Implementation of Simplified Custom Topology Framework in Mininet

Chandan Pal
Department of MCA
P.E.S. Institute of Technology
Bangalore, India
chandanpal06@gmail.com

Veena S, Ram P. Rustagi and K.N.B.Murthy
Research Center (Computer Science)
P.E.S. Institute of Technology
Bangalore, India
{sveena, rprustagi, principal}@pes.edu

***Abstract*—Openflow is an "open" protocol to enable communication between a Software defined Network controller and network elements. Openflow enables the controller to create flow table entries in the switches, and also enables the analysis for underlying network topologies. The Mininet provides a simple and inexpensive network test-bed for developing Openflow applications. The default version of the Mininet supports pre-defined topologies such as single, linear, tree etc., in a single IP network. To generate custom topology, one has to write programs in the Mininet. The goal of the proposed work is to provide a custom topology framework which enables the creation of any custom network topology of user's choice, including multiple IP networks. Thus, user can carry out research on different networks having separate broadcast domains. The framework has been so designed that it will not affect the performance of the existing Mininet significantly.**

Keywords - Controller, Mininet, Openflow, Software Defined networks, Virtualization.

I. INTRODUCTION

Nowadays computer networks are in constant evolution and require scalability and programmability to be ready for innovations of next generation networks. Existing operational networks does not facilitate testing of new innovative ideas and protocols. Thus, there is a need to have test-beds that facilitate new innovation and new protocol experimentation, and a virtual network infrastructure fills that need.

Virtualization is an act of creating a virtual version of something. It could be hardware virtualization, platform virtualization, server virtualization or network virtualization. Mininet is a virtual network test-bed that helps the researchers who would like to innovate and students who would like to understand the working of computer networks.

Mininet mainly gains importance in the study and implementation of Software Defined Network [9]. Software Defined Network (SDN) is one of the most promising and emerging network technology in recent years. It refers to a network infrastructure in which the control plane is decoupled from the plane that forwards the data packets. It enables separation of policy decision (control plane) from its implementation mechanism (data plane) where policy means the forwarding decisions made by the controller and implementation mechanism means the forwarding of the data packets by the switch. This separation enables implementation of control plane in an external software entity called controller [2], and implementation of data plane in the switch. The switch simply focuses on the forwarding task. This approach helps in quick testing and implementation of new ideas. It also increases the flexibility and agility, facilitates higher rate of innovation, and reduces complexity as well as the cost of switching elements.

Openflow [7] is a switching protocol that is based on the concept of software defined networks. Traditionally switches/routers tightly couple both the control plane and the data plane in the same device which leads to a networking topology which is non-scalable. SDN allows the network administrator to have easier control over the whole network from a central location. Openflow helps in moving the forwarding intelligence of the switch/router to a central controller. Each openflow switch maintains a flow table containing flow entries. It provides programmable interfaces for programming the data plane i.e., to add or remove flow entries from the flow tables [4]. Whenever a completely new packet is received by the switch, which it has not seen so far, it forwards this packet to the controller for the guidance on action to be taken. The central controller decides when, where and how to forward a data packet.

Most modern Ethernet switches and routers typically use *Ternary Content-Addressable Memory* (TCAMs) for flow-tables to provide packet forwarding at line-rate and implement functionalities such as firewalls, NAT, QoS etc., along with collection of statistics. While each vendor's flow-table may be different, there exists a common set of functions that run in many switches and openflow protocol is designed to make use of this common set of functions. Openflow provides an open protocol to program the flow-table in different network elements i.e. switches and routers. A network administrator can partition the traffic into

978-1-4799-4568-9/14 /$31.00 ©2014 IEEE

production and research traffic. Researchers can control their own flows by choosing the routes their packets follow, and the processing they receive. In this way, researchers can try new routing protocols, security models, addressing schemes, and even alternatives to IP in the existing operational network. On the same network, the production traffic is isolated and processed in the same way it is happening today. The data-path of an openflow switch makes use of the flow-table, and an action associated with each flow entry. The set of actions supported by an openflow switch is extensible.

To develop Openflow based applications, a small network of openflow switches can be built for testing and debugging purposes. Mininet emulator is developed to meet this demand. It is an excellent network emulator to develop Openflow based applications [5]. It provides an inexpensive test-bed for openflow application developers. Mininet creates a scalable virtual network on a simple desktop by using Linux processes in network namespaces. Mininet Emulator supports researchers, developers, debuggers and the people, who would like to understand the working of networks and test their ideas by having a complete experimental network on a laptop/desktop [1]. One can interact with the network using CLIs (API). Network designs that run in Mininet can easily be transferred to hardware switches supporting openflow for line rate packet forwarding. One can run Mininet on laptop, on a server, on a VM, on a native Linux box or in the cloud. Network topology is the arrangement of nodes (hosts, switches etc.) and links in a computer network. Existing Mininet supports some simple predefined topologies. These include a *single* topology (Fig.1), consisting of one switch and *'n'* number of hosts, *linear* topology (Fig.2), consisting of *'n'* number of switches, each connected with a single host, and *tree* topology(Fig.3), that connects switches and hosts in a hierarchical way. The tree topology defines depth of the tree and fan out at switch in this tree hierarchy.

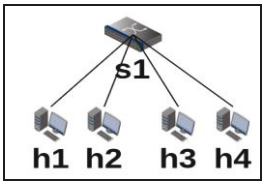

Fig.1. Single Topology

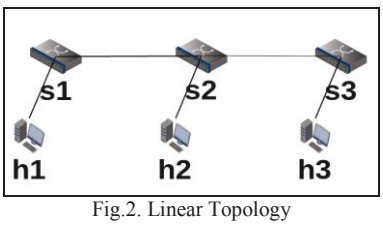

Fig.2. Linear Topology

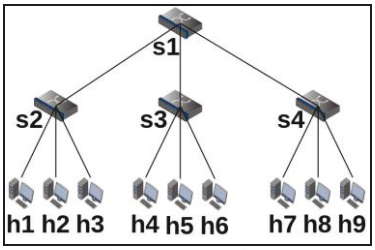

Fig.3. Tree Topology

If a user needs to create his/her own custom topology, Mininet provides programming support for the same. To create custom topologies, one has to write programs by modifying the existing Mininet code which may be a tedious job for a general user. In the proposed work, efforts are made towards creating a user friendly framework in Mininet which supports the creation of user defined topology as per his/her requirements, without writing any programming code. Users can specify the required topology either in a configuration file or interactively at run time. If the topology under consideration is very big, then it is suggested to specify it through the configuration file, for smaller topologies, either option can be used. This enhancement to the existing test-bed helps the researchers in creating the required topologies in an easy manner.

In a computer network, all the hosts connected to a single LAN belongs to the same broadcast domain, i.e. a host can directly communicate to another host in the same network without an intermediary i.e. router. In real life networks in general, it is required to group the hosts based on some criteria and separate the groups logically. Data packets communicating among hosts in one group will not be seen by hosts belonging to another group even though hosts of the other group may be connected under the same switch. The existing implementation of Mininet by default puts all the hosts in a single broadcast domain having the IP network address *'10.x.x.x/8'*. In this work efforts are made to overcome this limitation. Users can specify the details of different IP network addresses in a configuration file and create a number of IP networks as per his/her requirements. If IP network address is not specified in the configuration file, the default address *'10.x.x.x/8'* is taken for all the hosts.

II. RELATED WORK

Currently, network virtualization is the center of research interest because of its ability of dynamic programming, energy saving and low cost. In [6], authors have analyzed and compared the results of four projects based on virtualization concept. Mainly they focus on validating the openflow solution for an on demand virtual network architecture for which Mininet emulator is used. Nikhil Handigol and et al., [3] use an approach called Container-based Emulation where a virtual network environment of different networking elements was created to verify the published results. They tried to replicate the

results of 16 published networking papers using Mininet and found satisfactory results.

In general, simulators, test-beds and emulators are commonly used platforms in networking system research. Network simulators try to model the real world networks for the study purpose. The features of the models can be changed and the corresponding results can be analyzed. These platforms are relatively cheaper but cannot perfectly model all the details of the network [8]. Some example simulators are NS2, NS3, OPNET and OMNeT++.

OPNET is the most popular commercial network simulator used in industry for network research and development. It has a powerful GUI interface and a set of programming tools. The product comes with well supported documentation as it is a commercial product. NS2 is the most popular open source network simulator. It is an object oriented, discrete event simulator extensively used by networking research community. It uses C++ and OTcl for programming. It allows anybody to work on it, and contribute for its development. NS3 is also an open source, discrete event network simulator targeted for research and educational use. It is developed after gaining experience with NS2. It is designed to replicate the success mode of NS2 with some additional features. It is a new simulator and not backward compatible with NS2. One more open source, discrete event network simulator used in networking research is OMNeT++. It uses component based architecture and mainly used for the study of communication networks. All these network simulators can be used to study the behavior of the network by changing several parameters in the model. But the results obtained need to be carefully observed and analyzed using a series of offline test experiments. These simulators cannot be used in SDN research as they do not support openflow protocol.

Two important test-beds used in networking research are GENI (Global Environment for Network Innovation) and Planet-lab. Geni is a virtual lab created by Mobility first Project of NSF. It is used for exploring the future internet architectures. It creates major opportunities for the people to understand the working of networks worldwide and their interactions. Researchers can use this collaborative environment to test their innovative ideas. Planet-lab is an open platform for developing, deploying and accessing planetary scale services. It is a global research network that supports the development of new network services. It uses an implementation mechanism of container based virtualization in the Linux kernel to isolate slices. At present it consists of 1000+ nodes at 500+ sites which are globally dispersed. Many industry and academia research labs are part of this project.

Geni and Planet-labs are well designed test-beds which work on real network infrastructure. However, to use these test beds, a researcher needs to be a part of these working groups. Getting membership for these test-beds may involve a lengthy process and at times cumbersome for people with a small research environment. Another option available for researchers to test their ideas is to use emulators. Emulators are used to simulate an existing or planned network to assess the performance and predict the impact of changes. In an emulator the end systems attached to a network will behave as if they are attached to a real network. Real networks may introduce delays, errors during operation. Network may drop packets. Primary goal of a network emulator is to create an environment whereby users can connect their devices, applications, services and evaluate the performance and functionality in real world network-scenarios. Users can gather the results obtained by emulation and use these results for the required decision making.

NS3 can also be used as a limited functionality emulator. NS3 simulator gets enhanced into an emulator by integrating the test-bed and virtual machine environments. It provides two kinds of Net devices. *Emu* Net device allows NS3 simulations to send data to real network devices. The *tap* Net device allows a real host to participate in an NS3 simulation as if it were one of the simulated nodes. Thus the test-bed containing simulated nodes and real devices will give a better performance analysis for testing networking protocols and new ideas. Netem is another network emulator which provides network emulation functionality for testing protocols in Wide Area Networks. Here we can emulate the WAN by varying delay, loss, re-ordering of packets, sending duplicate packets etc. These emulators are designed for analyzing and understanding the working of existing networks and are based on the current network architecture. For the study of SDNs we require an emulator which supports openflow. So Mininet emulator has been considered for study in this work.

Mininet [5] is a network emulator using which we can create a network of virtual hosts, switches, controllers and links. Mininet hosts run on standard Linux software and the switches support the openflow protocol which is the de-facto protocol for SDN research. It is python based and the code developed on Mininet can be moved to real world networks with minimal changes. In existing Mininet, to create a custom topology other than the pre-defined ones, one needs to have python programming skills. The proposed work makes this functionality user friendly without the need for writing any programming code.

A. Creation of custom topology

In the proposed work, Users can create custom topologies of their choice simply by running some

commands. The required topology can be specified in a very simple manner via a configuration file in a human readable form and can be understood intuitively. All the links between different switches and the links from switches to hosts are specified as configuration data and this file is passed as parameter at command line when invoking Mininet (*sudo mn*). Thus, the framework facilitates easy creation of the topology of interest.

B. Creation of different Broadcast Domains

The framework allows creation of separate IP networks by specifying the preferred IP network addresses via the configuration file itself. Here, one also specifies the hosts belonging to different networks in a very simple manner. At the time of topology creation, the hosts will take the specified IP network address from the configuration file, instead of the single default IP address '10.x.x.x/8'. Thus multiple broadcast domains in the topology can be created very easily.

The above two results helps the researchers to develop, test and debug their innovative ideas and protocols. As Mininet is having openflow support, the topologies created using Mininet can be used directly for the study of Software Defined Networks.

III. EXPERIMRNTS, RESULTS AND DISCUSSIONS

In any organization, actual network is not confined to simple topologies like linear or tree. The real network topologies in these organizations are complex and represent more like hierarchical/graph connectivity. The switches and nodes are at different depth and even the fan out is different. Thus, for Mininet to be as close to real network as possible, some enhancements are required. In the proposed framework, any topology that is of interest to the user can be created resembling the real network. Care has been taken not to affect the performance of Mininet by doing this enhancement.

Mininet, by default, creates topologies with IP network address from the network number *'10.x.x.x/8'*. In the proposed framework one configuration file is added to the existing Mininet in addition to code enhancement. Users can create the topology of their choice by specifying the topology in a configuration file. Essentially, the configuration file contains the connectivity details between the switches and the hosts. An example of configuration file for the topology of Fig.4 is provided in Fig.5. Here switch-switch connectivity follows the syntax 'S_i-S_j-…-S_k', and switch-host connectivity follows the general syntax S_i-h_i[-N]. These two syntaxes are separated by double semicolon. The former specifies that switch S_i is connected to S_j through S_k, and the latter specifies switch S_i is connected to host h_i and this host belongs to network number N. If N is not specified, then the host belongs to the default network '10.x.x.x/8'.

In this topology (Fig 4) there are 4 switches, 7 hosts and one SDN controller. Switch s1 and s2 have 1 host each attached to them, s3 has 2 hosts attached to it and 3 hosts are connected to s4. Here one can observe that the fan out at each switch is different which can't be specified in default topology creation in the existing Mininet. To create such a topology in original Mininet, complex python code need to be written where each link need to be specified in programming constructs. In the proposed framework user can specify the links in a simple human readable text-file as in Fig.5. As the network number is not specified in the configuration file, the IP address of the hosts will belong to the default IP network address '10.x.x.x/8'.

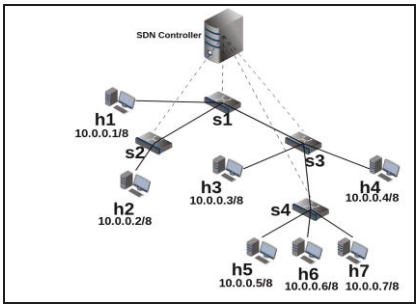

Fig.4. Simple Custom Topology

```
# Switch connectivity
s1-s2-s3
s3-s4;;
# Host connectivity
s1-h1
s2-h2
s3-h3
s3-h4
s4-h5
s4-h6
s4-h7
```

Fig.5. Configuration for Simple Custom Topology

If the user would like to have IP addresses belonging to specific network other than '*10.x.x.x/8'*, then that can be achieved by adding the required network address ranges in the configuration file as in Fig.6. The resultant topology looks as in Fig.7. Here all hosts in the resultant topology belong to the user specified network number *'192.168.10.x/24'*.

```
# Network numbers
192.168.10.0/24;
# Switch connectivity
s1-s2-s3
s3-s4;;
# Host connectivity
s1-h1-1
s1-h2-1
s2-h3-1
s2-h4-1
s2-h5-1
s3-h6-1
s3-h7-1
s4-h8-1
```

Figure 6. Configuration for Custom Topology with Single IP Network Address

Newly created topology can be verified using the *'net'* command of Mininet. We can also verify the IP addresses of each host using the regular Linux network commands e.g. *'ifconfig'*.

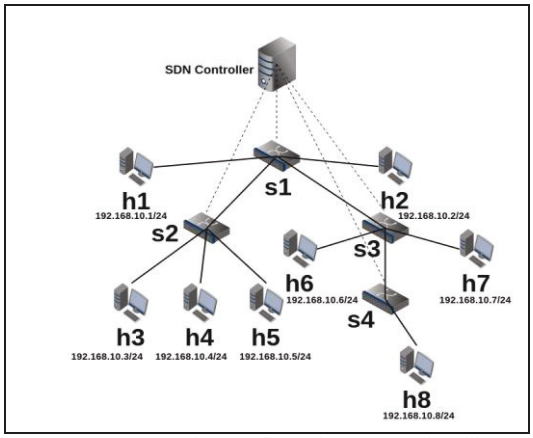

Fig.7. Custom Topology with Single IP Network Address

When it is necessary for the user to create multiple IP networks in the topology under consideration, same is supported by the proposed framework. Different network addresses can be mentioned in the configuration file for this purpose. Example Configuration file looks as in Fig. 8, and the corresponding topology diagram is as in Fig.9. In this topology there are four networks with different number of hosts.

```
# Network numbers
172.15.0.0/16
172.20.0.0/16
192.168.10.0/24
192.168.20.0/24;
# Switch connectivity
s1-s2-s3
s3-s4
s4-s5;;
# Host connectivity
s1-h1-1
s1-h2-2
s2-h3-3
s2-h4-4
s2-h5-3
s3-h6-4
s3-h7-3
s4-h8-4
s5-h9-1
s5-h10-2
s5-h11-2
s5-h12-3
```

Fig.8. Configuration for Custom Topology with Multiple IP Network Addresses

Specifying a host belonging to which network is done in the configuration file. In this topology h1 and h9 belongs to N1; h2, h10 and h11 belongs to N2; h3, h5, h7 and h12 belongs to N3; h4,h6 and h8 belongs to N4. If the network is not specified in the configuration file then the IP address for the host is taken from the default IP network address. Thus user can create the topology of his/her choice easily by specifying one configuration file.

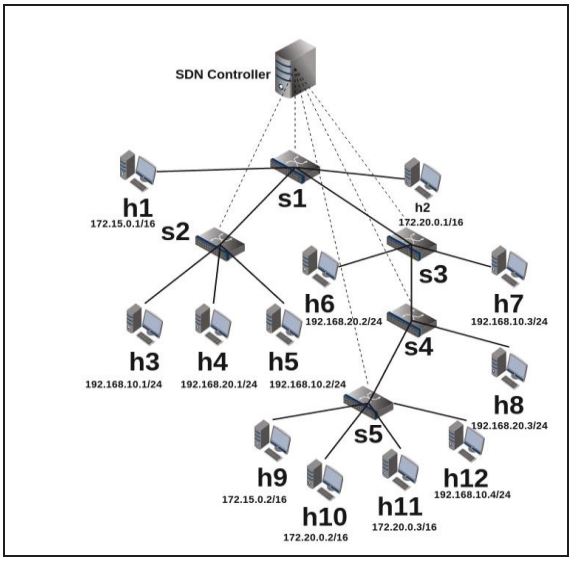

Fig.9. Custom Topology with Multiple IP Network Addresses

The host belonging to one network is reachable from any other host within the same group. This can be verified using the *'ping'* command. If the host belongs to different group then the packet need to be forwarded through the controller. In future, this is planned to be handled by enhancing the controller functionality.

Care should be taken not to give loops in the network while creating the configuration file. The framework allows for creation of loops and does not make a check. If topology contains the loops, then unless the controller implements spanning tree algorithm, any packet sent in the network may cause network flooding and broadcast storm and thus prevent one host from communicating with the other hosts.

IV. PERFORMANCE STUDY

After adding the proposed framework into the original Mininet the time taken for the creation of topology in original Mininet as well as in the enhanced Mininet was compared. The experiments were repeated on two different test setups; a) running the Mininet installed on the native machine, and b) running Mininet under VirtualBox. Studies were conducted by varying the number of switches created and the number of hosts created in the topology. The results obtained are given in the Table I.

The experiments were repeated by creating different network domains in the same topology. Again the number of switches created and the number of hosts created is varied and time taken to create the topology is analyzed. The results obtained are given in Table II.

The graphical depictions for the results of Table I and Table II are in Fig.10 and Fig.11 respectively.

Table I. Creation Time for Custom Topologies

Sl. No.	No. of Switches(No. of Hosts)	Avg. Time taken in Seconds	
		Existing Mininet	*Enhanced Mininet*
1	s1(5)	0.45	0.41
2	s1(5)s2(10)	1.09	1.19
3	s1(5)s2(10)s3(15)	2.39	2.53
4	s1(5)s2(10)s3(15)s4(20)	3.9	4.18
5	s1(5)s2(10)s3(15)s4(20)s5(25)	6.11	6.72

Table II. Creation Time for Multiple Broadcast Domains

Sl. No.	No. of Switches(No. of Hosts)	Avg. Time taken in Seconds	
		Existing Mininet	*Enhanced Mininet*
1	s1(5)	0.48	0.338
2	s1(5)s2(10)	1.10	1.19
3	s1(5)s2(10)s3(15)	2.43	2.55
4	s1(5)s2(10)s3(15)s4(20)	3.9	4.19
5	s1(5)s2(10)s3(15)s4(20)s5(25)	6.14	6.70

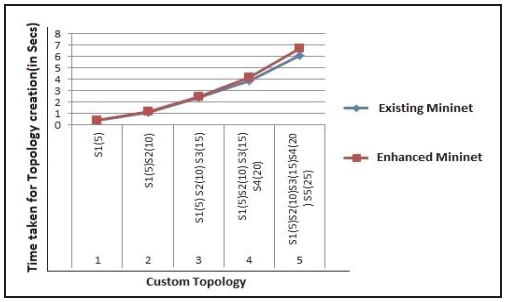

Fig.10 . Creation of Custom Topology

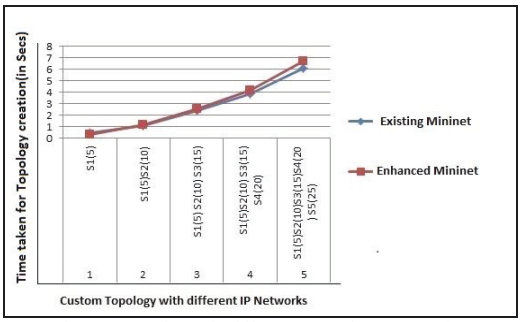

Fig.11. Creation of Different Broadcast Domains

These results show that the performance (time taken) of the original Mininet and the Enhanced Mininet for the creation of different topologies is comparable. As the number of switches and the number of hosts increase there is a slight increase (less than 10%) in the time taken by the Enhanced Mininet, but still this increase is negligible when compared to the ease at which the custom topology can be created. This framework is very much useful for the researchers to create the topologies of their interest to test their ideas and protocols.

V. CONCLUSION AND FUTURE WORK

Mininet emulator is a handy tool for networking researchers to emulate real operational network and to test their innovative ideas and new protocols. As it is openflow enabled, it is very much suitable for SDN research. Existing Mininet is enhanced to create custom topologies in a user friendly manner. The custom topology can be specified either through the command line or through the configuration file. The experimental results show that the time performance of topology creation practically remains unaffected by this additional feature. Users can also create different broadcast domain for their experiments.

In future, one can work on implementing spanning tree algorithm in SDN controller, to handle loops in the custom topology and to forward the packets between different networks. Similarly, the configuration file created in the new framework needs to be enhanced to specify the link characteristics such as bandwidth, delay, packet loss etc.

REFERENCES

[1] Bob Lantz, Brandon Heller and Nick McKeown "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks", Hotnets, October 2010.

[2] P. Fonseca, R. Bennesby, E. Mota and A, Passito, "A replication component for resilient openflow-based networking", NOMS, 2012.

[3] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz and Nick McKeown, " Reproducible Network Experiments Using Container-Based emulation", CoNEXT, December, 2012.

[4] Nick Mckeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Peterson, Scott Shenker, Jonathan Turner, 'OpenFlow: Enabling Innovation in Campus Networks', *ACM SIGCOMM Computer Communication Review, Volume 38, Number 2, April 2008*.

[5] Mininet.
http://www.openflow.org/wk/index.php/OpenFlow_Tutorial
http://mininet.org/overview/

[6] Msahli M., Pujolle G., Serhrouchni A., Fadlallah A., Guenane F., "Openflow and on demand Networks", Third international conference on "Network of the Future" Nov, 2012.

[7] OpenFlow Protocol.
http://www.openflow.org/wp/learnmore/

[8] Jianli Pan, Raj Jain, "A survey of Network Simulation tools – Current status and Future Development", project report, Nov, 2008.

[9] Software Defined Networks.
https://www.opennetworking.org/sdn-resources/sdn-definition