato on it. Ostinato crafts Ethernet packets of the user's wish and sends it on the links as desired by the user.
- The flows for the Open vSwitch are added dynamically and the behavior of the switch is changed dynamically which are then developed into applications according to the user's wish.
- Host H2 has the packet tracing software Wireshark running on it. This enables tracing of the packets that are sent by Host H1 through the Open vSwitch.

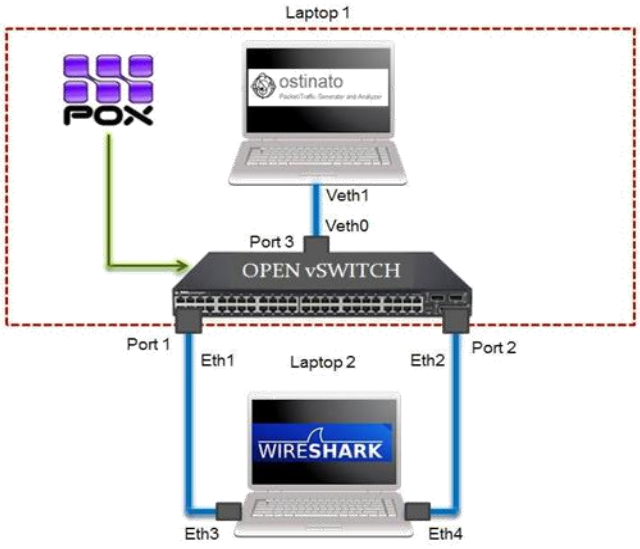

Fig.2. Physical set up of the SDN Test-Bed

Architecture of the SDN Test-Bed

A. Open vSwitch

Open vSwitch, is an open source implementation of a virtual switch that can be extended to work in multiple layers. In the hardware virtualization environments, the Open vSwitch provides the switching mechanism in a virtual way. It can be made compatible to multiple protocols and standards in networking. Open vSwitch can not only operate as a software based switch running with the virtual machines (VMs) or hypervisors but can also be ported to multiple virtualization platforms and switching chipsets. Open vSwitch can be created only in Linux environment.

B. POX Controller

POX is a Python-based open source development platform for Software Defined Networking (SDN) control applications, such as Open Flow SDN controllers. POX is used to create a modern SDN controller that offers a Pythonic OpenFlow interface, provides reusable components for path selection, node discovery, topology discovery, etc.,

C. Ostinato

Ostinato is an open-source, cross-platform network packet crafting software or traffic generator and analyzer with a friendly GUI that can craft and send packets of several streams with different protocols at different rate. Ostinato supports most of the common standard protocols such as TCP, UDP, ICMPv4, ICMPv6, IGMP and MLD. It allows a single client to control and configure multiple ports on multiple computers generating traffic. Exclusive control of a port to prevent the OS from sending stray packets is devised and it provides a controlled testing environment.

D. Wireshark

Wireshark is a freeware and open-source packet analyzer. It is used for network analysis, troubleshooting, software and communications protocol development. Wireshark, is released under the terms of the GNU General Public License. It is very similar to the command, tcpdump. It has a graphical front-end, plus some integrated sorting and filtering options. Wireshark allows the user to see all traffic visible on the network control interfaces. Port mirroring or various network taps extend capture to any point on the network. Wireshark can parse and display the fields, along with their meanings as specified by different networking protocols. Wireshark uses pcap to capture packets, so it can only capture packets on the types of networks that pcap supports.

E. Network Interface Controller (NIC) Cards

A Network Interface controller (NIC) also known as network adapter is a computer hardware component that connects a computer to a computer network. Early network interface controllers were commonly implemented on expansion cards that plugged into a computer bus. The low cost and ubiquity of the Ethernet standard means that most new computers have a network interface built into the motherboard.

The network controller implements the electronic circuitry required to communicate using a specific physical layer and also data link layer standard such as Ethernet, Wi-Fi or token ring. This provides a base for the full network protocol stack, allowing communication among small groups of computers on the same LAN and large-scale network communication through routable protocols, such as IP.

USB to NIC card convertors are used in the SDN Test-Bed since Laptops come with only 1 in-built NIC card.

Experiments Conducted on the SDN Test-Bed and Results

A. ***Application developed on the SDN Test-Bed: Frame level separation of data packets***

Flow separation in Layer 2 is a very unique aspect of Software Defined Networking. With the addition of dynamic flow rules to the switch, different packets can be separated from the stream and can be routed in a different link. It mainly helps in avoiding congested links and routing secure data in different links. For example in video streaming, re-routing only the multimedia flows through a different link can make videos jitter free. Another extension of this application of SDN can be in providing encapsulation of data to only specific links.

This flow separation is done in the frame level using the POX controller which is programmed to distinguish the packets that hit the controlled switch by using network prototype numbers as the parameter. The verification of the frame level flow separation is done by testing the different links using Wireshark.

B. ***Execution and Results of the experiment***

- After setting up the Test-Bed with a fully functional SDN switch, POX controller, Wireshark and Ostinato, all the links should be made active using up link command.
- Three packets namely TCP, UDP and IGMP are crafted in Ostinato in the Laptop 1 so as to hit the switch in an interleaved fashion.
- As a first step, the controller is added with the flow rule 'FLOOD' to check if all the crafted packets are received properly on both the links to Laptop 2. This verification is done by tracing the packets through Wireshark on Laptop 2.
- Both the links to Laptop 2 receive the stream of packets as expected to be received.
- Now the controller is added with Flow Rules to re-route the packets based on their network prototype numbers. We separate TCP packets from the interleaved stream of three packets and route it only to eth3. The remaining UDP and IGMP packets are made to flow through only eth4 (Refer Fig.2 for Test-Bed setup).
- This re-routing is successfully verified using Wireshark, on Laptop 2. The link eth3 receives only TCP packets while link eth4 receives the interleaved stream of UDP and IGMP.

We have employed 'Port-Based Forwarding' on the switch. This can be further extended to IP based forwarding by knowing the IP addresses of the systems that are connected in the network.

Conclusion

The work undertaken and explained in the paper involves the implementation of an SDN Test-Bed which is simple and well suited for experimental purposes, using Port-Based configuration which would work well for Local Area Networks (LAN).

The scope for improvement on our work is stated as follows:

- Implementation for Wide Area Networks (WAN) by interconnection of LANs can be ensured if Software Defined Networking (SDN) is implemented using IP based configuration.
- A mechanism to read the statistics of the controller to monitor the system can be included to detect possibilities of intrusion or other active attacks. The DdoS attack by flooding of irrelevant packets of data can be detected in the early stages using SDN.
- A programmable hardware SDN switch in place of a virtual switch can be implemented. For deployments that involve bigger networks, security and reliability, a hardware programmable switch that is manufactured by CISCO, IBM, JUNIPER and HP can be purchased and used.

References

[1] *Are we ready for SDN? Implementation challenges for software-defined networks* Sezer, S. ; Scott-Hayward, S. ; Chouhan, P.K. ; Fraser, B. ; Lake, D. ; Finnegan, J. ; Viljoen, N. ; Miller, M. ; Rao, N. Communications Magazine, IEEE Volume: 51, Issue: 7 Publication Year: 2013 , Page(s): 36- 43

[2] *Improving network management with software defined networking* Hyojoon Kim ; Feamster, N. communications Magazine, IEEE Volume:51, Issue: 2 Publication Year: 2013 , Page(s): 114- 119.

[3] *Software-Defined Networking: A survey* ,Computer Networks, Volume 81, 22 April 2015, Pages 79-95 Hamid Farhady, HyunYong Lee, Akihiro Nakao

[4] *A roadmap for traffic engineering in SDN-OpenFlow networks*, Original Research Article Computer Networks, Volume 71, 4 October 2014, Pages 1-30 Ian F. Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, Wu Chou

[5] www.monash.edu

[6] www.opennetworking.org/sdn-resources

[7] *Network Hypervisors: Enhancing SDN Infrastructure Original Research Article* Computer Communications, Volume 46, 15 June 2014, Pages 87-96 Shufeng Huang, James Griffioen, Kenneth L. Calvert

[8] *Software-Defined Networking: Challenges and research opportunities for Future Internet* Computer Networks, Volume 75, Part A, 24 December 2014, Pages 453-471 Akram Hakiri, Aniruddha Gokhale, Pascal Berthou, Douglas C. Schmidt, Thierry Gayraud

[9] *Software Defined Networking: Why we like it and how we are building on it*: White paper by CISCO.

[10] *How SDN will Shape Networking* - key note address by Nick McKeown.

[11] *Software Defined Networking using Open Flow* by Siamak Azodolmolky, PACKT publications.

[12] www.sdncentral.com

[13] www.openflowtutorial.com

[14] www.mininet.org

[15] www.criterionnetworks.com/academy