International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 4, Issue 1, January-February 2015 **ISSN 2278-6856**

Flow Installation in Open Flow Based Software Defined Network; A Security Perspective

Sandeep Singh[1], R.A. Khan[2] and Alka Agrawal[3]

[1] Research Scholar, D.I.T.
Babasaheb Bhimrao Ambedkar University, Lucknow

[2] Associate Professor, D.I.T.
Babasaheb Bhimrao Ambedkar University, Lucknow

[3] Assistant Professor, D.C.S.
Khwaja Moinuddin Chishti Urdu, Arabi~Farsi University, Lucknow

Abstract

In present world Software Defined Networks are setting new definitions of high speed and advance networking. These networks are growing very efficiently and spreading their popularity in industry as well as in research and development. There is a wide range of scope in this kind of advanced network from security point of view. In this research paper communication pattern and different flows coming to software defined networks are being analyzed. These flows are having their own property because of the protocols attached with them. Every time a new flow came into the software defined network, the controller generates a separate flow table for this flow. Here the process of manual installation of different flows is discussed for the security point of view of software defined network.

Keywords:- Open Flow, Software Defined Network, Flow installation, Security

1. INTRODUCTION

Whenever a network like software defined network is being setup in between the normal and traditional IP based network, we have to define its properties manually. Because of being an intelligent network, this network separates itself from others. The main property of software defined network is that both data plane and control plane are separated from each other for smooth and efficient processing of communication. When any external and traditional network came in touch with SD network, the controller attached with it must be aware of the flow properties it has to deal with. These flows having different kinds of data should have to import their flow table in the data base of controller. Otherwise it may be difficult for controller to handle with them. Here in this research work the process of manual installation of different flows is being discussed. When a network is developed under the circumstances of software defined network, the network emulator mininet can be used for designing purpose. The use of mininet can be an advantage because it provides a live test bed to implement in real life network without any change. While creating a simple network a command can be given as:

```
$ sudo mn
```

This simply starts mininet emulator with a single switch S1 and two hosts h1, h2 attached with it. We did not start any controlled since this will automatically choose its default controller of remote utility. As shown in figure 1.

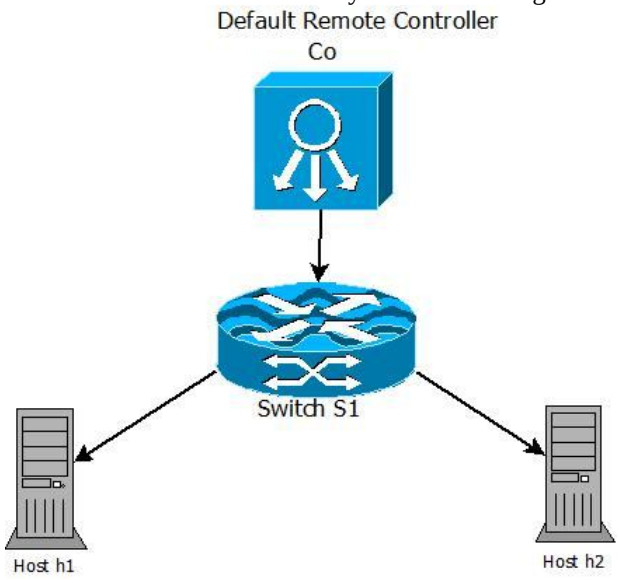


Figure 1. Default topology developed in Mininet.

2. BACKGROUND

Open Flow is the first standard designed for SDN [1]. The Open Flow concept was first proposed in the March 2008 in the paper "Open Flow: enabling innovation in campus networks" published by Nick Mc Keown, a professor at Stanford University [2]. Open Flow was initially designed for innovative development for a campus network. In a traditional campus network, introducing innovative ideas or new designs for the network is very hard to do, because researchers cannot modify the underlying network hardware [4]. For this reason, Open Flow separates the control plane from forwarding plane; hence it extracts the control logic from the network equipment [5]. This

innovation enables researchers to perform experiments via a programmable interface to the underlying network hardware. This makes it possible to verify new network protocols or topologies without modifying the underlying network equipment [6]. Each piece of network equipment maintains a flow table. All the traffic received by the equipments is forwarded based on this flow table. The flow table's configuration is fully controlled by an Open Flow controller. The flow table is not only concerned with IP, but also utilizes other information (for example for higher layer protocols or for non-IP network protocols) [7]. The Open Flow 1.0 specification defines twelve key fields including ingress port number, VLAN, Layer 2, Layer 3, and Layer 4 information and each field can be wildcard [3]. Network operators can create forwarding rules based on any field. For instance, operators only needing routing based on the destination IP address, can configure the controller to only validating utilize the destination IP field [8]. Open Flow is a communication interface and mechanism between the control plane and the data plane in an SDN environment [9]. Each switch contains a flow table and a secure channel interface. The flow table processes and forwards traffic flows using sets of flow matching rules, the flow entries [10]. The flow rules contain header fields to identify certain traffic packets and actions to perform on the corresponding packets. Each entry in the flow table also contains a list of counters which hold statistical information about the matching packets. The network control plane decides how to treat packets which have no matching flow rules in the flow table of a switch [11]. A controller installs and removes flow entries in a flow table through the secure channel of the switch using Open Flow-based messages. Open Flow is currently the only standardized protocol which enables Software Defined Networking and can be integrated on current physical and virtual networks [12].

3 FLOWS IN SDN

While starting mininet first we have to generate a sample network by given command.

$ sudo mn --topo single, 4 – mac --switch ovsk --controller remote

This command will generate a topology of four hosts connected with a single open virtual switch. This command assigns the mac address of all connected hosts according to their IP. A default remote controller is invoked through this command. The topological description of the command is shown in figure 2. While dealing with mininet we launched oracle Virtual Machine Box (VM Ware) and after successfully login into mininet system we assigns IP address to the link on which we will start our communication. For checking the available links the given command shall be run in to terminal.

$ sudo ifconfig -a

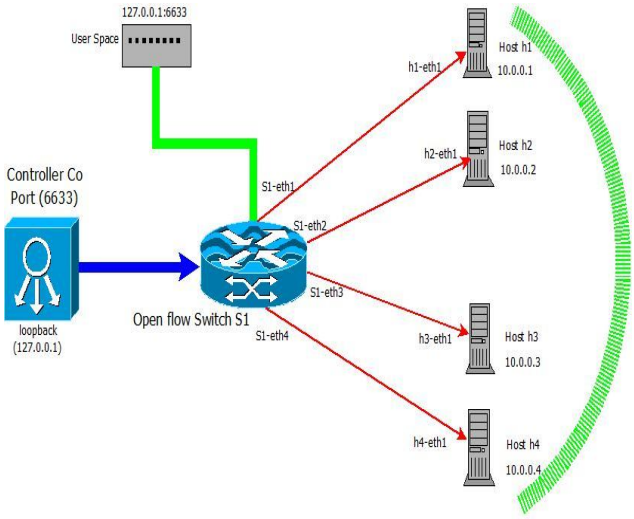


Figure 2. A sample topology of 4 hosts and single open Flow controller.

This command will result like figure 3.

```
mininet@mininet-vm:~$ sudo ifconfig -a
eth0      Link encap:Ethernet  HWaddr 08:00:27:6b:e3:3c
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1180 (1.1 KB)  TX bytes:684 (684.0 B)

eth1      Link encap:Ethernet  HWaddr 08:00:27:af:70:23
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:480 errors:0 dropped:0 overruns:0 frame:0
          TX packets:480 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:38208 (38.2 KB)  TX bytes:38208 (38.2 KB)

mininet@mininet-vm:~$ _
```

Figure 3. Testing of available connections

Since our developed topology is showing Ethernet connection 1. So we will assign this eth1 to a dynamic IP by given command.

$ sudo dhclient eth1

This command will assign an IP address to the Ethernet connection 1. Now we can revalidate it using given command.

$ sudo ifconfig eth1

This time it shows an IP address 192.168.56.101 which was assigned by previous command. As shown in figure 4.

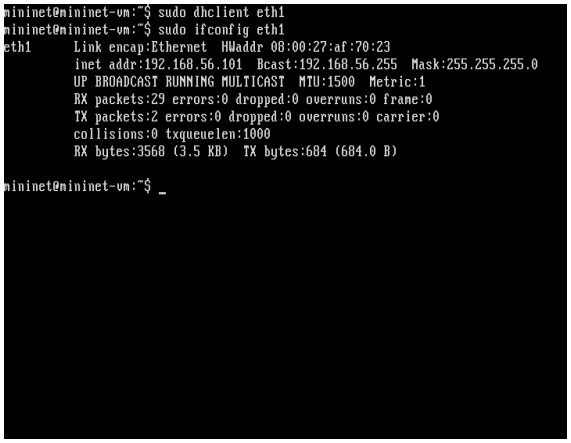


Figure 4. IP Assignment of Ethernet 1 link.

This command of creating a sample network topology just created four virtual hosts with separate IP address and generates a single open flow software switch in kernel with four ports. It has connected each virtual hosts to the switch with a virtual Ethernet cable. It assigns every host to a MAC address according to its IP address and finally configures the open flow switch to connect to a remote controller. We can start the putty terminal for checking the status of different hosts, so we first enable X.11 forwarding under the network assignment. Then putty server can be launched by XTerm using given command.

```
Xterm h1        Xterm h2
```

These commands will simply opens a separate terminal for different hosts. We can ping the packets using these XTerm terminals.

4 MANUAL INSTALLATION OF FLOWS

Since a simple topology is successfully created in mininet. Now its time to check the available flows inside the network. For testing this functionality we have to run the command.

```
mininet > h1 ping –c9 h2
```

This ping testing will be failed because switch tables are empty and none of the controller is connected to the switch. Therefore switch does not know what to do with the incoming traffic. For shorting out this problem we have to manually install the flows in our SSH terminal. This manual installation can be done using “dpctl” command. Basically “dpctl” is a utility that comes with the open flow reference distribution and enables visibility and control over a single switch flow table. It is specially useful for debugging, by viewing flow states and flow counters. Most open flow switches can start up with a passive listening port (in our current scenario it is 6633) from where we can port the switch without having to add debugging code to the controller. For this purpose we will create a second SSH terminal by using given command.

```
$ sudo dpctl show tcp:127.0.0.1:6633
```

This show command simply establishes a connection with the switch and dumps out its port state and capability. We can also check back the dump flows by given command in second SSH terminal.

```
$ sudo dpctl dump-flows tcp:127.0.0.1:6633
```

Since we have not connected any controller yet so the flow tables are empty. For manual installation of required flows we have to insert it by given command.

```
$ sudo dpctl add-flow tcp:127.0.0.1:6633 in_port=1, actions=output:2
$ sudo dpctl add-flow tcp:127.0.0.1:6633 in_port=2, actions=output:1
```

This can now be verified by pinging the particular flows.

```
$ sudo dpctl dump-flows tcp:127.0.0.1:6633
```

It gives the details about cookies, duration of activated flows and timeout etc. Since these flows are installed for a particular time limit and after reaching at its limit these will automatically removed from the database of remote controller. To test the installed flows we can run the given command to make sure.

```
mininet > h1 ping –c7 h2
```

This will result as shown in figure 5.

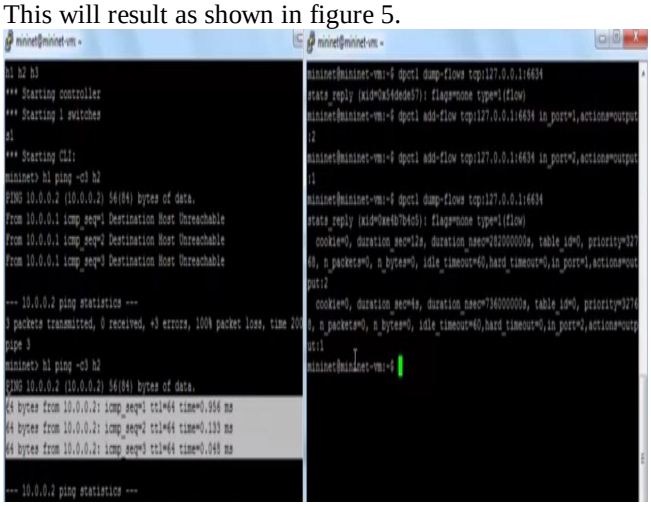
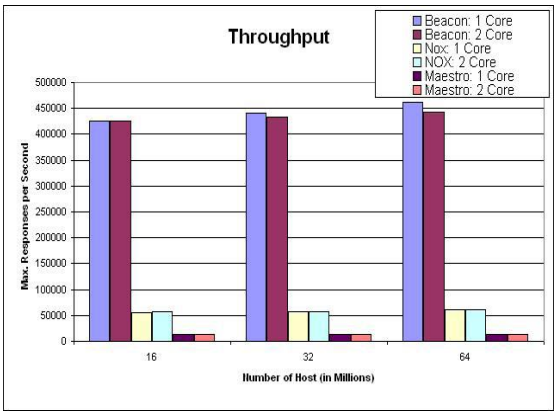


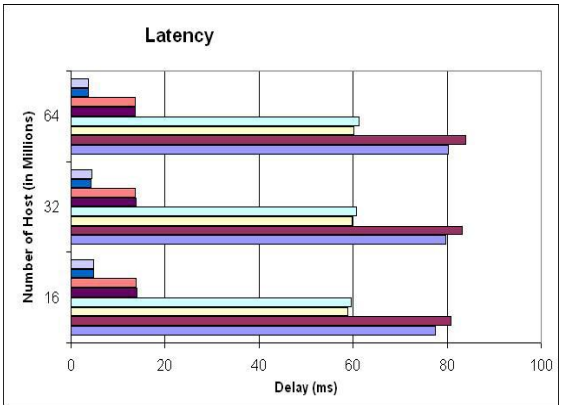
Figure 5. Response of installed flows in SSH terminal

This time ping requests were successfully generated and replied back.

5.Conclusion

Since we know that without having any flows in the database of a controller it is very difficult for smooth communication. Manual flow according to the requirement can be installed in the database of controller. This results the successful packet delivery over the software defined network. Every time a new flow came into the SDN network, controller has to decide about the treatment for flow. There are several controllers available in the industry where someone can select according to their need. If a separate controller is not used and we are dealing with the default controller of mininet emulator then these steps for manual installation of open flows can help in hassle free communication.





Throughput & Latency performance test results.

References

[1] "Software Defined Networking: The New Norm for Networks" ONF White Paper April 13, 2012.

[2] G. Goth, "Software-Defined Networking Could Shake Up More than Packets" Internet Computing,vol.15, no.4,July-Aug.2011, IEEE, pp. 6–9.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks" SIGCOMM Comput. Commun. Rev. 38, 2, March 2008, ACM, New York, NY, USA , pp. 42–47.

[4] H. Brandon, "OpenFlow Switch Specification, Version 1.0.0" Open Networking Foundation(ONF), December 2009, http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf, Last access date: Dec 10, 2014.

[5] J. Pettit, J. Gross, B. Pfaff, M. Casado and S. Crosby, " Virtual Switching in an Era of Advanced Edges" http://openvswitch.org/papers/dccaves2010.pdf, Last access date: Dec 09, 2014.

[6] OpenFlow Team. OpenFlow Tutorial, Required Software Installation Instructions http://archive.openflow.org/wk/index.php/OpenFlow Tutorial#Pre-requisites. Last visit on Dec 10, 2014.

[7] Open vSwitch. Open vSwitch, An Open Virtual Switch. http://openvswitch.org/. Last visit on Dec 12, 2014.

[8] L. Jose, M. Yu, and J. Rexford. Online measurement of large traffic aggregates on commodity switches. In Proc. Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), March 2011.

[9] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN Programming with Pyretic," USENIX ;login, vol. 38, no. 5, pp. 128–134, Oct. 2013.

[10] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs, and P. Skoldstrom, "Scalable fault management for OpenFlow," in IEEE International Conference on Communications (ICC), 2012, pp. 6606–6610.

[11] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King, "Debugging the data plane with Anteater," in Proceedings of the ACM SIGCOMM Conference. New York, NY, USA: ACM, 2011, pp. 290–301.

[12] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, "Netlord: A scalable multi-tenant network architecture for virtualized datacenters," in Proceedings of the ACM SIGCOMM 2011 Conference. ACM, 2011, pp. 62–73.

AUTHOR

Sandeep Singh has received his B. Tech. degree in Electronics and Communication Engineering from Chaudhary Charan Singh University, Meerut. He completed his M. Tech. in Wireless Communication & Networks from Gautam Buddha University, Greater Noida. He is a life time member of IACSIT, Singapore and member of CSI. His research interest includes wireless networks, Mobile IP; MPLS based networks and security of SDN.