International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

A Framework for Implementing Realistic Custom Network Topology in Mininet

Veena S[1], Chandan Pal[2], Ram P. Rustagi[3], K. N. B. Murthy[4]

[1,2,3,4] Department of Master of Computer Applications, PES Institute of Technology,
100 ft. Ring Road, BSK III Stage, Bangalore, India

Abstract: ***Software Defined Networking (SDN) is an emerging technology which enables the separation of control plane and the data plane in the networking infrastructure. Communication between the SDN controller and the network elements can be achieved using the Openflow switching protocol. Using Openflow the controller can create/modify the entries in the flow tables of switches which in turn can define the working of underlying networks. Mininet provides a simple and useful network test-bed for the development of Openflow applications and experimental network infrastructure. The default version of Mininet allows creation of few pre-defined topologies such as single, linear, tree etc. To create a custom topology, lot of programming effort is required in Mininet software. The goal of the proposed work is to develop a new framework for the creation of custom topology very easily without any programming. The new framework also enables the users to create separate IP subnets, the networks with different link parameters such as Bandwidth, Latency, Packet loss etc. Thus researchers can test their ideas and protocols using this new test bed. People working on Data Center Technology research can also make use of this framework. Experimental implementation shows that the modification made to the default version of Mininet does not affect the performance noticeably.***

Keywords: Controller, Data Center Technology, Link Parameters, Mininet, Openflow, Software Defined Networks, Virtualization.

1. Introduction

Nowadays computer networks are in constant evolution. The next generation networks must be scalable, programmable, flexible and be adaptable to innovative ideas. Researchers continue to work on providing innovative solutions or modifications to the current network infrastructure. But such experiments cannot be practically conducted on existing operational networks as these may interrupt the operational traffic on networks. Thus, there is a need to have customized test-beds for the experimentation and testing of new innovations and new protocols, which can be solved by the virtual network infrastructure.

Virtualization is an act of creating a virtual version of something. It could be hardware virtualization, platform virtualization, server virtualization or network virtualization. Researcher's new ideas and protocols cannot be tested on real networks as it may affect the working of the operational network. Virtual network infrastructure would be the answer in such situations. We can create such an experimental test-bed using Mininet Emulator. In the proposed work efforts are made to enhance such a virtual network test-bed, to help the researchers for their experimentation and students to understand the working of computer networks.

To address virtualization, agility, mobility and manageability of today's networks Software Defined Networks (SDN) [2],[10],[19] is gaining importance. SDN enables programmability and configurability of the underlying networks by decoupling the control plane and the data plane of the network [1],[17]. The control plane requires an SDN controller which makes policy decisions and dictates the switches about the action to be taken by updating the flow tables of the switches. The switching elements do the forwarding job as per the entries in the flow table (instructions by the controller). This forwarding mechanism is called the data plane. The controller can be implemented as internal or an external software entity [3], and the data plane is implemented in the switch for the forwarding of data packets. This approach helps in quick testing and implementation of new ideas increases the flexibility and agility, facilitates higher rate of innovation, and reduces complexity as well as the cost of switching elements.

Communication between an SDN controller and the switches is carried out by a switching protocol, called Openflow[14]. In Traditional switches/routers both the control plane and the data plane are embedded in the same device. This leads to a situation where in introducing modifications to the existing algorithms/implementation is practically impossible as these are proprietary protocols and the networking topology becomes non-scalable. SDN allows the network administrator to have easier control over the whole network from a central location. The forwarding intelligence of the switch/router can be moved to a central Openflow controller. Each openflow switch maintains a flow table containing flow entries which is programmable. Programming the data plane enables us to add or remove, modify the flow entries from the flow tables [7]. Whenever a completely new packet is received by the switch which it has not seen so far, the switch forwards this packet to the central controller for the guidance on action to be taken. The decision about when, where and how to forward a data packet is done by the central controller.

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. But this implementation is vendor dependent. Even though the way in which the data stored in the flow-table is different, there exists some common set of functions that run in many switches. The Openflow protocol is designed to capture these common set of functionalities. It provides an open protocol to program

the flow-table in different network elements i.e. switches and routers. A network administrator can partition traffic into production and research networks. By writing new rules the researchers can control their own flows without disturbing the operational traffic. In this way, researchers can try out new routing protocols, other different ideas on security models, new addressing schemes in the existing operational networks. The production traffic is isolated and processed in the same way it is happening today without any disturbance. The data-path of an openflow switch consists of a flow-table, and an action associated with each flow entry [7][9]. The set of actions supported by an openflow switch is extensible.

To develop an Openflow based applications, a small virtual network of openflow switches is very much handy for testing and debugging purposes. Mininet emulator [8] is developed to meet this demand. One can create virtual openflow network in desktop/laptop through Mininet emulator. Using Mininet, openflow applications could be developed in an easy and inexpensive manner. Mininet uses Linux processes in network namespaces on a simple desktop for the creation of scalable, virtual, Software Defined Networks. Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop/Desktop [6]. One can interact with the network using CLIs (API). Network designs created using Mininet could be customized and easily ported to real hardware switches which supports openflow. Mininet could be run on different platforms and different hardware devices with various capabilities like laptops, servers, Virtual Machines, Linux boxes or even in the cloud. The default version of Mininet supports some simple predefined network topologies (a network topology is the arrangement of nodes e.g. hosts, switches etc. and links in a computer network). These pre-defined topologies are i) a single topology (Figure 1), consisting of one switch and 'n' number of hosts, ii) linear topology (Figure 2), consisting of 'n' number of switches, each connected with a single host, and iii) tree topology (Figure 3), that connects switches and hosts in a hierarchical way. One can specify the depth of the tree and fan out at switch in the tree hierarchy for the tree topology. In the tree topology, all switches have same fan out value.

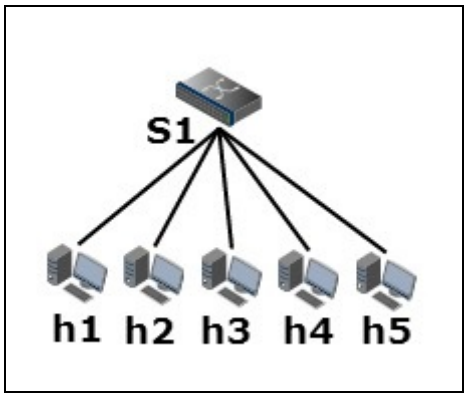

Figure 1: Single Topology

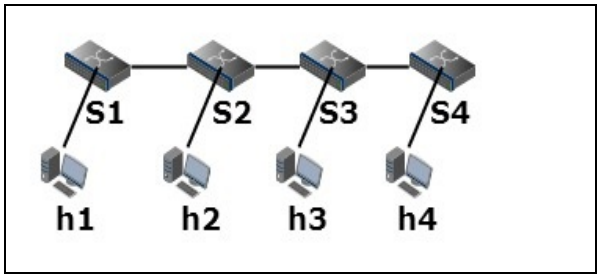

Figure 2: Linear Topology

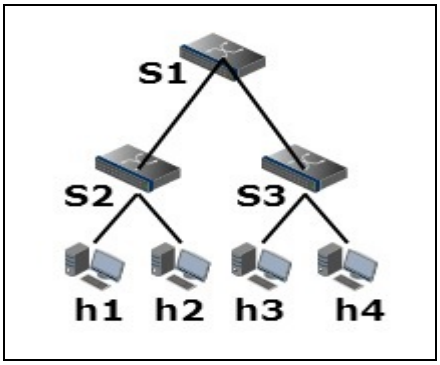

Figure 3: Tree Topology

These pre-defined topologies helps one to understand the basic operation of Mininet with openflow, but does not mimic real life networks which follow a custom designed topology specific to that network requirements. If a user needs to create his/her own custom topology, Mininet provides programming support for the same. However, 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 in a simple ascii text which can also include different link parameters for each of the link in the 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 could be used. This enhancement to the existing Mininet test-bed helps the researchers in creating the required topologies which may be helpful in their work, in a very easy manner. The new framework enables very easy creation of fat-tree topology which is extensively used in Data Center networks. In a fat-tree the tree link becomes fatter i.e., the link bandwidth gets increases, as it goes higher the level in the tree.

As we know, in a computer network, all the hosts connected to a single LAN(Local Area Network) belongs to the same broadcast domain by having the same network number. Here one host can directly communicate to another host in the same network without the help of an intermediary device router. In real life networks in general, it may be required to group the hosts in the network based on some criteria into separate logical groups. 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 default

implementation of Mininet puts all the hosts into a single LAN having a single broadcast domain with the IP network address '10.x.x.x/8'. In the proposed work efforts are made to overcome this limitation by creating the networks with different broadcast domains. Users can specify the details of different IP network addresses and also the link details like bandwidth, packet loss, delay etc. 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.

2. Related Work

Currently, network virtualization is the center of research interest because of its ability of dynamic programming, energy saving and low cost. In [9], 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.[5] use an approach called container based emulation where an environment of virtual hosts, switches and links were created which combines the best features of software simulators and hardware test-beds. They tried to replicate the results of 16 published networking papers using Mininet and found satisfactory results.

In general, research on networking systems normally uses platforms like simulators, test-beds and emulators. Network simulators try to model the real world networks for the study purpose. One can analyze the outcomes of experiments by changing the features of the models. These are relatively cheaper but can not perfectly model all the details of the network [16]. Some example simulators are NS2, NS3, OPNET and OMNeT++.

OPNET [15] is the most popular commercial network simulator used in industry for networking research and development. Its GUI interface is very powerful and has several programming constructs. Users can create networks easily using this as it is supported by good documentation. It is a commercial product, so maintenance is also very good. NS2 [11] is the most popular open source network simulator. It is an object oriented, discrete event simulator targeted at networking research. It uses C++ and OTcl for programming. It allows anybody to work on it, and contribute for its development. NS3 [12] is also an open source, discrete event network simulator targeted for research and educational use. It is developed after gaining experience with NS2. It includes all the successful and attractive features of NS2 with some additions. It is considered as a new simulator and not backward compatible with NS2. OMNeT++ [13] is also an open source, discrete event network simulator having component based architecture. This is mainly used for experimentation with 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 SDN architecture and openflow protocol.

Two important test-beds used in networking research are GENI (Global Environment for Network Innovation) and Planet-lab. GENI [4] is a virtual lab created by Mobility first Project of NSF. It is used for exploring future internets. One can create scalable, heterogeneous, interactive global networks, do innovations, study and understand their interactions with society. It provides collaborative and exploratory environment for researchers to test their ideas and protocols on wide area networks. Planet-lab [18] is an open platform for developing, deploying and accessing planetary scale services. It is also 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. Currently 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 using the emulators. Emulators are used to simulate an existing or planned network. By using emulators one can assess the performance of networks, analyze 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. Here there would not be any significant difference between the real devices and virtual devices. Real networks may introduce delays in communication, transmission errors during operation. Network may drop the packets. Primary goal of a network emulator is to create an environment which models the real networks, whereby users can connect their devices, applications, services and evaluate the performance and functionality in real world network scenarios. Users gather the results obtained by emulation and use these results directly for decision making purpose.

NS3 can also be used as a limited functionality emulator [12]. 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. Both type of devices can communicate to one another as if they belong to same category. Netem [10] is another network emulator which provides network emulation functionality for testing protocols in Wide Area Networks. People working on WAN technology can make use of this emulator. Here we can emulate the WAN by varying delay, packet loss, changing the order of packets, sending duplicate packets etc. to replicate the real time situations. 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 and notion of control plane and data plane. So Mininet emulator has been considered for study in this work as it supports Openflow. This enhanced framework is mainly helpful for the research community pursuing their work in the area of Software Defined networks.

Mininet [8] is a network emulator which creates 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 very attractive and handy for SDN research. Openflow is the most prominent SDN component supported by several vendors. In [1] the Openflow controller is used to design a mobile cloud management system. Such an Openflow, SDN network can be created by Mininet very easily. 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, which could be a humongous task especially for creating large custom network topology. The proposed work makes the functionality of creating custom topology user friendly without the need for writing any programming code All the required details of the network connectivity and characteristics are specified in a simple configuration file (described in next section). The prosed framework enhances Mininet in the following ways.

2.1 Creation of custom topology

In the proposed work, Users can create custom topologies of their choice by running some commands. The required topology can be specified in a very simple manner in 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 and also optionally link parameters can be 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 any custom topology of user's interest.

2.2 Creation of different Broadcast Domains

The framework allows creation of separate IP networks by specifying the preferred IP network prefixes 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 default IP address '10.x.x.x/8'. This approach provides creation of multiple broadcast domains in a simple way.

2.3 Creation of custom Topology with different Link parameters

The framework further allows a user to specify the different link parameters such as link bandwidth, delay, queue size and packet loss. These values can be specified for each of the links in the configuration file of the custom topology. At the time of topology creation the link details are taken from the configuration file and network is created corresponding to these specified values. Using this feature, realistic topologies as used in Data Centers like fat-trees can be created in a very simple manner.

The above three enhancements 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. As the topology of user's interest can be created so easily, it can be adapted for the study of Data Center Technologies.

3. Experiments, Results and Discussions

In the proposed framework, any topology of interest to the user can be created resembling the real network. The framework is so implemented so as not to affect the performance of Mininet by making these enhancements.

Mininet, by default, creates topologies with IP network address from the network '10.x.x.x/8'. In the proposed framework one configuration file is added to the existing Mininet. Essentially, the configuration file contains the connectivity details between the switches and the hosts. An example of configuration file for the topology of Figure 5 is provided in Figure 4. Here switch-switch connectivity follows the syntax 'S_i-S_j-…-S'_k, and switch-host connectivity follow the general syntax S_i-h_i[-N]. Optionally for each of the links the user can specify the parameters like bandwidth, delay, loss etc. These two syntaxes are separated by double semicolon. The former specifies that switch S_i is connected to S_j through S_k. 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'. Comments corresponds to the line starts with '#' character.

```
#Switch Connectivity
s1-s2-s3
s3-s4;;
#Host Connectivity
s1-h1
s1-h2
s2-h3
s2-h4
s2-h5
s3-h6
s3-h7
s4-h8
```

Figure 4: Configuration for Simple Custom Topology

In this topology there are 4 switches, 8 hosts and one SDN controller. Switch s_1 and s_3 has 2 hosts each attached to them, s_2 has 3 hosts attached to it and 1 host is connected to s_4. 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 Figure 4.

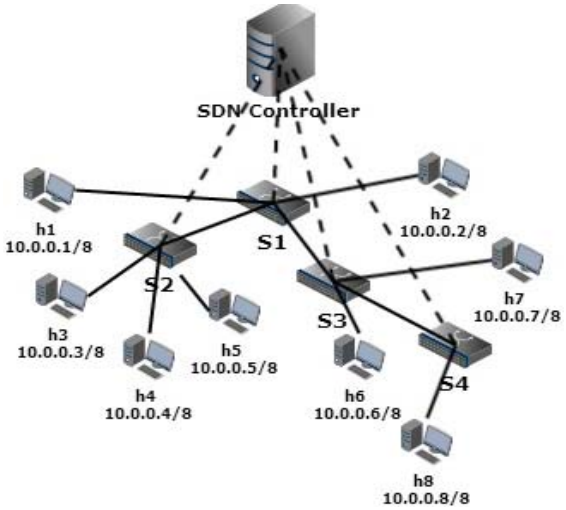

Figure 5: Simple Custom Topology

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'.If the user would like to have IP addresses belonging to specific network, then that can be achieved by adding the required network address in the configuration file as in Figure 6. The resultant topology looks as in Figure 7.

```
#Network Numbers
192.168.10.0/24;
#Switch Connectivity
s1-s2
s2-s3-s4;;
#Host Connectivity
s1-h1-1
s2-h2-1
s2-h3-1
s3-h4-1
s3-h5-1
s4-h6-1
```

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

Here all hosts in the resultant topology belong to the user specified network number '192.168.10.x/24'.

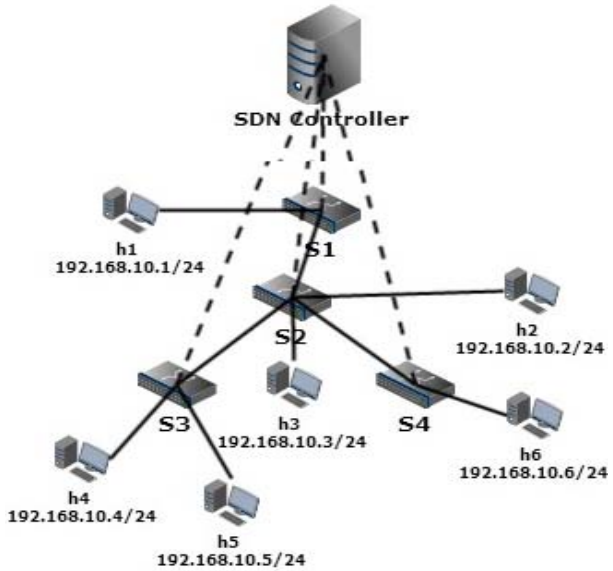

Figure 7: Custom Topology with Single IP Network Address

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*'.

To create multiple IP networks in a topology under consideration, different network addresses can be mentioned in the configuration file. An example of such a configuration file is given in Figure 8, and the corresponding topology diagram is in Figure 9. In this topology there are four networks with different number of hosts. Here h_1,h_7 and h_8 belongs to N_1, h_2, h_5 and h_{10} belongs to N_2, h_3 and h_9 belongs to N_3, h_4 and h_6 belongs to N_4. 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.

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

Figure 8: Configuration for Custom Topology with Multiple IP Network Addresses

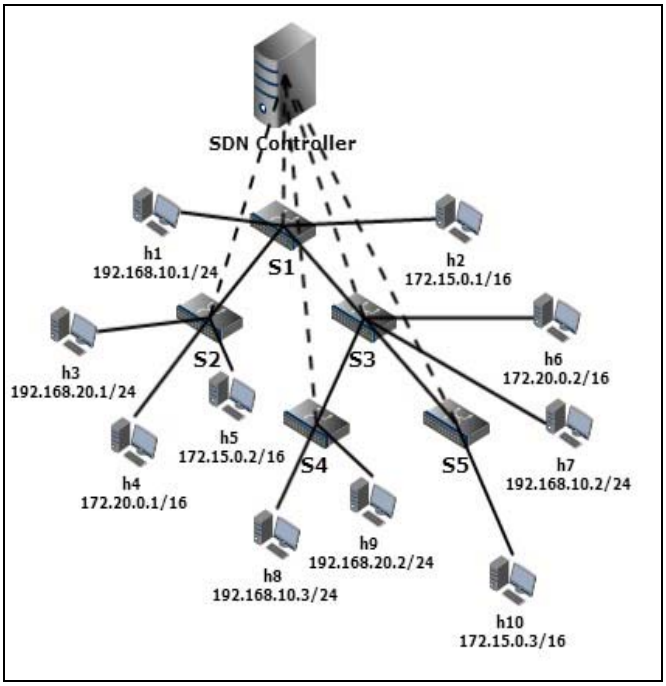

Figure 9: Custom Topology with Multiple IP Network Addresses

Normally in real networks different links can have different parameters. An example of such a configuration file for the topology diagram in Figure 11 is as shown in Figure 10. Thus user can create the topology of his/her choice easily by specifying one configuration file. If the link parameters are not specified in the configuration file then the default values will be taken.

```
#Network Numbers
192.168.20.0/8
172.15.0.0/16;
#Switch, host Connectivity with link parameters
s1-h1(BW=20)-s2(BW=10, Delay='10ms')-s3(BW=20, Delay = '5ms'-h2(BW=10)
s2-h3(BW=10)-s4(BW=20, Delay='5ms', Loss=10)
s3-h4(BW=10)-h5(BW=5, Delay = '5ms')-h6(BW = 5, Delay= '10ms')
s4-h7(BW = 10, Delay = '10ms') -h8(BW = 10, Delay = '10ms');;
#IP to host Details
h1-2
h2-1
h3-2
h4-2
h5-1
h6-2
h7-1
```

Figure 10: Configuration for Custom Topology with Different Link Parameters

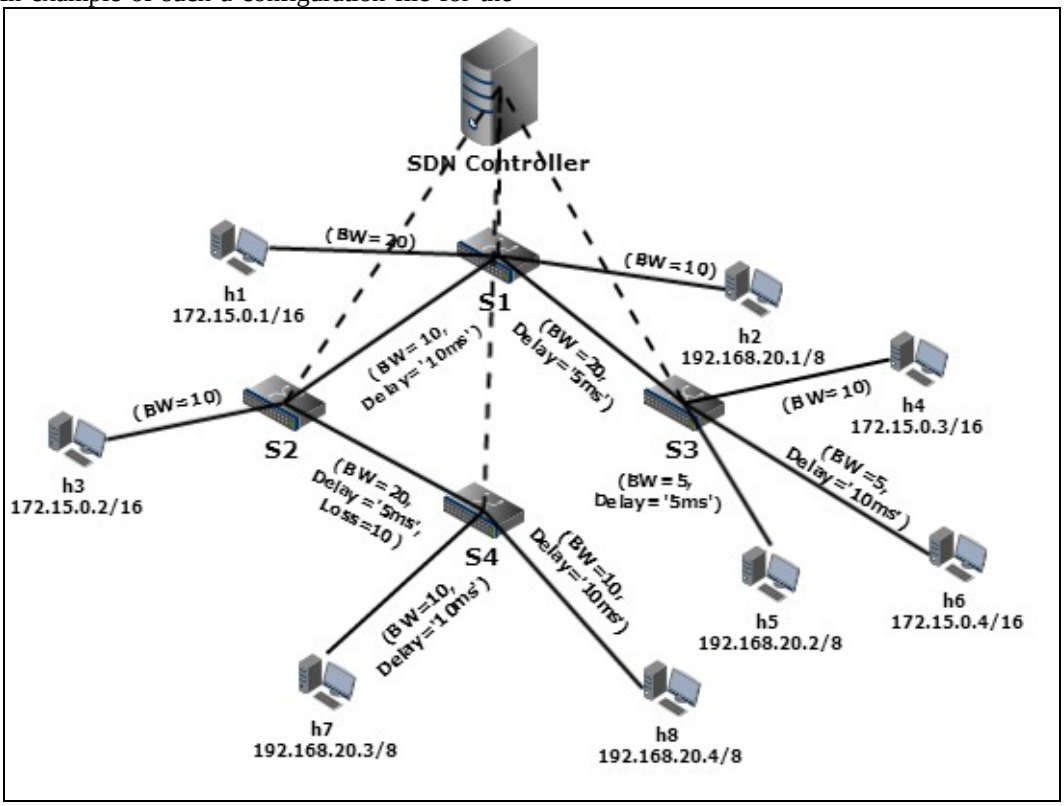

Figure 11: Custom Topology with Different Link Parameters

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 network element working as L3 switch designated by controller. In future, this is planned to be handled by enhancing the controller functionality.

In the current framework, care should be taken not to create 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.

4. 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.

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	Sl(5)	0.45	0.41
2	Sl(5)S2(10)	1.09	1.19
3	Sl(5)S2(10)S3(15)	2.39	2.53
4	Sl(5)S2(10)S3(15) S4(20)	3.9	4.18
5	Sl(5)S2(10)S3(15)S4(20) S5(25)	6.11	6.72

The graphical depictions for the results of Table I is shown in Figure.12.

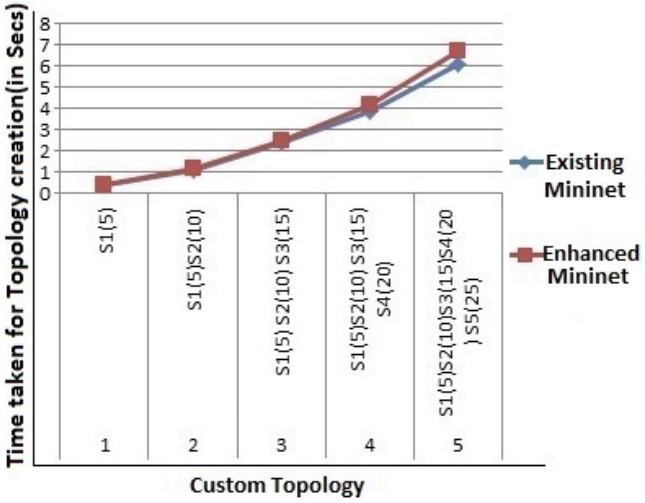

Figure 12: Creation of Custom Topology

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

Table 2: Creation Time for Multiple Broadcast Domains

Sl. No	*No. of Switches (No. of Hosts)*	*Avg Time taken in seconds*	
		Existing Mininet	*Enhanced Mininet*
1	Sl(5)	0.48	0.338
2	Sl(5)S2(10)	1.1	1.19
3	Sl(5)S2(10)S3(15)	2.43	2.55
4	Sl(5)S2(10)S3(15)S4(20)	3.9	4.19
5	Sl(5)S2(10)S3(15)S4(20)S5(25)	6.14	6.7

The graphical depiction for the results of Table II is shown in Figure 13.

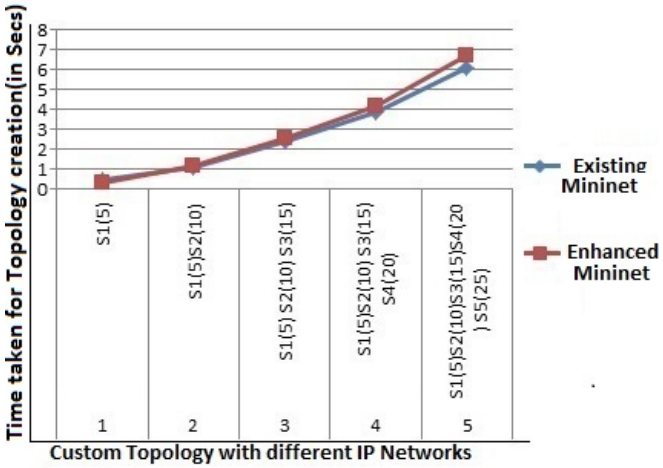

Figure 13: Creation of Different Broadcast Domains

The results show that the performance (time taken) by the original Mininet and the Enhanced Mininet supporting 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.

5. 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 topologies which model the real time network topologies by specifying different depth and fan-out, creating different broadcast domains, specifying different link parameters etc. in a user friendly manner. The topology can be specified either through the command line or through the configuration file. The experimental results show that the time taken for the creation of topology of user's choice practically remains unaffected by this additional feature. Users working on data center research can also create different topologies like fat-trees 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, other data center technologies such as TRILL can be implemented in controller and Mininet can be upgraded accordingly.

References

[1] Roberto Bifulco, Roberto Canonico, Marcus Brunner, Peer Hasselmeyer, Faisal Mir, " A Practical Experience in Designing an Openflow Controller", European Workshop on Software Defined Networking, 2012.

[2] Kostas Choumas, Nikos Makris, Max Ott, "Exploiting Openflow Resources towards a Content-Centric LAN", Second European Workshop on Software Defined Networks, 2013.
[3] P.Fonseca, R. Bennesby, E. Mota and A, Passito, "A replication component for resilient openflow-based networking", NOMS, 2012.
[4] GENI – Global Environment for Network Innovation Available: http://www.geni.net.
[5] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz and Nick McKeown, " Reproducible Network Experiments Using Container-Based emulation", CoNEXT, December, 2012.
[6] Bob Lantz, Brandon Heller and Nick McKeown "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks", Hotnets, October 2010.
[7] 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*.
[8] Mininet Available: http://www.openflow.org/wk/index.php/OpenFlow_Tutorial http://mininet.org/overview/
[9] Msahil M, Pujolle G, Serhrenchni A, Fadlallah A, Guenane F, "Openflow and on demand Networks", Third international conference on "Network of the Future" Nov, 2012.
[10] Netem – Network Emulator Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem
[11] NS2 Available : http://www.isi.edu/nsnam/ns/ NS3 Available : http://www.nsnam.org/
[12] OMNeT++ Available:http://www.omnetpp.org/ Openflow Protocol http://www.openflow.org/wp/learnmore/
[13] OPNET Modeller http://www.opnet.com/
[14] Jilani Pan, Raj Jain, "A survey of Network Simulation tools – Current status and Future Development", project report, Nov, 2008.
[15] Peter Peresini, Maciej Kuzniar and Dejan Kostic, "Openflow needs you! A call for a Discussion about a Cleaner Openflow API", Second European Workshop on Software Defined Networks, 2013.
[16] Planet Lab Available: https://www.planet-lab.org/node/263
[17] Software Defined Networks https://www.opennetworking.org/sdn-resources/sdn-definition

Author Profile

Mrs. Veena S. holds M.C.A. degree from MCE, Hassan, University of Mysore and M. Tech. Degree from AAIDU Allahabad. She is currently pursuing her Ph.D. under VTU in the area of Software Defined Networks. Mrs. Veena has 18 years of teaching experience in various institutions and taught post-graduate, under-graduate and Diploma students in different areas. She has completed her Instructor Training for CCNA Exploration (now Routing and Switching) Modules 1 through 4 from Cisco Networking Academy and training students for the CCNA Routing and Switching course. She has also been an invited speaker in Network Hands-On workshops as well as imparting hands-on training on IPv6 networking. Currently she is working as Associate Professor at PESIT, Bangalore, India.

Dr. Ram P Rustagi, a senior IEEE member, is working as Professor, Info Sc & Engg Dept, PESIT and teaching Undergraduates, Post Graduates as well as guiding few Research scholars. Prior to joining PESIT, he was Vice President at Kirusa Software Pvt Ltd, India/USA. As VP, he developed applications in Cellular VAS areas, He also managed Deployment and Operations Support for Kirusa's customers in India and APAC region. Besides academic activity, he is working on few research projects as per grants received from Pluribus Networks, USA, Kirusa Software, Bangalore, India. . He is also serving as a member of board of directors with M/s Altima Technologies Inc, USA. *Academic background*: Ph.D from IIT Delhi, India, 1998 and M.Tech from Indian Institute of Science, Bangalore, India, 1981.

Dr. K. N. Balasubramanya Murthy is currently the Vice-Chancellor of PES University, Bangalore. Dr. Murthy has served PES Institute of Technology (PESIT), Bangalore as the Principal & Director from 2005 to 2014 and has over 30 years of experience in Teaching, Training, research, industry and Administration. He holds a Bachelors in Electrical power Engineering from University of Mysore (1980), Masters in Electrical Engineering from Indian Institute of Science, Bangalore (1986) and Ph.D. in Computer Science and Engineering from Indian Institute of Technology - Madras, Chennai (1996). He was invited to New Mexico State University as a research specialist in Parallel Computing during the year 1998-99. He served Malnad College of Engineering, Hassan as a faculty in Electrical Engineering and East West Institute of Technology, Bangalore as its Founder Principal. Dr. Murthy has supervised 4 research scholars for Ph.D. degree and authored over 90 research papers in reputed journals and conferences. He served as a member of the Academic Senate and Executive Council as well as several academic committees of Visvesvaraya Technological University (VTU) - Belgaum of Karnataka. He was a Board member of BITES under the chairmanship of Prof.R.Natarajan for a period of three years during 2008-11. He is currently serving as a member of a committee constituted by Karnataka Knowledge Council (KKC) for recommending mechanisms to enhance employability of Engineering graduates in the state of Karnataka under chairmanship of Prof.G.Padmanabhan of IISc.