EE 555 Major Project, Spring 2020
Deadline – May 1, 2020 11:45 PM
(Hard deadline for both On-campus and DEN students)

Overview

The objective of this project is to build an SDN controller that has the capabilities of a router and a switch. SDN stands for Software Defined Networking. Traditional / legacy networking devices mainly have two components / modules in them. One is the forwarding plane / data plane and the other is the control plane. The control plane is responsible for interacting with peer devices and evaluate network topology, develop routing tables and ARP caches so that packets can be forwarded to either destination hosts or next hop routers depending on the destination IP Address / MAC Address. For example, a switch will have a MAC to port info mapping table. In this table, the switch will keep a mapping of the MAC address of the host and the output port to use, so that when the switch receives a packet with a specific MAC address, it can forward the packet in the corresponding port. Similarly, routers have a routing table that has the destination network addresses, their subnet masks and the details of the next hop router and output port. Whenever a router receives a packet, it performs longest prefix matching and then decides the output port on to which a packet must be forwarded. In legacy networking devices, both control plane and forwarding plane reside in the same hardware.

But, in case of SDN, both control plane and data plane are separated. SDN has two main types of devices in its architecture. The first is a controller that is responsible for control plane and the second is a packet forwarding switch that is responsible for data plane. Instead of every router computing and deriving the routing tables, SDN uses a centralized controller that does all the computation part. The SDN controller has complete visibility of the total network and is aware of which subnets are connected to which router. Once the controller gathers all the information about the network topology, it computes the routing tables. When a packet forwarding switch receives a packet, it sends the packet to the controller. The controller examines various headers of the packet and then makes a routing / forwarding decision. The controller then conveys the routing / forwarding decision to the packet forwarding switch, and this enables the packet forwarding switch to forward the packet to the appropriate output port after making any required changes. The packet forwarding switch and the controller interact with each other using a protocol called OpenFlow. OpenFlow is not the only protocol that is designed for SDN. There are a bunch of other protocols that can be used to realize SDN. But, in this project, we will be concerned only with OpenFlow and your project is going to be implemented using OpenFlow.

If the packet forwarding switch sends every packet to the controller, then the latency in the network will be increased. Increase in latency is not desirable as far as a network performance is concerned. So, to avoid this, the controller will "*push*" OpenFlow rules to the packet forwarding switch. Let us look at a small example that can tell you how these rules can be helpful. If a host, say 'A' is attached to a switch on port 1. All the packets with a destination MAC of host A, needs to be forwarded to port 1. Once the controller realizes that host A is on port 1 of the switch, it can push a rule to switch telling "forward all the packets with destination MAC address of A on output port 1." The same is the case with a router, except the conditions and actions for the router might be different.

In this project, you will be developing a controller that has the capabilities of both a router and a switch.

First steps

We will be using the Mininet emulation software for this project. A very customized and trimmed version of Linux is available with Mininet installed in it at the github link below.

https://github.com/mininet/openflow-tutorial/wiki

You will be needing Oracle VM Virtual Box, the VM image that has Mininet software, some SSH applications like putty or Mobaxterm, X11 display connectors like Xming etc., to execute the lab. Please go through installing required software section of the tutorial at the above link.

Please go through all the instructions of set up virtual machine and Virtual Box Specific Instructions carefully.

Note: Please ensure that you have the host-only network configured properly in your Virtual Box, as the host only network will be used to SSH into your VM from your Host Operating System (Windows / MAC). As mentioned above, the Mininet VM is a highly trimmed version of Linux and you will not have GUI and it will be cumbersome and difficult to access the VM from the virtual machine console. So, please ensure that your host-only network is setup properly and you can SSH into your VM from your guest OS (Windows / MAC). Once VM is setup, please go through the below link.

Learn Development Tools and create a learning switch.

There are multiple controllers available in Mininet. But you are required to implement the project using Python POX library only. There are three scenarios in the project. Their description is as below.

Scenarios Description

Scenario 1

In this scenario, you will have a single layer 2 switch that has 3 hosts connected to it. Since all the hosts are connected by a layer 2 switch, all the hosts will be in the same subnet. The below is the topology of scenario 1.

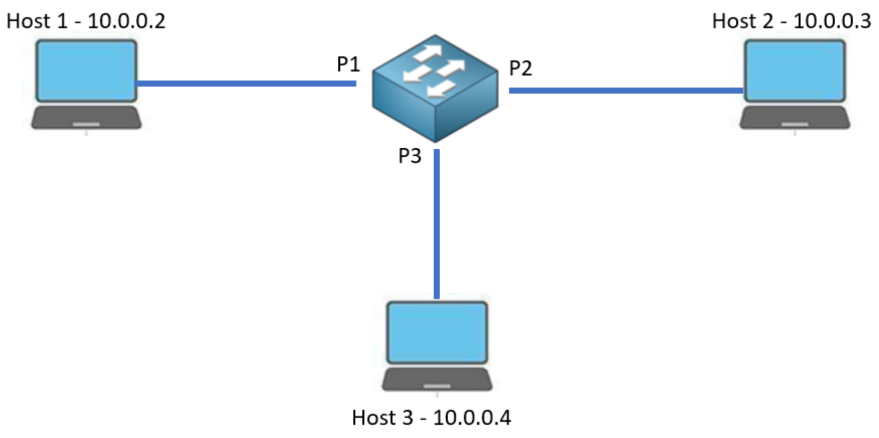

Figure 1 Scenario 1 Topology

The IP addresses of the three hosts are shown above and all the three hosts belong to the network 10.0.0.0/24. The IP address, network ID and subnet mask detail of each host is provided below:

Host Details				
Host	IP Address	Default GW	Network	Subnet Mask
1	10.0.0.2	10.0.0.1	10.0.0.0	255.255.255.0
2	10.0.0.3	10.0.0.1	10.0.0.0	255.255.255.0
3	10.0.0.4	10.0.0.1	10.0.0.0	255.255.255.0

Table 1 Host details for scenario 1

Scenario 2

Topology of Scenario 2 is similar to that of scenario 1. But in this case, the layer 2 switch in scenario 1 will be replaced by a router. So, the three hosts in scenario 2 will be in different subnets. The below is the topology of scenario 1.

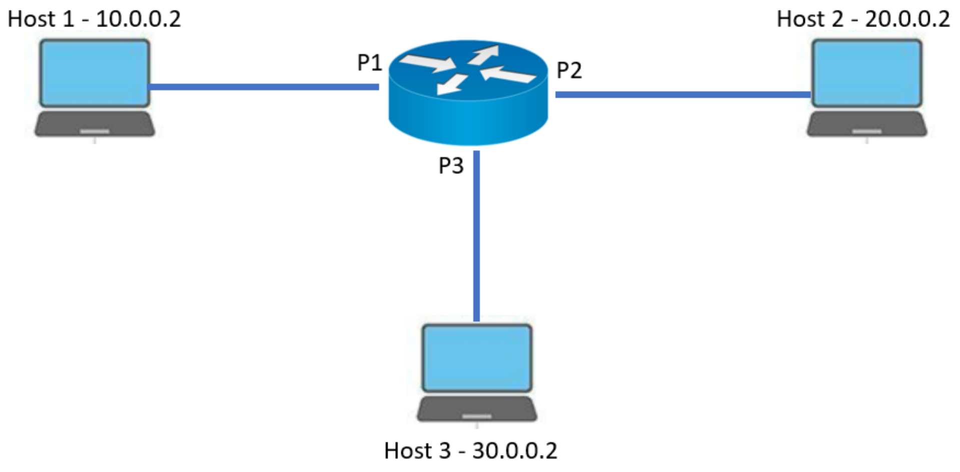

Figure 2 Scenario 2 Topology

The detailed parameters of the host, the router port info and the router routing table are provided here below:

Host Details				
Host	IP Address	Default GW	Network	Subnet Mask
1	10.0.0.2	10.0.0.1	10.0.0.0/24	255.255.255.0
2	20.0.0.2	20.0.0.1	20.0.0.0/24	255.255.255.0
3	30.0.0.2	30.0.0.1	30.0.0.0/24	255.255.255.0

Table 2 Host details for scenario 2

Router Interface Details			
Interface	Network Attached	Interface IP	Interface MAC
1	10.0.0.0/24	10.0.0.1	02:00:DE:AD:BE:11
2	20.0.0.0/24	20.0.0.1	02:00:DE:AD:BE:12
3	30.0.0.0/24	30.0.0.1	02:00:DE:AD:BE:13

Table 3 Router Interface details for scenario 2

Routing Table Info					
Entry no	Network Address	Subnet Mask	Prefix Length	Next Hop Router	Output Interface
1	10.0.0.0	255.255.255.0	24	0.0.0.0	1
2	20.0.0.0	255.255.255.0	24	0.0.0.0	2
3	30.0.0.0	255.255.255.0	24	0.0.0.0	3

Table 4 Routing Table Info for scenario 2

Scenario 3

Scenario 3 is the most complex of all the three and it is designed to be as close as possible to real-life deployments. This scenario has both switches and routers and multiple networks all inter-connected with these switches and routers. The below is the topology for scenario 3.

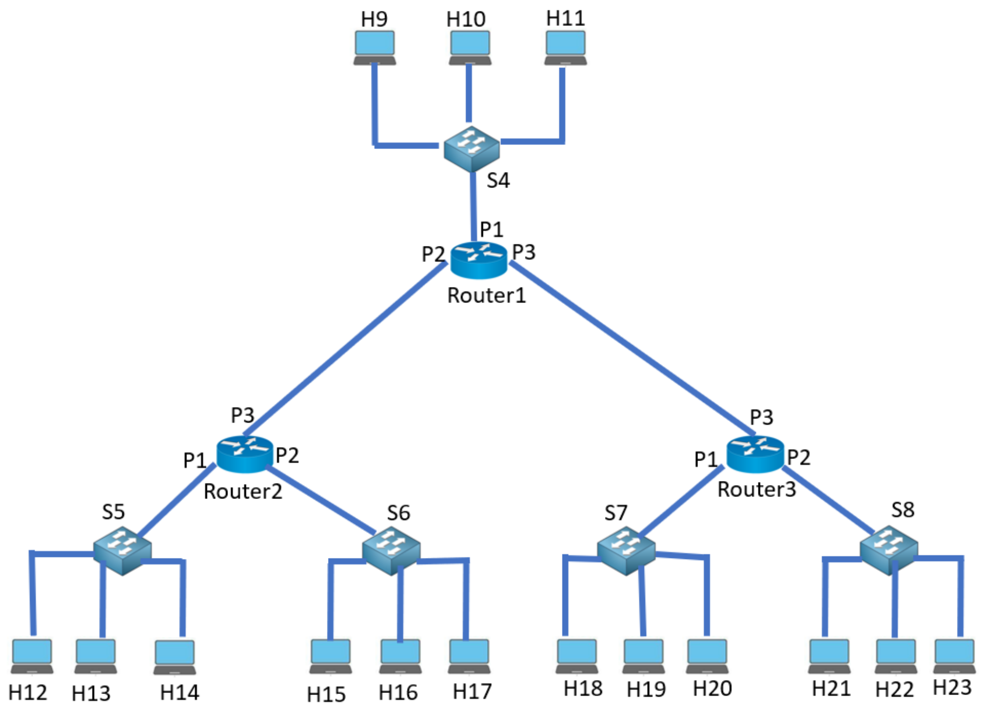

Figure 3 Scenario 3 Topology

In the above figure, h9 to h23 represents hosts, S4 to S8 represents switches and we have three routers Router1, Router2, Router3. The detailed parameters of the hosts, the router port info and routing tables of all the routers are given below:

Host Details				
Host	IP Address	Default GW	Network	Subnet Mask
h9	172.17.16.2	172.17.16.1	172.17.16.0	255.255.255.0
h10	172.17.16.3	172.17.16.1	172.17.16.0	255.255.255.0
h11	172.17.16.4	172.17.16.1	172.17.16.0	255.255.255.0
h12	10.0.0.2	10.0.0.1	10.0.0.0	255.255.255.128
h13	10.0.0.3	10.0.0.1	10.0.0.0	255.255.255.128
h14	10.0.0.4	10.0.0.1	10.0.0.0	255.255.255.128
h15	10.0.0.130	10.0.0.129	10.0.0.128	255.255.255.128
h16	10.0.0.131	10.0.0.129	10.0.0.128	255.255.255.128
h17	10.0.0.132	10.0.0.129	10.0.0.128	255.255.255.128
h18	20.0.0.2	20.0.0.1	20.0.0.0	255.255.255.128
h19	20.0.0.3	20.0.0.1	20.0.0.0	255.255.255.128
h20	20.0.0.4	20.0.0.1	20.0.0.0	255.255.255.128
h21	20.0.0.130	20.0.0.129	20.0.0.128	255.255.255.128
h22	20.0.0.131	20.0.0.129	20.0.0.128	255.255.255.128
h23	20.0.0.132	20.0.0.129	20.0.0.128	255.255.255.128

Table 5 Host details for Scenario 3

Interface Details for Router 1			
Interface	Network Attached	Interface IP	Interface MAC
1	172.17.16.0/24	172.17.16.1	02:00:DE:AD:BE:11
2	192.168.0.0/30	192.168.0.1	02:00:DE:AD:BE:12
3	192.168.0.4/30	192.168.0.5	02:00:DE:AD:BE:13
Interface Details for Router 2			
Interface	Network Attached	Interface IP	Interface MAC
1	10.0.0.0/25	10.0.0.1	02:00:DE:AD:BE:21
2	10.0.0.128/25	10.0.0.129	02:00:DE:AD:BE:22
3	192.168.0.0/30	192.168.0.2	02:00:DE:AD:BE:23
Interface Details for Router 3			
Interface	Network Attached	Interface IP	Interface MAC
1	20.0.0.0/25	20.0.0.1	02:00:DE:AD:BE:31
2	20.0.0.128/25	20.0.0.129	02:00:DE:AD:BE:32
3	192.168.0.4/30	192.168.0.6	02:00:DE:AD:BE:33

Table 6 Router Interface details for scenario 3

Routing Table Info for Router 1					
Entry no	Network Address	Subnet Mask	Prefix Length	Next Hop Router	Output Interface
1	172.17.16.0	255.255.255.0	24	0.0.0.0	1
2	10.0.0.0	255.255.255.0	24	192.168.0.2	2
3	20.0.0.0	255.255.255.0	24	192.168.0.6	3
Routing Table Info for Router 2					
Entry no	Network Address	Subnet Mask	Prefix Length	Next Hop Router	Output Interface
1	172.17.16.0	255.255.255.0	24	192.168.0.1	3
2	0.0.0.0	0.0.0.0	0	192.168.0.1	3
3	10.0.0.0	255.255.255.128	25	0.0.0.0	1
4	10.0.0.128	255.255.255.128	25	0.0.0.0	2
Routing Table Info for Router 3					
Entry no	Network Address	Subnet Mask	Prefix Length	Next Hop Router	Output Interface
1	172.17.16.0	255.255.255.0	24	192.168.0.5	3
2	10.0.0.0	255.255.255.0	24	192.168.0.5	3
3	20.0.0.0	255.255.255.128	25	0.0.0.0	1
4	20.0.0.128	255.255.255.128	25	0.0.0.0	2

Table 7 Routing Table Info for Scenario 3

Files provided to you

You will be performing a total of three scenarios in this project. Please go to the major project folder in the DEN website. There you can find 8 python files and 1 shell script along with this project description pdf. Please download all those files. The below is the list of files and the locations to copy them.

controller_1.py - controller file for scenario 1 - "/home/mininet /pox/pox/misc/"

controller_2.py - controller file for scenario 2 - "/home/mininet /pox/pox/misc/"

controller_3.py - controller file for scenario 3 - "/home/mininet /pox/pox/misc/"

router.py - file for writing the code of router - "/home/mininet /pox/pox/misc/"

switch.py - file for writing the code for switch - "/home/mininet /pox/pox/misc/"

topology_1.py - topology file for scenario 1 - "/home/mininet/mininet/custom/"

topology_2.py - topology file for scenario 2 - "/home/mininet/mininet/custom/"

topology_3.py - topology file for scenario 3 - "/home/mininet/mininet/custom/"

submit.sh - submission script – “/home/mininet/”

Code Description

The files controller_1.py, controller_2.py and controller_3.py are the controller files corresponding to scenario 1, scenario 2 and scenario 3 respectively. The main functions in these files and their description is provided below:

launch ()

This function is invoked whenever a packet forwarding switch connects to the controller. This function then creates a unique object of the Tutorial class for every packet forwarding switch. From this function, the control goes to the tutorial class initializing function __init__()

In the __init__ function, you need to classify whether the device is a router or a switch based on the DPID of the device. The DPIDs will be assigned depending on how you define the devices in your topology files. So, be careful while writing your topology files. For scenario 1 and scenario 2, there is only one device – switch in scenario 1 and router in scenario 2, so you will not have much complexity in these two scenarios. But, be careful while writing the topology file of scenario 3.

Once you know the type of device based on its DPID, you need to initialize all the data structures that are required for it. Read the comments given in the function for more details.

handle_PacketIn ()

Whenever the controller receives a packet from a packet forwarding switch (could be a router or a switch) over OpenFlow, this function gets invoked. “packet” variable that gets created in this function is the actual ethernet frame that the packet forwarding switch has received from other device in the network (other networking device can be a host or another packet forwarding device). The “packet_in” variable that gets created in this function is the OpenFlow header inside which the “packet” ethernet frame is encapsulated and sent to the controller. Depending on whether the packet forwarding switch from which the packet is received is a router or a switch, you should either invoke the switch_handler or the router_handler passing the appropriate arguments. Read the comments given in the function for more details.

switch_handler ()

This function is in the switch.py file and this function is the beginning for your switch code. This function gets invoked when a packet is received by controller from a switch in the topology. This function receives three input arguments. The switch object, the packet and the packet_in. Write the switch code here and if you wish to have any helper functions for switch, write them in the switch.py file. Read the comments given in the function for more details.

router_handler ()

This function is in the router.py file and this function is the beginning for your router code. This function gets invoked when a packet is received by controller from a router in the topology. This function receives three input arguments. The router object, the packet and the packet_in. Write the router code here and if you wish to have any helper functions for router, write them in the router.py file. Read the comments given in the function for more details.

Topology Files

You need to write the code for the topology for the three scenarios in the topology files provided to you. topology_1.py, topology_2.py and topology_3.py are for the topologies of scenario 1, scenario 2 and scenario 3 respectively.

Note: The comments are mentioned as [555 Comments] in each file.

Design of data structures and code

Remember one thing while you design your data structures for switches and routers and the code for switch and router. Even though you have three different scenarios, they will all use two and only two functions viz., switch_handler and router_handler. The first two scenarios are to keep things easy and let you help start with the coding. In real-life deployments there will be many routers and switches and all these devices will be controlled by a single controller. So, your design of the code and data structures should be efficient in the sense that, if a new scenario or topology is provided to you, the only files that you will need to create should be the controller file and the topology file. Your router and switch code should be efficient to support as simple scenarios as scenario 1 and scenario 2 and also as complex scenarios as scenario 3. There are absolutely no restrictions about which data structures to use. You can use any data structures you are comfortable with.

Support required in Switch

Your switch should support the following:

- Broadcast ethernet frames with unknown destination MAC address on all ports
- Broadcast ethernet frames with Layer 2 Broadcast MAC address on all ports
- Broadcast ethernet frames with known destination MAC only on the port corresponding to the MAC
- Once the MAC to port mapping is learnt by the controller for a switch port, it should install OpenFlow rules in the switch so that subsequent packets should not come to the controller. The conditions and actions should be determined by you and the OpenFlow rules should be installed accordingly by controller in the switches in the topology.

Support Required in Router

Your router should support the following:

- Reply to ARP requests that are destined to router.
- Ignore ARP requests that are not destined to router.
- Packet forwarding based on destination IP address in packet header by making a route lookup in the routing table using Longest Prefix Matching (LPM).
- Support for default routes.
- Support for buffering packets if there is no MAC address for a destination host in the router cache. ARP request should be generated by router when a packet destined to a host in a network attached to the routers interface is received. Once the ARP reply is received by the

router, the router should retrieve the queued packets and forward them to the hosts on the appropriate port.

- Support for ICMP Echo Reply (Ping reply if ping request received for router's interface IP address)
- Support for ICMP Network Unreachable - in case router cannot find a route to the destination IP address in the packet.
- Once the controller determines the output port for a destination IP address, it should install OpenFlow rules in the router so that subsequent packets should not come to the controller. The conditions and actions should be determined by you and the OpenFlow rules should be installed accordingly by controller in the routers in the topology.

Commands to start controller and topology

Commands to start controllers to be given from the folder – "/home/mininet/pox"

Commands to start topology to be given from the folder – "/home/mininet/mininet/custom"

Ensure you issue "sudo mn -c" (from any folder) before starting the controller and topology. Always controller should be started first, followed by topology.

Scenario 1

Controller - ./pox.py log.level --DEBUG misc.controller_1

Topology - sudo mn --custom topology_1.py --topo mytopo --mac --controller remote

Scenario 2

Controller - ./pox.py log.level --DEBUG misc.controller_2

Topology - sudo mn --custom topology_2.py --topo mytopo --mac --controller remote

Scenario 3

Controller - ./pox.py log.level --DEBUG misc.controller_3

Topology - sudo mn --custom topology_3.py --topo mytopo --mac --controller remote

Report

Please submit a report in PDF format that contains a high-level description of your router design, switch design and the data structures you used to achieve the functionalities of router and switch. The report must be named "report.pdf" (all small case letters) and be placed in the home folder of the mininet VM i.e, "/home/mininet"

Grading Guidelines

The below are the grading guidelines for the three scenarios.

Scenario 1	Points
h1 ping h2	2
pingall	2
iperf h1 h2	2
rules installation in switch	3

Table 8 Grading guidelines for scenario 1

Scenario 2	Points
h1 ping h2	2
pingall	2
iperf h1 h2	2
rules installation in router	3
h1 ping 10.0.0.1	3
h1 ping 20.0.0.1	3
h1 ping 30.0.0.1	3
h1 ping 8.8.8.8	3

Table 9 Grading guidelines for scenario 2

Scenario 3	Points
h9 ping h10	3
h9 ping h12	3
h9 ping h15	3
h9 ping h18	3
h9 ping h21	3
pingall	3
h9 ping 172.17.16.1	3
h9 ping 10.0.0.1	3
h9 ping 10.0.0.129	3
h9 ping 20.0.0.1	3
h9 ping 20.0.0.129	3
h9 ping 8.8.8.8	3
h12 ping 8.8.8.8	3
h21 ping 8.8.8.8	3
iperf h9 h12	3
iperf h9 h15	3
iperf h9 h18	3
iperf h9 h21	3
rules installation in routers	3
rules installation in switches	3

Table 10 Grading guidelines for scenario 3

Report: 10 Points

Total Score possible: 100 points

Submission Guidelines

Please ensure that all the required files are in place. Three controller files, three topology files, router.py, switch.py and report.

Once you ensure that all the files are in place, go to the home folder of the mininet VM, i.e, "/home/mininet/". The Mininet VM does not come with zip and dos2unix installed. So, run the below two commands in the mininet CLI to install zip and dos2unix.

```
sudo apt-get install zip
sudo apt-get install dos2unix
```

(Ensure that VM has internet connectivity, before issuing these two commands)

Now run the following commands

```
dos2unix submit.sh

chmod 777 submit.sh

./submit.sh <team_member1_first_name>  <team_member2_first_name>
```

Example: If me and sonal were a team and need to submit the project, then the command would be

```
./submit ramnath sonal
```

If you are submitting the project alone and not working with anyone, then enter “none” as the name of the team_member2_first_name.

Ex: `./submit ramnath none`

A zip file will be generated by the script. The zip file will have the name EE555_Spring_2020_<team_mamber1>_<team_member2>. Submit the zip file to the DEN submission folder.