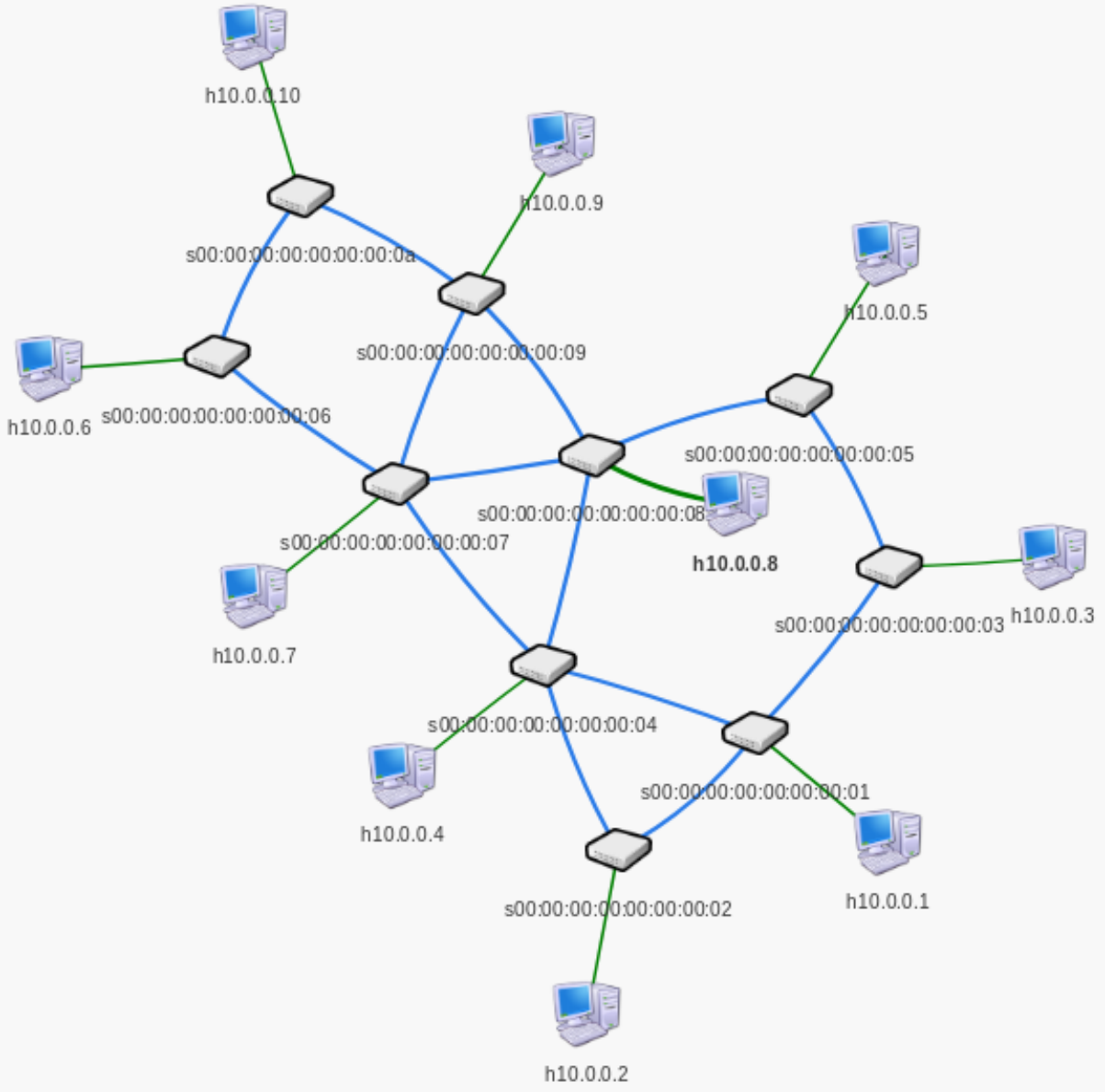
Jaber Daneshamooz, 9325893

Advanced Computer Networks

Software Defined Networking Project

Project Report

Simulation of network with Mininet and Floodlight



The topology of the network

This project contains 3 source codes written in python language. The first one is “project.py” which creates the network topology in Mininet. At the beginning of the code, I created 10 hosts and 10 switches in a for loop naming from h1 to h10 and s1 to s10. Then I connected switches together in a special manner to create the above topology.

In “pusher.py", I created 6 flows and pushed them into floodlight controller. The paths are chose so that the end hosts have maximum distance(in term of hops) and at the same time, have minimum overlap and convergence.

1. Flow from h10 to h3. The path contains s10, s9, s8, s5 and s3 switches.
2. Flow from h6 to h1. The path contains s6, s7, s4 and s1 switches.
3. Flow from h9 to h1. The path contains s9, s7, s8, s4, s2 and s1 switches.
4. Flow from h10 to h2. The path contains s10, s6, s7, s4 and s2 switches.
5. Flow from h3 to h7. The path contains s3, s5, s8 and s7 switches.
6. Flow from h4 to h10. The path contains s4, s8, s9 and s10 switches.

First of all, you should run the Floodlight controller using:

```
$ java -jar target/floodlight.jar
```

Then, you should run “project.py” to create topology in Mininet using the following command:

```
$ sudo mn - - custom project.py - - topo mytopo - -
controller=remote,IP=127.0.0.1,port=6653
```

After that, run the “pusher.py” to push 6 flows into flow tables of switches by:

```
$ python pusher.py
```

After this, you are able to ping hosts through specified path and confirm the correctness of the code by looking at floodlight UI and the packet count associated with each flow.

For providing quality of service in 3 different classes, we should push 3 different entry in flow table of each switch that exist in the path. The only differences between these entries are “ip_tos” and action. I set 3 different TOS for these entries and based on the TOS value, the action would be to enqueue the incoming packet to the related queue(EF, AF or BF). I did this operation for all switches in the path that should support quality of service using DiiffSrv.

I used the following command to create 3 queues at output interface of the switches:

ovs-vsctl set port s10-eth3 qos=@newqos -- --id=@newqos create qos type=linux-htb other-config:max-rate=200000000 queues:1=@EF queues:2=@AF queues:3=@BE -- --id=@EF create queue other-config:min-rate=80000000 other-config:max-rate=120000000 -- --id=@AF create queue other-config:min-rate=30000000 other-config:max-rate=50000000 -- --id=@BE create queue other-config:min-rate=10000000 other-config:max-rate=30000000

This code creates 3 queue at output interface 3 of switch10, for other switches would be the same and I only change the name of switch and port for creating queue on other switches. This code assumes that max-rate of 200Mbps and sets max-rate of 120Mbps, 50Mbps and 30 Mbps for EF, AF and BE classes respectively; and min-rate of 80Mbps, 30Mbps and 10Mbps for EF, AF and BE class.

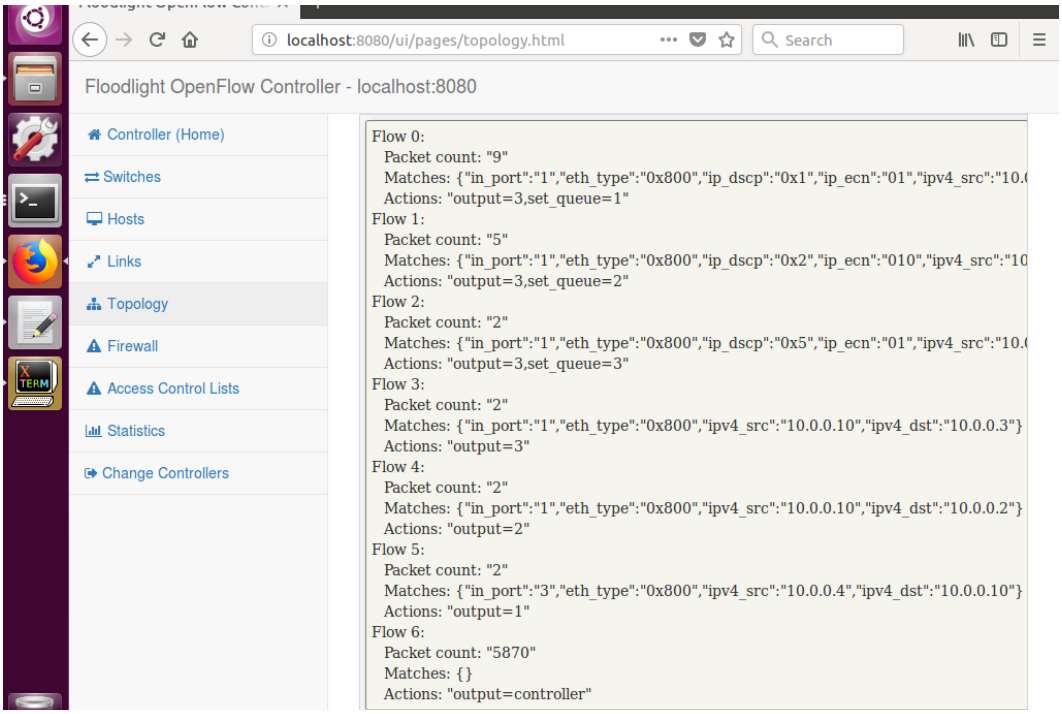
After that, you should run the "quality.py" file to push flows related to QoS to flow tables of switches, then by using ping with modified header(to set DSCP tag) you can verify that it works correctly. The path that I selected to provide QoS contains s10, s9, s8, s5 and s3 switches. It is a flow from h10(source) and h3(destination).

mininet> xterm h10

$ ping 10.0.0.3 -Q <tos>

Put value 5, 10 or 21 as TOS field to indicate EF, AF or BE level of QoS. You can also write this value in hexadecimal format: 0x05, 0x0a or ox15

Here is a picture of s10 flow table with count of packets matched this type of flows:



Flow table of switch 10