Software-Defined Networks

Lab 8
“Push of a Button” Routing

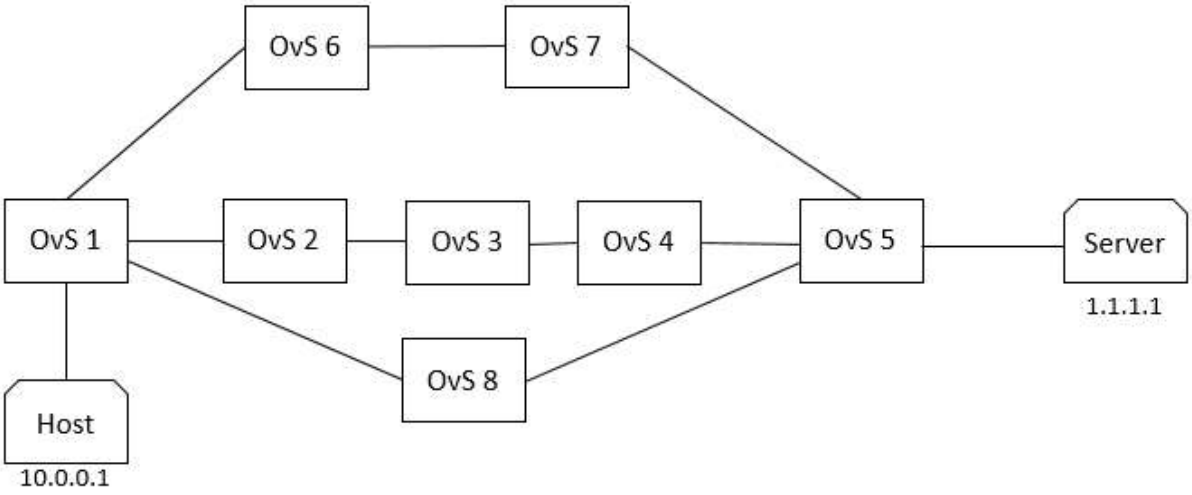
University of Colorado Boulder
Department of Computer Science

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.

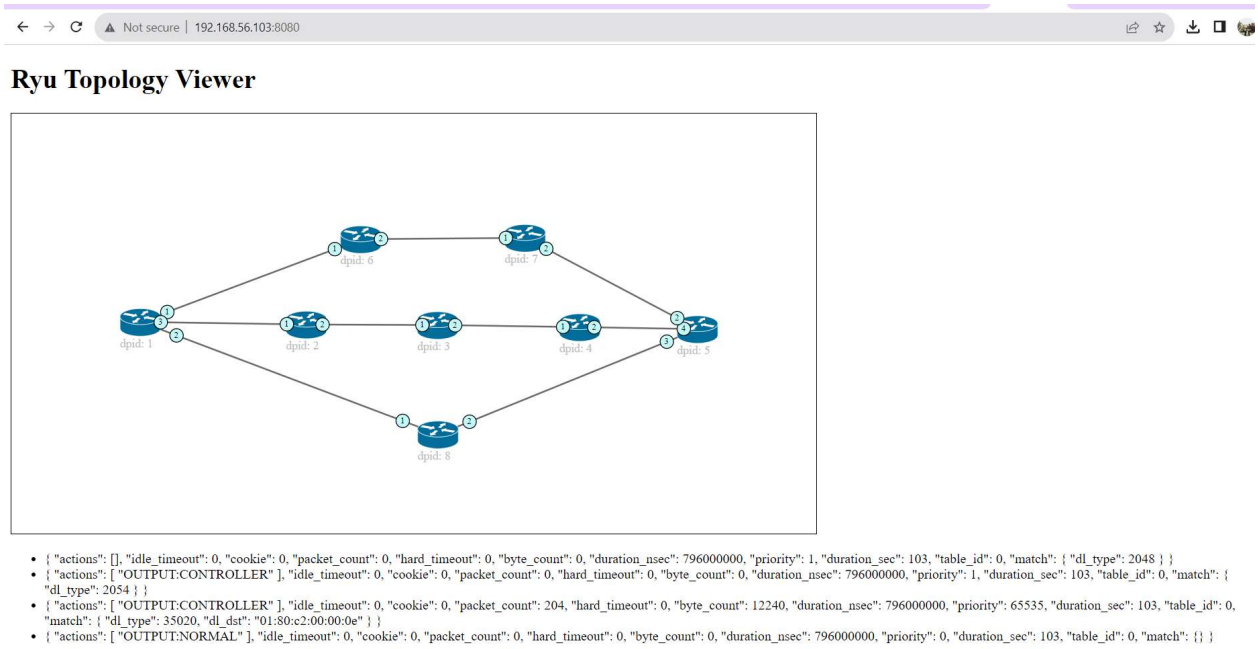
Objective 1 – Network topology



1. Set up the above topology on the Mininet VM.
    a. 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. → RYU
3. Paste screenshots of the custom topology file and the created topology in the controller's UI. [**10 points**]

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    RyuRemoteController=net.addController(name='RyuRemoteController',
                      controller=RemoteController,
                      ip='192.168.56.101',
                      protocol='tcp',
                      port=6653)

    info( '*** Add switches\n')
    ovs3 = net.addSwitch('ovs3', cls=OVSKernelSwitch)
    ovs4 = net.addSwitch('ovs4', cls=OVSKernelSwitch)
    ovs1 = net.addSwitch('ovs1', cls=OVSKernelSwitch)
    ovs6 = net.addSwitch('ovs6', cls=OVSKernelSwitch)
    ovs2 = net.addSwitch('ovs2', cls=OVSKernelSwitch)
    ovs5 = net.addSwitch('ovs5', cls=OVSKernelSwitch)
    ovs7 = net.addSwitch('ovs7', cls=OVSKernelSwitch)
    ovs8 = net.addSwitch('ovs8', cls=OVSKernelSwitch)

    info( '*** Add hosts\n')
    server = net.addHost('server', cls=Host, ip='1.1.1.1/24', defaultRoute='via 1.1.1.254/24')
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute='via 10.0.0.254')

    info( '*** Add links\n')
    net.addLink(ovs5, server)
    net.addLink(ovs1, ovs6)
    net.addLink(ovs6, ovs7)
    net.addLink(ovs7, ovs5)
    net.addLink(ovs1, ovs8)
    net.addLink(ovs8, ovs5)
    net.addLink(ovs1, ovs2)
    net.addLink(ovs2, ovs3)
    net.addLink(ovs3, ovs4)
    net.addLink(ovs4, ovs5)
    net.addLink(ovs1, h1)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
"lab8.py" [readonly] 77L, 2545C
```

Topology -



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

Objective 2 – Path routing application

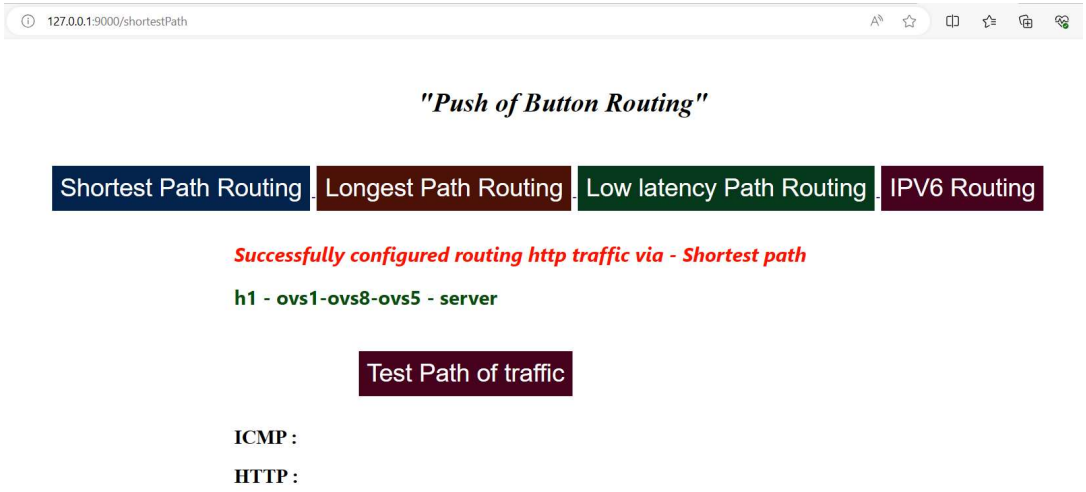
1. The default path for all traffic between the host and server should be OvS 1 <-> OvS 6 <-> OvS 7 <-> OvS 5.

2. Create an application that at a “push of a button” you can force *specific* traffic to take each of the following:

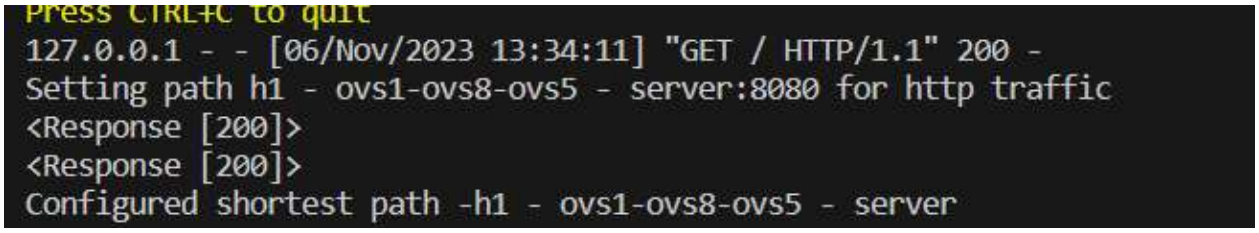
a. Shortest path – HTTP traffic from host to server should take the shortest path while all other traffic takes the default path. Paste relevant screenshots of the application working. [**30 points**]

Click on Shortest Path Routing

Output -



Code Logs -



Click on test Path of traffic to verify the flow of http and ICMP traffic from h1 to server

"Push of Button Routing"

Shortest Path Routing | Longest Path Routing | Low latency Path Routing | IPV6 Routing

Successfully tested traffic

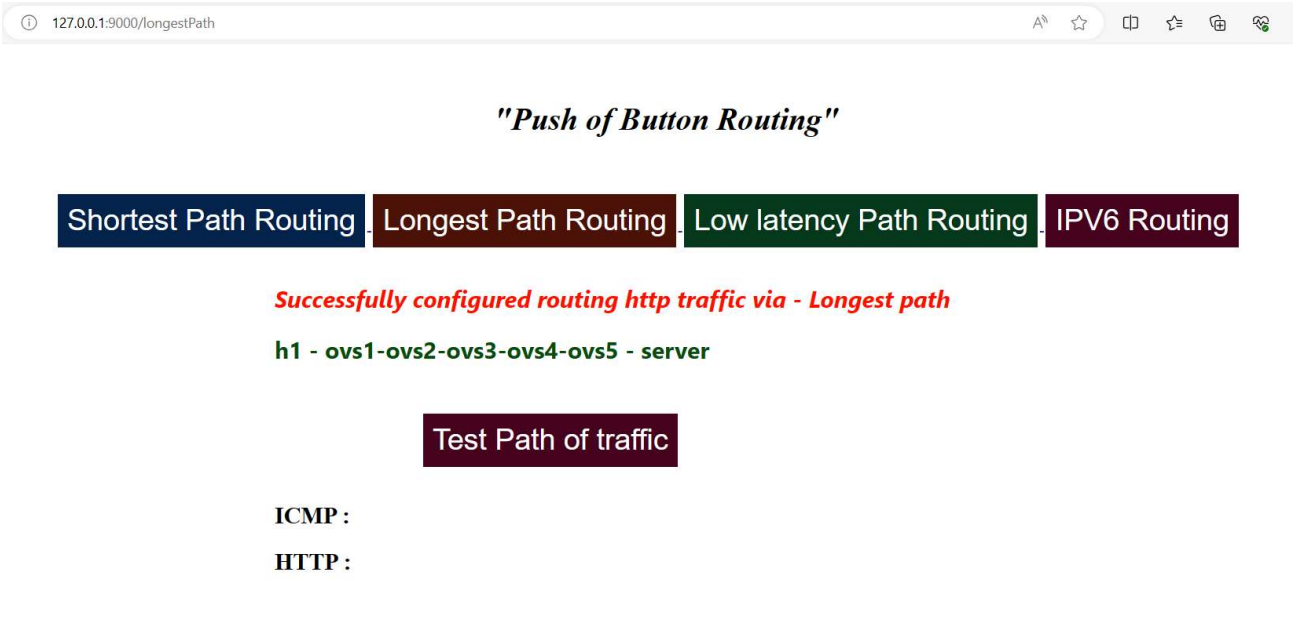
Test Path of traffic

ICMP : ovs1-ovs6-ovs7-ovs5

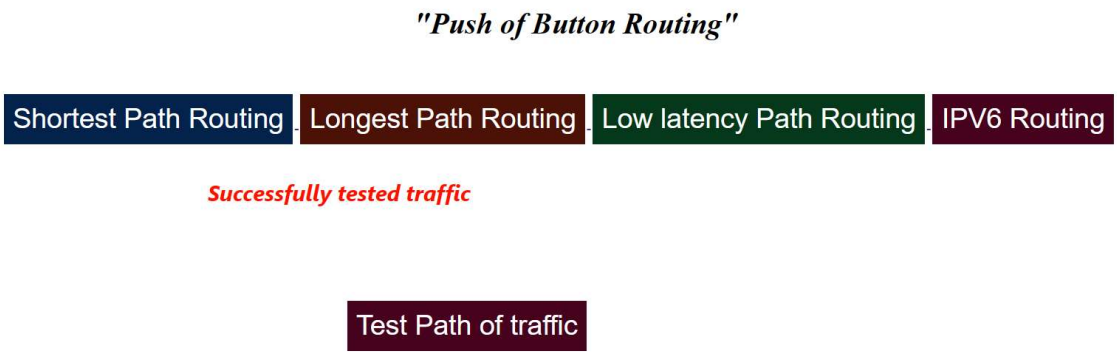
HTTP : ovs1-ovs8-ovs5

```
<Response [200]>
Configured shortest path -h1 - ovs1-ovs8-ovs5 - server
127.0.0.1 - - [06/Nov/2023 13:43:49] "GET /shortestPath HTTP/1.1" 200 -
R1 port stats
{3: (5, 8), 1: (10, 13), 4: (6, 14), 2: (4, 8)}
R5 port stats
{3: (4, 8), 1: (7, 14), 2: (9, 12), 4: (4, 8)}
R1 port stats
{3: (5, 8), 1: (20, 23), 4: (16, 24), 2: (4, 8)}
R5 port stats
{3: (4, 8), 1: (18, 25), 2: (19, 22), 4: (4, 8)}
{'R1': {3: (5, 8), 1: (10, 13), 4: (6, 14), 2: (4, 8)}, 'R5': {3: (4, 8), 1: (7, 14), 2: (9, 12), 4: (4, 8)}}
{'R1': {3: (5, 8), 1: (20, 23), 4: (16, 24), 2: (4, 8)}, 'R5': {3: (4, 8), 1: (18, 25), 2: (19, 22), 4: (4, 8)}}
INPUT
{'R1': {3: (5, 8), 1: (10, 13), 4: (6, 14), 2: (4, 8)}, 'R5': {3: (4, 8), 1: (7, 14), 2: (9, 12), 4: (4, 8)}}
{'R1': {3: (5, 8), 1: (20, 23), 4: (16, 24), 2: (4, 8)}, 'R5': {3: (4, 8), 1: (18, 25), 2: (19, 22), 4: (4, 8)}}
Change Rate is
{'R1': {1: (10, 10), 2: (0, 0), 3: (0, 0), 4: (10, 10)}, 'R5': {1: (11, 11), 2: (10, 10), 3: (0, 0), 4: (0, 0)}}
------Change Rate
{'R1': {1: (10, 10), 2: (0, 0), 3: (0, 0), 4: (10, 10)}, 'R5': {1: (11, 11), 2: (10, 10), 3: (0, 0), 4: (0, 0)}}
ICMP Path is ---> ovs1-ovs6-ovs7-ovs5
-------------------- http ----------------------------
R1 port stats
{3: (5, 8), 1: (20, 23), 4: (16, 24), 2: (4, 8)}
R5 port stats
{3: (4, 8), 1: (18, 25), 2: (19, 22), 4: (4, 8)}
R1 port stats
{3: (5, 8), 1: (20, 23), 4: (22, 30), 2: (10, 14)}
R5 port stats
{3: (10, 14), 1: (24, 31), 2: (19, 22), 4: (4, 8)}
INPUT
{'R1': {3: (5, 8), 1: (20, 23), 4: (16, 24), 2: (4, 8)}, 'R5': {3: (4, 8), 1: (18, 25), 2: (19, 22), 4: (4, 8)}}
{'R1': {3: (5, 8), 1: (20, 23), 4: (22, 30), 2: (10, 14)}, 'R5': {3: (10, 14), 1: (24, 31), 2: (19, 22), 4: (4, 8)}}
Change Rate is
{'R1': {1: (0, 0), 2: (6, 6), 3: (0, 0), 4: (6, 6)}, 'R5': {1: (6, 6), 2: (0, 0), 3: (6, 6), 4: (0, 0)}}
127.0.0.1 - - [06/Nov/2023 13:44:50] "GET /testTraffic HTTP/1.1" 200 -
```

b. 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**]



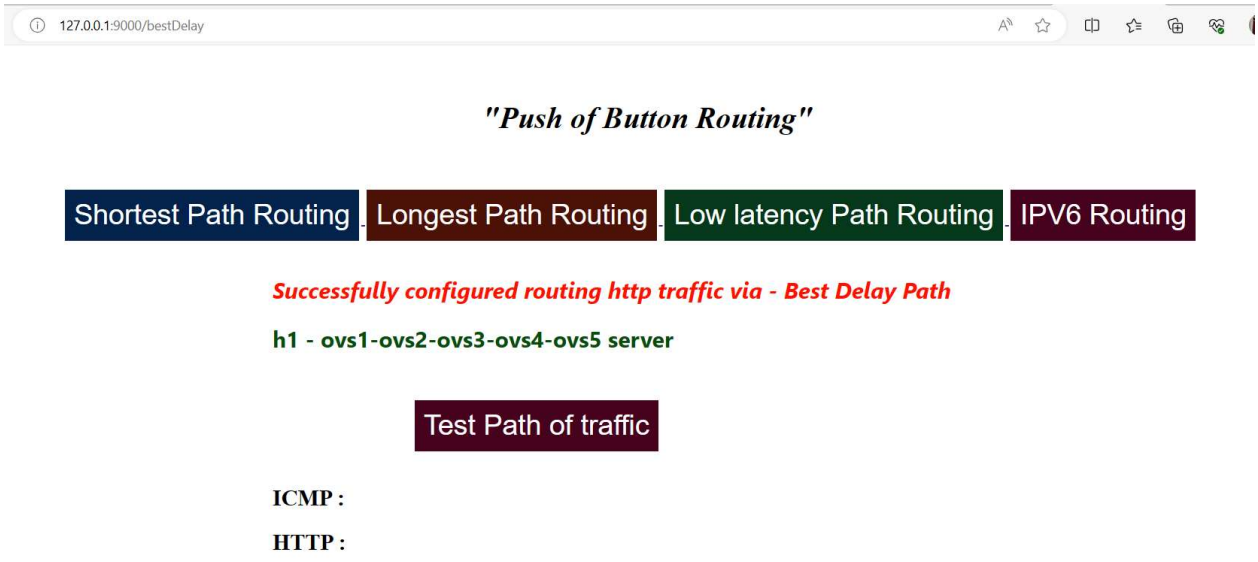
Once longest Path is configured, test the path taken by traffic



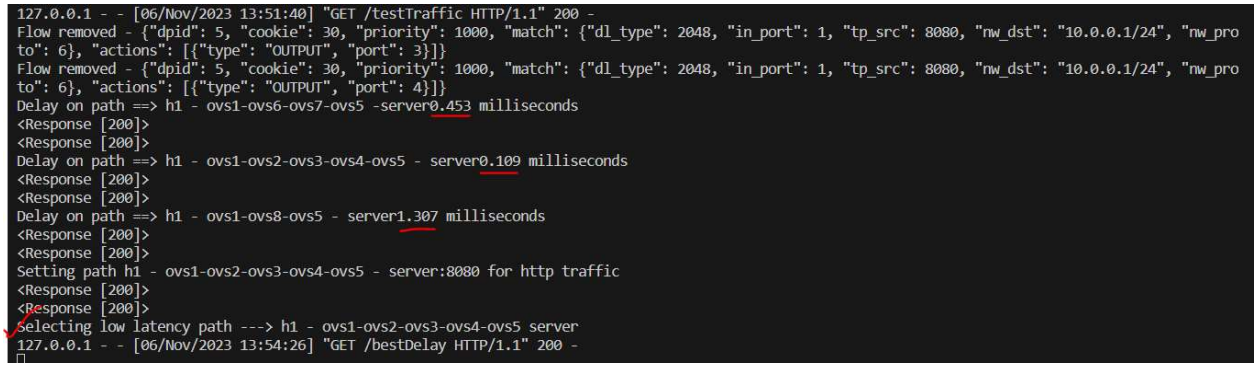
Logs -

```
<Response [200]>
Configured longest path -h1 - ovs1-ovs2-ovs3-ovs4-ovs5 - server
127.0.0.1 - - [06/Nov/2023 13:48:01] "GET /longestPath HTTP/1.1" 200 -
R1 port stats
{3: (5, 8), 1: (20, 23), 4: (22, 30), 2: (10, 14)}
R5 port stats
{3: (10, 14), 1: (24, 31), 2: (19, 22), 4: (4, 8)}
R1 port stats
{3: (5, 8), 1: (30, 33), 4: (32, 40), 2: (10, 14)}
R5 port stats
{3: (10, 14), 1: (35, 42), 2: (29, 32), 4: (4, 8)}
{'R1': {3: (5, 8), 1: (20, 23), 4: (22, 30), 2: (10, 14)}, 'R5': {3: (10, 14), 1: (24, 31), 2: (19, 22), 4: (4, 8)}}
{'R1': {3: (5, 8), 1: (30, 33), 4: (32, 40), 2: (10, 14)}, 'R5': {3: (10, 14), 1: (35, 42), 2: (29, 32), 4: (4, 8)}}
INPUT
{'R1': {3: (5, 8), 1: (20, 23), 4: (22, 30), 2: (10, 14)}, 'R5': {3: (10, 14), 1: (24, 31), 2: (19, 22), 4: (4, 8)}}
{'R1': {3: (5, 8), 1: (30, 33), 4: (32, 40), 2: (10, 14)}, 'R5': {3: (10, 14), 1: (35, 42), 2: (29, 32), 4: (4, 8)}}
Change Rate is
{'R1': {1: (10, 10), 2: (0, 0), 3: (0, 0), 4: (10, 10)}, 'R5': {1: (11, 11), 2: (10, 10), 3: (0, 0), 4: (0, 0)}}
------Change Rate
{'R1': {1: (10, 10), 2: (0, 0), 3: (0, 0), 4: (10, 10)}, 'R5': {1: (11, 11), 2: (10, 10), 3: (0, 0), 4: (0, 0)}}
ICMP Path is ---> ovs1-ovs6-ovs7-ovs5
--------------------- http ----------------------------
R1 port stats
{3: (5, 8), 1: (30, 33), 4: (32, 40), 2: (10, 14)}
R5 port stats
{3: (10, 14), 1: (35, 42), 2: (29, 32), 4: (4, 8)}
R1 port stats
{3: (10, 15), 1: (30, 33), 4: (39, 45), 2: (10, 14)}
R5 port stats
{3: (10, 14), 1: (40, 49), 2: (29, 32), 4: (11, 13)}
INPUT
{'R1': {3: (5, 8), 1: (30, 33), 4: (32, 40), 2: (10, 14)}, 'R5': {3: (10, 14), 1: (35, 42), 2: (29, 32), 4: (4, 8)}}
{'R1': {3: (10, 15), 1: (30, 33), 4: (39, 45), 2: (10, 14)}, 'R5': {3: (10, 14), 1: (40, 49), 2: (29, 32), 4: (11, 13)}}
Change Rate is
{'R1': {1: (0, 0), 2: (0, 0), 3: (5, 7), 4: (7, 5)}, 'R5': {1: (5, 7), 2: (0, 0), 3: (0, 0), 4: (7, 5)}}
127.0.0.1 - - [06/Nov/2023 13:51:40] "GET /testTraffic HTTP/1.1" 200 -
```

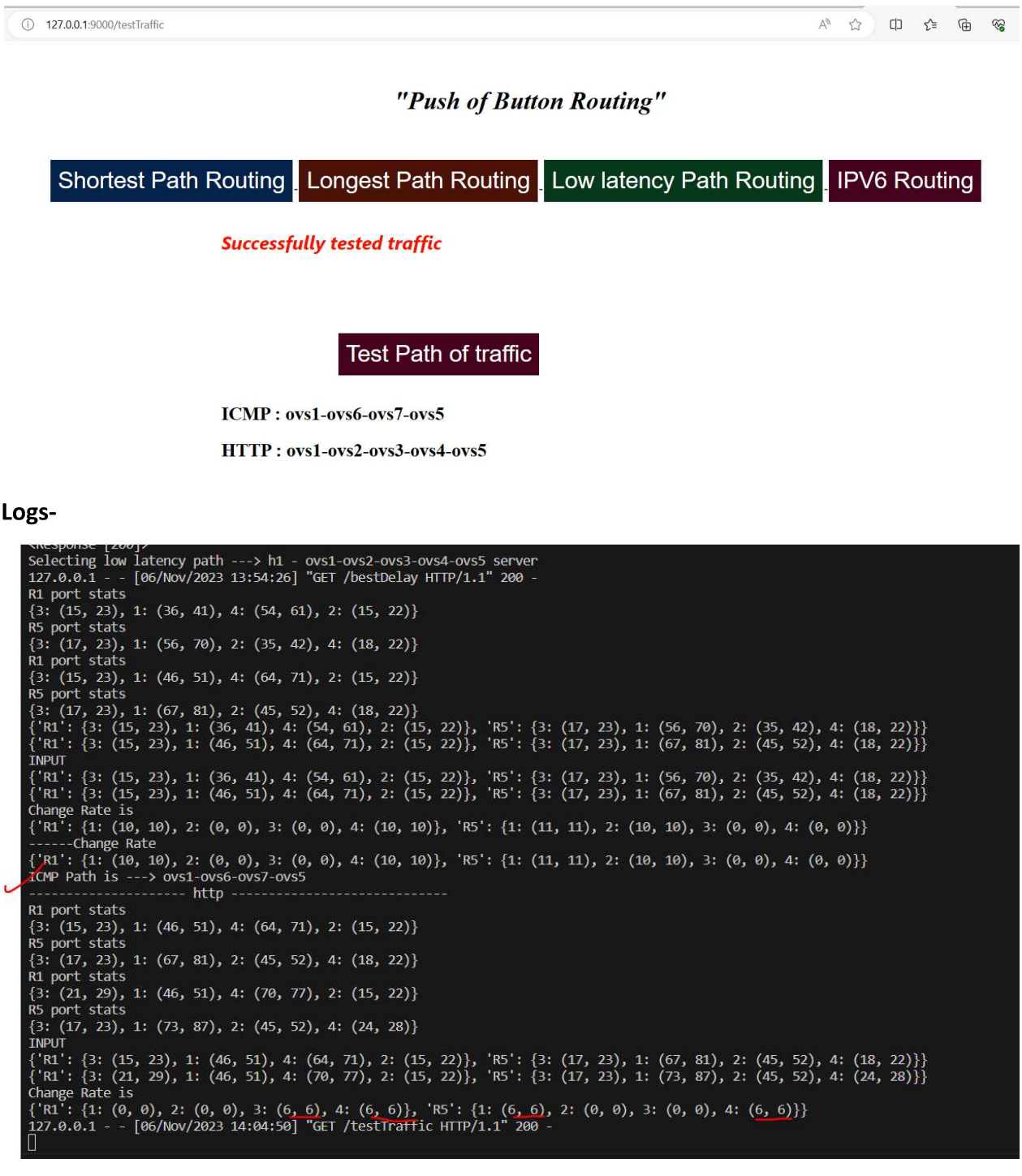
c. Path selection on best delay – Simulate additional delay of 100 ms on the link between OvS 1 and OvS 8, and of 50 ms on the link between OvS 1 and OvS 6; HTTP traffic should take the path with the best end-to-end delay while all other traffic takes the default path. Paste relevant screenshots of the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



Logs -



Test Path of traffic -



Logs-

```
Selecting low latency path ---> h1 - ovs1-ovs2-ovs3-ovs4-ovs5 server
127.0.0.1 - - [06/Nov/2023 13:54:26] "GET /bestDelay HTTP/1.1" 200 -
R1 port stats
{3: (15, 23), 1: (36, 41), 4: (54, 61), 2: (15, 22)}
R5 port stats
{3: (17, 23), 1: (56, 70), 2: (35, 42), 4: (18, 22)}
R1 port stats
{3: (15, 23), 1: (46, 51), 4: (64, 71), 2: (15, 22)}
R5 port stats
{3: (17, 23), 1: (67, 81), 2: (45, 52), 4: (18, 22)}
{'R1': {3: (15, 23), 1: (36, 41), 4: (54, 61), 2: (15, 22)}, 'R5': {3: (17, 23), 1: (56, 70), 2: (35, 42), 4: (18, 22)}}
{'R1': {3: (15, 23), 1: (46, 51), 4: (64, 71), 2: (15, 22)}, 'R5': {3: (17, 23), 1: (67, 81), 2: (45, 52), 4: (18, 22)}}
INPUT
{'R1': {3: (15, 23), 1: (36, 41), 4: (54, 61), 2: (15, 22)}, 'R5': {3: (17, 23), 1: (56, 70), 2: (35, 42), 4: (18, 22)}}
{'R1': {3: (15, 23), 1: (46, 51), 4: (64, 71), 2: (15, 22)}, 'R5': {3: (17, 23), 1: (67, 81), 2: (45, 52), 4: (18, 22)}}
Change Rate is
{'R1': {1: (10, 10), 2: (0, 0), 3: (0, 0), 4: (10, 10)}, 'R5': {1: (11, 11), 2: (10, 10), 3: (0, 0), 4: (0, 0)}}
------Change Rate
{'R1': {1: (10, 10), 2: (0, 0), 3: (0, 0), 4: (10, 10)}, 'R5': {1: (11, 11), 2: (10, 10), 3: (0, 0), 4: (0, 0)}}
ICMP Path is ---> ovs1-ovs6-ovs7-ovs5
---------- http ----------
R1 port stats
{3: (15, 23), 1: (46, 51), 4: (64, 71), 2: (15, 22)}
R5 port stats
{3: (17, 23), 1: (67, 81), 2: (45, 52), 4: (18, 22)}
R1 port stats
{3: (21, 29), 1: (46, 51), 4: (70, 77), 2: (15, 22)}
R5 port stats
{3: (17, 23), 1: (73, 87), 2: (45, 52), 4: (24, 28)}
INPUT
{'R1': {3: (15, 23), 1: (46, 51), 4: (64, 71), 2: (15, 22)}, 'R5': {3: (17, 23), 1: (67, 81), 2: (45, 52), 4: (18, 22)}}
{'R1': {3: (21, 29), 1: (46, 51), 4: (70, 77), 2: (15, 22)}, 'R5': {3: (17, 23), 1: (73, 87), 2: (45, 52), 4: (24, 28)}}
Change Rate is
{'R1': {1: (0, 0), 2: (0, 0), 3: (6, 6), 4: (6, 6)}, 'R5': {1: (6, 6), 2: (0, 0), 3: (0, 0), 4: (6, 6)}}
127.0.0.1 - - [06/Nov/2023 14:04:50] "GET /testTraffic HTTP/1.1" 200 -
```

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

→ ***shortest path, longest path, best delay***

screenshots attached above

3. 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**]

Data Structure for below code

```
ipAddressTopology = {
    #interfaces on routers
    "r1": ["10.0.0.254/24","2.2.2.1/24","3.3.3.1/24","4.4.4.1/24"],
    "r2": ["3.3.3.2/24","7.7.7.2/24"],
    "r3": ["7.7.7.1/24","8.8.8.1/24"],
    "r4": ["8.8.8.2/24","9.9.9.2/24"],
    "r5": ["10.10.10.2/24","9.9.9.1/24","5.5.5.1/24","1.1.1.254/24"],
    "r6": ["2.2.2.2/24","6.6.6.2/24"],
    "r7": ["6.6.6.1/24","10.10.10.1/24"],
    "r8": ["4.4.4.2/24","5.5.5.2/24"]
}
routes = {
    #prefix
    "1.1.1.0/24": {
        #nexthop
        "r1": ["2.2.2.2","3.3.3.2","4.4.4.2"],
        "r2": ["7.7.7.1"],
        "r3": ["8.8.8.2"],
        "r4": ["9.9.9.1"],
        "r6": ["6.6.6.1"],
        "r7": ["10.10.10.2"],
        "r8": ["5.5.5.1"]
    },
    #prefix
    "10.0.0.0/24": {
        #next hops
        "r2": ["3.3.3.1"],
        "r3": ["7.7.7.2"],
        "r4": ["8.8.8.1"],
        "r5": ["10.10.10.1","9.9.9.2","5.5.5.2"],
        "r6": ["2.2.2.1"],
        "r7": ["6.6.6.2"],
        "r8": ["4.4.4.1"]
    }
}
```

Everything in the code is a REST API call to the controller using python.requests library

```
#install ips
for ipAddr in ipAddressTopology[router]:
    payload = json.dumps({ "address": ipAddr })
    response = requests.request("POST", url=apiEndpoint, \
                                headers={'Content-Type': 'application/json'}, \
                                data=payload)
    print(response)
    #time.sleep(2)
    if response.status_code == 200:
        print("Router: "+router+" --> "+ payload+" ip addresses configured")
    else:
        sys.exit()

#install static routes
for prefix in routes.keys():
    if router not in routes[prefix]:
        continue

    nextHops = routes[prefix][router]
    #prefix
    #nextHops
    for nexthop in nextHops:
        payload = json.dumps({"destination": prefix, "gateway": nexthop })
        response = requests.request("POST", url=apiEndpoint, \
                                headers={'Content-Type': 'application/json'}, \
                                data=payload)
        #time.sleep(2)
        if response.status_code == 200:
            print("Router: "+router+" --> "+ payload+" route configured")
        else:
            sys.exit()
        break
```

2. IPv6 must be used somewhere in the network. Paste relevant screenshots. [**15 points**]

"Push of Button Routing"

Shortest Path Routing | Longest Path Routing | Low latency Path Routing | IPV6 Routing

Successfully configured routing for IPV6 traffic

(IPV6) h1 -ovs1-ovs8-ovs5 - server (IPv6)

Test Path of traffic

ICMP : PING 5501::2(5501::2) 56 data bytes 64 bytes from 5501::2: icmp_seq=1 ttl=64 time=50.7 ms 64 bytes from 5501::2: icmp_seq=2 ttl=64 time=1.98 ms 64 bytes from 5501::2: icmp_seq=3 ttl=64 time=1.99 ms 64 bytes from 5501::2: icmp_seq=4 ttl=64 time=1.98 ms 64 bytes from 5501::2: icmp_seq=5 ttl=64 time=1.98 ms 64 bytes from 5501::2: icmp_seq=6 ttl=64 time=1.95 ms 64 bytes from 5501::2: icmp_seq=7 ttl=64 time=1.45 ms 64 bytes from 5501::2: icmp_seq=8 ttl=64 time=1.76 ms 64 bytes from 5501::2: icmp_seq=9 ttl=64 time=1.97 ms 64 bytes from 5501::2: icmp_seq=10 ttl=64 time=2.01 ms --- 5501::2 ping statistics --- 10 packets transmitted, 10 received, 0% packet loss, time 9008ms rtt min/avg/max/mdev = 1.453/6.786/50.755/14.657 ms mininet>

```
127.0.0.1 - - [06/Nov/2023 14:20:09] "GET /bestDelay HTTP/1.1" 500 -
Setting path h1 - ovs1-ovs8-ovs5 - server for IPV6 traffic
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
IPV6 path configured
PING 5501::2(5501::2) 56 data bytes
64 bytes from 5501::2: icmp_seq=1 ttl=64 time=50.7 ms
64 bytes from 5501::2: icmp_seq=2 ttl=64 time=1.98 ms
64 bytes from 5501::2: icmp_seq=3 ttl=64 time=1.99 ms
64 bytes from 5501::2: icmp_seq=4 ttl=64 time=1.98 ms
64 bytes from 5501::2: icmp_seq=5 ttl=64 time=1.98 ms
64 bytes from 5501::2: icmp_seq=6 ttl=64 time=1.95 ms
64 bytes from 5501::2: icmp_seq=7 ttl=64 time=1.45 ms
64 bytes from 5501::2: icmp_seq=8 ttl=64 time=1.76 ms
64 bytes from 5501::2: icmp_seq=9 ttl=64 time=1.97 ms
64 bytes from 5501::2: icmp_seq=10 ttl=64 time=2.01 ms

--- 5501::2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9008ms
rtt min/avg/max/mdev = 1.453/6.786/50.755/14.657 ms
mininet>
127.0.0.1 - - [06/Nov/2023 14:20:24] "GET /ipv6 HTTP/1.1" 200 -
```

3. 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.

   Pasted screenshot of testing immediately after configured,

   Functionality provided → Test traffic button

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**]

The test traffic path button does following

- There are three possible paths for h1 to reach server

  h1 - **ovs1 - ovs6 - ovs7 - ovs5** - server

  h1 - **ovs1 - ovs2 - ovs3 - ovs4 - ovs5** - server

  h1 - **ovs1 -ovs8 -ovs5 -** server

Once the required path intent is configured - ***shortest, longest, best delay.***

- The algorithm captures stats on both **OVS1 and OVS5** switch before the traffic and after the traffic is initiated.
- A **delta/difference** value of **rx and tx** packets is calculated on each of the ports.
- This will give which port experiences the change for sending and receiving packets.
- Using this method, we can identify the path that is taken.

Screenshot of each of the case if captured and is tested in each case

Total Points _____ / 180 + 20 Bonus