Software-Defined Networks

Lab 8

“Push of a Button” Routing

University of Colorado Boulder

Department of Computer Science

Professor Levi Perigo, Ph.D.

# Summary

The decoupled control and data planes enabled by SDN allows enhancements to the traditional methods of routing. Since the centralized controller has a global view of the network, routing policies can be configured by a “push of a button”. The purpose of this lab to develop an application that allows for different flavors of routing to be implemented on the virtual setup. The objectives of this lab are to be used as guidelines, and additional exploration by the student is strongly encouraged.

# Objective 1 – Network topology

# 

1. Set up the above topology on the Mininet VM.
   1. Create a custom Mininet topology which creates the above interconnections between OvS’s and hosts. The server can be just another host node in Mininet. The switch names, hostnames, and IP addresses should be configured as shown.
2. Connect all switches to a remote controller running on the Controllers VM. Use any controller of your choice.
3. Paste screenshots of the custom topology file and the created topology in the controller’s UI. [**10 points**]

A computer code with a diagram of a number

Description automatically generated with medium confidence

A screen shot of a computer program

Description automatically generated

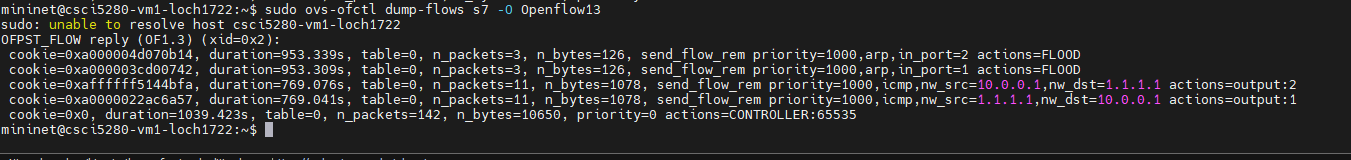
1. Run a HTTP server on port 8080 on the server node.

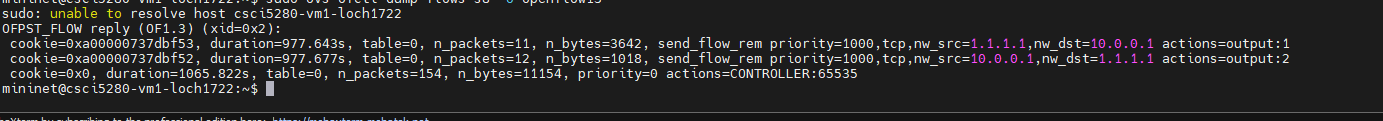
# Objective 2 – Path routing application

1. The default path for all traffic between the host and server should be OvS 1 <-> OvS 6 <-> OvS 7 <-> OvS 5.
2. Create an application that at a “push of a button” you can force *specific* traffic to take each of the following:
   1. Shortest path – HTTP traffic from host to server should take the shortest path while all other traffic takes the default path. Paste relevant screenshots of the application working. [**30 points**]

A screen shot of a computer

Description automatically generated



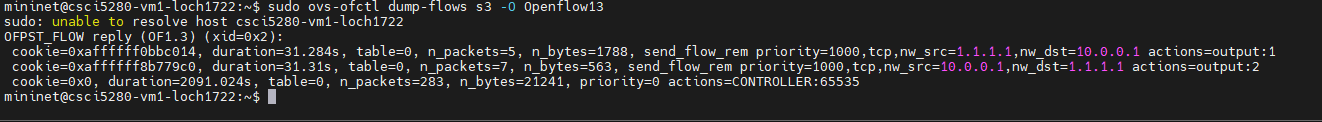


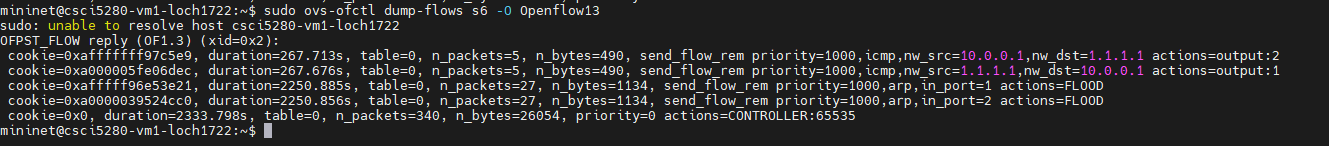
These 2 screenshots shows that the behavior wanted above is working. 1st screenshot of flows s7 should only see packet flow of icmp traffic which makes sense, and screenshot 2 of s8 flows should only see tcp traffic which is sees. This concludes that ‘other’ traffic takes the default route and http traffic takes the shortest path.

* 1. Longest path – HTTP traffic from host to server should take the longest path while all other traffic takes the default path. Paste relevant screenshots of the application working. [**30 points**]

A screen shot of a computer

Description automatically generated





Here we can see the http traffic is taking the longest path given by flows on s3. We see that tcp traffic is flowing and that icmp traffic is flowing on the default route on s6.

* 1. Path selection on best delay – Simulate additional delay of 100 ms on the link between OvS 1 and OvS 8, and of 50 ms on the link between OvS 1 and OvS 6; HTTP traffic should take the path with the best end-to-end delay while all other traffic takes the default path. Paste relevant screenshots of the application working. [**50 points**]

A black screen with white text

Description automatically generated

The way this works is I have an api call that calculates the fastest path and outputs the corresponding latency, I pick out the fastest latency and then set the path for the lowest latency.

**A black and white screen

Description automatically generated**

Here we see that the longest path had the lowest latency and will be chosen for http traffic. Other traffic will take the default path.

* 1. Create a web front end of your application. Paste relevant screenshots of the application working. [**30 points**]

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

1. Submit your script along with the report.

# Objective 3 – Additional Requirements

1. REST API must be used somewhere by your application. Paste screenshots of your script where you use REST. [**15 points**]

As you see above, my applications solely relies on REST API calls to push the flows via the Floodlight static flow pusher tool. Here’s an example screenshot of the shortest path being created via ALL REST API calls:

A screen shot of a computer

Description automatically generated

1. IPv6 must be used somewhere in the network. Paste relevant screenshots. [**15 points**]
2. Prove that your application works for each objective – shortest path, longest path and path selection on best delay. Think about and perform tests on how you could prove that the application selects different paths for ICMP and HTTP traffic.

# Objective 4 – Automated Testing [Extra Credit]

Create an automated test framework in your Web GUI to test the application functionality for each objective. For each push of the button, the application should configure the network according to requirement as well as display the paths (DPIDs) traversed for both ICMP and HTTP traffic on the GUI. [**20 points**]

# Total Points \_\_\_\_\_\_\_ / 180 + 20 Bonus