

Software-Defined Networking

Assignment 05

Peipei He, st169741@stud.uni-stuttgart.de, 3442500
Huicheng Qian, st169665@stud.uni-stuttgart.de, 3443114
Kuang-Yu Li, st169971@stud.uni-stuttgart.de, 3440829

Task1

simple_firewall

start our pyretic firewall module:

```
ex5$ pyretic.py -v low -m r0 simple_firewall
```

start mininet and ssh & web services on srv & inet, and a xterm for all nodes:

```
ex5$ sudo ./mininet5.py mininet
```

```
> startservers mininet
```

```
> xterm h1 h2 srv inet mon
```

On h1, h2, and inet access web & ssh services through:

```
curl -sS -m 5 10.0.0.X:Y0Y0
```

On mon start monitoring with the packet capture tool `tcpdump` to print information about received packets

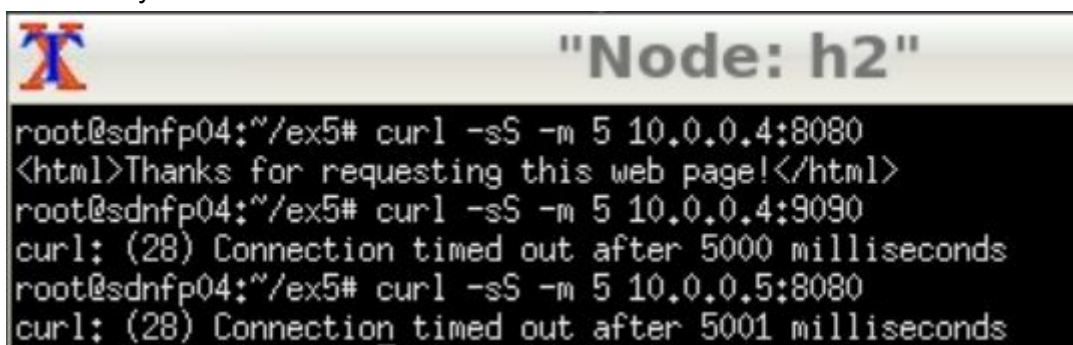
Output:

h1 can access all services running on **srv** and on **inet**



```
"Node: h1"
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
<html>Thanks for requesting this web page!</html>
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
Thanks for requesting this ssh service!
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.5:8080
<html>Thanks for requesting this web page!</html>
```

h2 can only access the web service on **srv**



```
"Node: h2"
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
<html>Thanks for requesting this web page!</html>
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5000 milliseconds
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.5:8080
curl: (28) Connection timed out after 5001 milliseconds
```

inet can only access the web service on srv

```

Node: inet
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
<html>Thanks for requesting this web page!</html>
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5001 milliseconds

```

monitor does not receive any packet

```

Node: h1
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
<html>Thanks for requesting this web page!</html>
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
Thanks for requesting this ssh service!
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.5:8080
<html>Thanks for requesting this web page!</html>
root@sdnfp04:~/ex5#

Node: h2
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
<html>Thanks for requesting this web page!</html>
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5000 milliseconds
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.5:8080
curl: (28) Connection timed out after 5000 milliseconds
root@sdnfp04:~/ex5#

Node: inet
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
<html>Thanks for requesting this web page!</html>
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5001 milliseconds
root@sdnfp04:~/ex5#

Node: mon
root@sdnfp04:~/ex5# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on mon-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
mon receives nothing

```

simple_monitor

The monitor policy is: if there is a packet sent from h2 or a packet that wants to access the ssh service on srv, we forward it to mon.

```
monitorPolicy = (match(srcip=ip_h2) | match(dstip=ip_srv,
dstport=9090, ethtype=packet.IPV4, protocol=packet.TCP_PROTO)) >>
(fwd(3))
```

```

Node: mon
root@sdnfp04:~/ex5# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on mon-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:56:14.524164 IP 10.0.0.2.57106 > 10.0.0.4.http-alt: Flags [S], seq 2558515113, win 42340, options [mss 1460,sackOK,TS val 237809437 ecr 0,nop,wscale 9], length 0
15:56:15.393154 IP 10.0.0.2.57106 > 10.0.0.4.http-alt: Flags [S], seq 2558515113, win 42340, options [mss 1460,sackOK,TS val 237810460 ecr 0,nop,wscale 9], length 0
15:56:17.342883 IP 10.0.0.2.57106 > 10.0.0.4.http-alt: Flags [S], seq 2558515113, win 42340, options [mss 1460,sackOK,TS val 237812476 ecr 0,nop,wscale 9], length 0
15:56:23.473014 IP 10.0.0.2.45366 > 10.0.0.4.9090: Flags [S], seq 3554230280, win 42340, options [mss 1460,sackOK,TS val 237817368 ecr 0,nop,wscale 9], length 0
15:56:24.438531 IP 10.0.0.2.45366 > 10.0.0.4.9090: Flags [S], seq 3554230280, win 42340, options [mss 1460,sackOK,TS val 237818372 ecr 0,nop,wscale 9], length 0
15:56:25.454353 IP 10.0.0.2.45366 > 10.0.0.4.9090: Flags [S], seq 3554230280, win 42340, options [mss 1460,sackOK,TS val 237820387 ecr 0,nop,wscale 9], length 0
15:56:34.787117 IP 10.0.0.2.53982 > 10.0.0.5.http-alt: Flags [S], seq 2027653468, win 42340, options [mss 1460,sackOK,TS val 3359757373 ecr 0,nop,wscale 9], length 0
15:56:35.801372 IP 10.0.0.2.53982 > 10.0.0.5.http-alt: Flags [S], seq 2027653468, win 42340, options [mss 1460,sackOK,TS val 3359758375 ecr 0,nop,wscale 9], length 0
15:56:37.802640 IP 10.0.0.2.53982 > 10.0.0.5.http-alt: Flags [S], seq 2027653468, win 42340, options [mss 1460,sackOK,TS val 3359760331 ecr 0,nop,wscale 9], length 0
15:56:38.579204 IP 10.0.0.1.53612 > 10.0.0.4.9090: Flags [S], seq 1165930277, win 42340, options [mss 1460,sackOK,TS val 1451341748 ecr 0,nop,wscale 9], length 0
15:56:44.594468 IP 10.0.0.1.53612 > 10.0.0.4.9090: Flags [S], seq 1165930277, win 42340, options [mss 1460,sackOK,TS val 1451342771 ecr 0,nop,wscale 9], length 0
15:56:56.395091 IP 10.0.0.1.53612 > 10.0.0.4.9090: Flags [S], seq 1165930277, win 42340, options [mss 1460,sackOK,TS val 1451344791 ecr 0,nop,wscale 9], length 0
15:57:25.003103 IP 10.0.0.5.37166 > 10.0.0.4.9090: Flags [S], seq 7503593328, win 42340, options [mss 1460,sackOK,TS val 1073575541 ecr 0,nop,wscale 9], length 0
15:57:27.018395 IP 10.0.0.5.37166 > 10.0.0.4.9090: Flags [S], seq 7503593328, win 42340, options [mss 1460,sackOK,TS val 1073576554 ecr 0,nop,wscale 9], length 0
15:57:29.019322 IP 10.0.0.5.37166 > 10.0.0.4.9090: Flags [S], seq 7503593328, win 42340, options [mss 1460,sackOK,TS val 1073576880 ecr 0,nop,wscale 9], length 0

Node: h2
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
curl: (28) Connection timed out after 5000 milliseconds
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5001 milliseconds
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.5:8080
curl: (28) Connection timed out after 5000 milliseconds
root@sdnfp04:~/ex5#

Node: h1
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
curl: (28) Connection timed out after 5001 milliseconds
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5001 milliseconds
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.5:8080
curl: (28) Connection timed out after 5001 milliseconds
root@sdnfp04:~/ex5#

Node: inet
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:8080
curl: (28) Connection timed out after 5000 milliseconds
root@sdnfp04:~/ex5# curl -sS -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5000 milliseconds
root@sdnfp04:~/ex5#

```

Task2

Start the pyretic module:

```
ex5$ pyretic.py -v low -m r0 qos
```

On inet and srv, run ./udpreceiver 4000, receiving datagrams on port 4000

On h1 and h2, run ./udpsender 10.0.0.X 4000 300, sending 300 datagrams to node inet:4000 or node srv:4000

The udp sender sends 10 datagrams per second and our policy `count_packets(0.05, ['srcip'])` calls its listeners every 50 milliseconds and provides each listener with a dictionary mapping source IP addresses to the cumulative number of packets containing that source IP address received by the bucket. The dictionary is shown as below:

```
{match: ('srcip', IPv4Network('10.0.0.1/32')): 256}
```

In the callback function, we perform the remainder calculation. If the remainder of the count divided by 10 is 0 for the sender h1, we change the policy to drop the packet sending from h1, otherwise we do not change the policy. Similarly, if the remainder of the count divided by 5 is 0 for the sender h2, the policy will be changed.

The screenshot shows four terminal windows with the following content:

- Node: h1**:


```
root@sdnfp04:~/ex5# ./udp_sender 10.0.0.5 4000 300
Sending 3000 datagrams to 10.0.0.5:4000 ... done
root@sdnfp04:~/ex5#
```
- Node: h2**:


```
root@sdnfp04:~/ex5# ./udp_sender 10.0.0.4 4000 300
Sending 3000 datagrams to 10.0.0.4:4000 ... done
root@sdnfp04:~/ex5#
```
- Node: inet**:


```
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
```

drop every 10th packet sent from h1
- Node: srv**:


```
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
```

drop every 5th packet sent from h2

Task3

monitor_firewall

start the pyretic module:

```
ex5$ pyretic.py -v low -m r0 monitor_firewall
```

start mininet and ssh & web services on srv & inet, and a xterm for all nodes:

```
ex5$ sudo ./mininet5.py mininet
```

```
> startservers mininet
```

```
> xterm h1 h2 srv inet mon
```

On h1, h2, and inet access web & ssh services through:

```
curl -sS -m 5 10.0.0.X:Y0Y0
```

On mon start monitoring with the packet capture tool `tcpdump` to print information about received packets

Output:

mon prints the traffic between h1 and web & ssh services on srv & inet

```

root@ndnf04:~/ex8 tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eno-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:15:38.383340 IP 10.0.0.1.37834 > 10.0.0.4.http-alt: Flags [S], seq 3661020573, win 42340, options [mss 1460,sackOK,TS val 1453066709,ecr 0,nop,wscale 9], length 0
15:15:38.387720 IP 10.0.0.4.http-alt > 10.0.0.1.37834: Flags [S], seq 3123642359, ack 3661020574, win 43440, options [mss 1460,sackOK,TS val 532758586,ecr 1453066709,nop,wscale 9], length 0
15:15:38.395413 IP 10.0.0.1.37834 > 10.0.0.4.http-alt: Flags [..], ack 1, win 83, options [nop,nop,TS val 1453066772,ecr 532758586], length 0
15:15:38.398164 IP 10.0.0.1.37834 > 10.0.0.4.http-alt: Flags [F..], seq 18, win 83, options [nop,nop,TS val 1453066772,ecr 532758586], length 0
15:15:38.415720 IP 10.0.0.4.http-alt > 10.0.0.1.37834: Flags [..], seq 78, win 85, options [nop,nop,TS val 532758587,ecr 1453066772], length 0
15:15:38.589774 IP 10.0.0.4.http-alt > 10.0.0.1.37834: Flags [F..], seq 118, ack 78, win 85, options [nop,nop,TS val 532758327,ecr 1453066772], length 17: HTTP/1.0 200 OK
15:15:39.631319 IP 10.0.0.4.http-alt > 10.0.0.1.37834: Flags [F..], seq 18:212, ack 78, win 85, options [nop,nop,TS val 532758327,ecr 1453066772], length 194: HTTP
15:15:39.638164 IP 10.0.0.1.37834 > 10.0.0.4.http-alt: Flags [..], ack 18, win 83, options [nop,nop,TS val 1453067382,ecr 532758327], length 0
15:15:39.680164 IP 10.0.0.1.37834 > 10.0.0.4.http-alt: Flags [F..], seq 78, ack 215, win 83, options [nop,nop,TS val 1453067428,ecr 532758327], length 0
15:15:39.731816 IP 10.0.0.4.http-alt > 10.0.0.1.37834: Flags [..], seq 79, win 85, options [nop,nop,TS val 532759403,ecr 1453067428], length 0
15:15:38.155332 IP 10.0.0.1.52688 > 10.0.0.4.9090: Flags [S], seq 1593044905, win 42340, options [mss 1460,sackOK,TS val 1453085900,ecr 0,nop,wscale 9], length 0
15:15:38.156131 IP 10.0.0.4.9090 > 10.0.0.1.52688: Flags [S..], seq 393642350, ack 1593044906, win 43440, options [mss 1460,sackOK,TS val 532777877,ecr 1453085900,nop,wscale 9], length 0
15:15:38.225338 IP 10.0.0.1.52688 > 10.0.0.4.9090: Flags [..], ack 1, win 83, options [nop,nop,TS val 1453085938,ecr 532777877], length 0
15:15:38.236586 IP 10.0.0.4.9090 > 10.0.0.1.52688: Flags [F..], seq 18, win 83, options [nop,nop,TS val 1453085937,ecr 532777877], length 77
15:15:38.311720 IP 10.0.0.4.9090 > 10.0.0.1.52688: Flags [..], seq 78, win 85, options [nop,nop,TS val 532777991,ecr 1453085937], length 0
15:15:38.804417 IP 10.0.0.4.9090 > 10.0.0.1.52688: Flags [F..], seq 118, ack 78, win 85, options [nop,nop,TS val 532778492,ecr 1453085937], length 17
15:15:38.861764 IP 10.0.0.4.9090 > 10.0.0.1.52688: Flags [F..], seq 18:202, ack 78, win 85, options [nop,nop,TS val 532778492,ecr 1453085937], length 184
15:15:38.868164 IP 10.0.0.1.52688 > 10.0.0.4.9090: Flags [..], ack 18, win 83, options [nop,nop,TS val 1453085937,ecr 532778492], length 0
15:15:38.885396 IP 10.0.0.1.52688 > 10.0.0.4.9090: Flags [F..], seq 78, ack 205, win 83, options [nop,nop,TS val 1453086540,ecr 532778492], length 0
15:15:38.983361 IP 10.0.0.4.9090 > 10.0.0.1.52688: Flags [..], ack 79, win 85, options [nop,nop,TS val 532778638,ecr 1453086540], length 0
15:15:04.419435 IP 10.0.0.1.51174 > 10.0.0.5.http-alt: Flags [S], seq 859488486, win 42340, options [mss 1460,sackOK,TS val 3688862814,ecr 0,nop,wscale 9], length 0
15:15:04.430575 IP 10.0.0.5.http-alt > 10.0.0.1.51174: Flags [S..], seq 4293131183, ack 859488486, win 43440, options [mss 1460,sackOK,TS val 2135721468,ecr 3688862814,nop,wscale 9], length 0
15:15:04.437580 IP 10.0.0.1.51174 > 10.0.0.5.http-alt: Flags [..], ack 1, win 83, options [nop,nop,TS val 2135721468,ecr 2135721468], length 0
15:15:04.475930 IP 10.0.0.1.51174 > 10.0.0.5.http-alt: Flags [F..], seq 18, win 83, options [nop,nop,TS val 3688862853,ecr 2135721468], length 77: HTTP: GET / HTTP/1.1
15:15:04.497375 IP 10.0.0.5.http-alt > 10.0.0.1.51174: Flags [..], seq 78, win 85, options [nop,nop,TS val 2135721525,ecr 3688862853], length 0
15:15:05.013453 IP 10.0.0.5.http-alt > 10.0.0.1.51174: Flags [F..], seq 118, ack 78, win 85, options [nop,nop,TS val 2135722026,ecr 3688862853], length 17: HTTP: HTTP/1.0 200 OK
15:15:05.026560 IP 10.0.0.5.http-alt > 10.0.0.1.51174: Flags [F..], seq 18:212, ack 78, win 85, options [nop,nop,TS val 2135722026,ecr 3688862853], length 194: HTTP
15:15:05.026560 IP 10.0.0.1.51174 > 10.0.0.5.http-alt: Flags [..], ack 18, win 83, options [nop,nop,TS val 3688863440,ecr 2135722026], length 0
15:15:05.062429 IP 10.0.0.1.51174 > 10.0.0.5.http-alt: Flags [F..], seq 78, ack 215, win 83, options [nop,nop,TS val 3688863444,ecr 2135722026], length 0
15:15:05.107367 IP 10.0.0.5.http-alt > 10.0.0.1.51174: Flags [..], seq 79, win 85, options [nop,nop,TS val 2135722113,ecr 3688863444], length 0

```

mon only prints the traffic between h2 and the web service on srv

```
root@dnf04:/# nslookup www.google.com
Server: dnsmasq[98]
Address: 10.0.0.1
Name: google.com
Address: 74.125.236.100

```

```
root@dnf04:/# ./tcpdump -n -v -c 1000 -w tcp.pcap &
tcpdump: verbose output suppressed, use -v or --vv for full protocol decode
listening on mon-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:19:00.302976 IP 10.0.0.2.57194 >> 10.0.0.4.http-alt: Flags [S], seq 12505933660, win 42340, options [mss 1460,sackOK,T.S val 233174771 ecr 0,nop,wscale 9], length 0
16:19:00.312306 IP 10.0.0.4.http-alt >> 10.0.0.2.57194: Flags [S], seq 2384817121, ack 12505933661, win 43440, options [mss 1460,sackOK,T.S val 1089463942 ecr 233174771,nop,wscale 9], length 0
16:19:00.418234 IP 10.0.0.2.57194 >> 10.0.0.4.http-alt: Flags [I], ack 1, win 83, options [nop,nop,T.S val 233174878 ecr 1089463942], length 0
16:19:00.419870 IP 10.0.0.2.57194 >> 10.0.0.4.http-alt: Flags [P], seq 178, ack 1, win 83, options [nop,nop,T.S val 233174878 ecr 1089463942], length 77: HTTP: GET / HTTP/1.1
16:19:00.440452 IP 10.0.0.4.http-alt >> 10.0.0.2.57194: Flags [.], ack 78, win 85, options [nop,nop,T.S val 1089464057 ecr 233174878], length 0
16:19:00.395429 IP 10.0.0.4.http-alt >> 10.0.0.2.57194: Flags [F], seq 118, ack 78, win 85, options [nop,nop,T.S val 1089464557 ecr 233174878], length 17: HTTP: HTTP/1.0 200 OK
16:19:00.418234 IP 10.0.0.2.57194 >> 10.0.0.4.http-alt: Flags [F], seq 18120, ack 78, win 85, options [nop,nop,T.S val 1089464558 ecr 233174878], length 194: HTTP: 
16:19:00.407349 IP 10.0.0.2.57194 >> 10.0.0.4.http-alt: Flags [R], seq 18, win 83, options [nop,nop,T.S val 23317474 ecr 1089464557], length 0
16:19:01.029705 IP 10.0.0.2.57194 >> 10.0.0.4.http-alt: Flags [F], seq 78, ack 213, win 83, options [nop,nop,T.S val 233175526 ecr 1089464558], length 0
16:19:01.071837 IP 10.0.0.4.http-alt >> 10.0.0.2.57194: Flags [.], ack 79, win 85, options [nop,nop,T.S val 1089464662 ecr 233175526], length 0
```

```
root@dnf04:/# curl -s http://www.google.com/
<html></html>
```

```
root@dnf04:/# ./nslookup www.google.com
Server: dnsmasq[98]
Address: 10.0.0.1
Name: google.com
Address: 74.125.236.100
```

mon only prints the traffic between inet and web service on srv

```

root@dnfp04:~# ss -t sntdp
State     Recv-Q     Send-Q     Local Address:Port  Peer Address:Port
LISTENING  on m0n0eth0, link-type ENUMB (Ethernet), capture size 262144 bytes
16:19:59.130973 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [S], seq 566913601, win 42340, options [mss 1460,sackOK,TS val 1075028755 ecr 0,nop,wscale 9], length 0
16:19:59.141849 IP 10.0.0.4.http-alt > 10.0.0.5.40368: Flags [S.], seq 3158997647, ack 566913602, win 43440, options [mss 1460,sackOK,TS val 180548758 ecr 1075028755,nop,wscale 9], length 0
16:19:59.187985 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [L], ack 1, win 83, options [nop,nop,TS val 1075028789 ecr 180548758], length 0
16:19:59.187844 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [P.], seq 1:78, ack 1, win 83, options [nop,nop,TS val 1075028789 ecr 180548758], length 77: HTTP: GET / HTTP/1.1
16:19:59.251726 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [L], ack 78, win 85, options [nop,nop,TS val 180548916 ecr 1075028789], length 0
16:19:59.274446 IP 10.0.0.4.http-alt > 10.0.0.5.40368: Flags [P.], seq 1:15, ack 78, win 65, options [nop,nop,TS val 180549316 ecr 1075028789], length 17: HTTP: HTTP/1.0 200 OK
16:19:59.731457 IP 10.0.0.4.http-alt > 10.0.0.5.40368: Flags [FP.], seq 82:212, ack 78, win 85, options [nop,nop,TS val 180549316 ecr 1075028789], length 194: HTTP
16:19:59.775730 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [L], ack 18, win 83, options [nop,nop,TS val 1075029375 ecr 180549316], length 0
16:19:59.775382 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [F.], seq 78, ack 213, win 83, options [nop,nop,TS val 1075029380 ecr 180549316], length 0
16:19:59.799481 IP 10.0.0.4.http-alt > 10.0.0.5.40368: Flags [L.], ack 79, win 85, options [nop,nop,TS val 180549403 ecr 1075029380], length 0

```

```

root@dnfp04:~# ss -t sntdp
State     Recv-Q     Send-Q     Local Address:Port  Peer Address:Port
LISTENING  on m0n0eth0, link-type ENUMB (Ethernet), capture size 262144 bytes
16:19:59.130973 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [S], seq 566913601, win 42340, options [mss 1460,sackOK,TS val 1075028755 ecr 0,nop,wscale 9], length 0
16:19:59.141849 IP 10.0.0.4.http-alt > 10.0.0.5.40368: Flags [S.], seq 3158997647, ack 566913602, win 43440, options [mss 1460,sackOK,TS val 180548758 ecr 1075028755,nop,wscale 9], length 0
16:19:59.187985 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [L], ack 1, win 83, options [nop,nop,TS val 1075028789 ecr 180548758], length 0
16:19:59.187844 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [P.], seq 1:78, ack 1, win 83, options [nop,nop,TS val 1075028789 ecr 180548758], length 77: HTTP: GET / HTTP/1.1
16:19:59.251726 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [L], ack 78, win 85, options [nop,nop,TS val 180548916 ecr 1075028789], length 0
16:19:59.274446 IP 10.0.0.4.http-alt > 10.0.0.5.40368: Flags [P.], seq 1:15, ack 78, win 65, options [nop,nop,TS val 180549316 ecr 1075028789], length 17: HTTP: HTTP/1.0 200 OK
16:19:59.731457 IP 10.0.0.4.http-alt > 10.0.0.5.40368: Flags [FP.], seq 82:212, ack 78, win 85, options [nop,nop,TS val 180549316 ecr 1075028789], length 194: HTTP
16:19:59.775730 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [L], ack 18, win 83, options [nop,nop,TS val 1075029375 ecr 180549316], length 0
16:19:59.775382 IP 10.0.0.5.40368 > 10.0.0.4.http-alt: Flags [F.], seq 78, ack 213, win 83, options [nop,nop,TS val 1075029380 ecr 180549316], length 0
16:19:59.799481 IP 10.0.0.4.http-alt > 10.0.0.5.40368: Flags [L.], ack 79, win 85, options [nop,nop,TS val 180549403 ecr 1075029380], length 0

```

```

root@dnfp04:~# curl -s -m 5 10.0.0.4:8080
<html>Thanks for requesting this web page</html>
root@dnfp04:~# curl -s -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5001 milliseconds
root@dnfp04:~# curl -s -m 5 10.0.0.4:9090
curl: (28) Connection timed out after 5001 milliseconds

```

monitor firewall qos

The basic logic is: if there is an UDP datagram, then we perform the QoS policy, otherwise we perform the firewall and monitoring policy:

```
if (match(protocol=packet.UDP PROTO), qos(), monitor firewall)
```

Start the pyretic module:

```
ex5$ pyretic.py -v low -m r0 monitor firewall qos
```



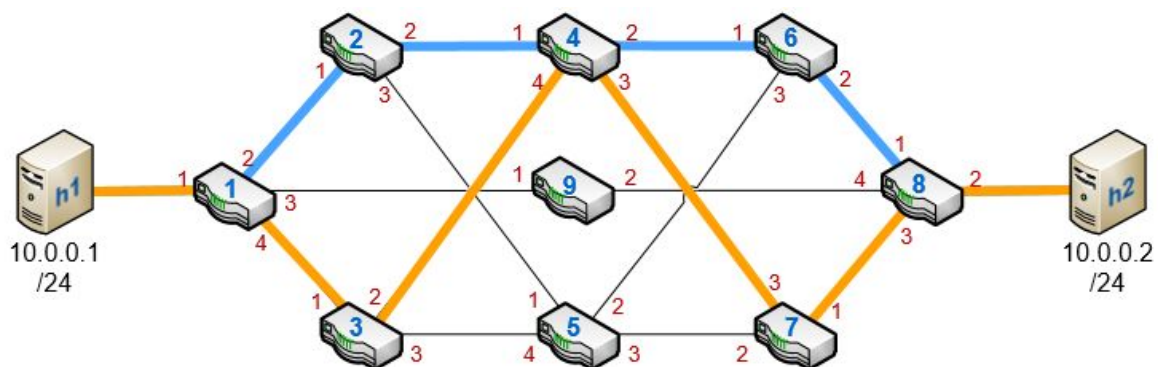
```
student@sdnfp04:~/ex5$ pyretic.py -v low -m r0 monitor_firewall_qos
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
Connected to pyretic frontend.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
```

On inet and srv, run `./udpreceiver 4000`, receiving datagrams on port 4000

On h1 and h2, run `./udpsender 10.0.0.X 4000 300`, sending 300 datagrams to node inet:4000 or node srv:4000

The monitor and firewall part is the same as above.

Task4



blue.py

```
(match(switch=1) | match(switch=2) | match(switch=4) |
match(switch=6) | match(switch=8)) & match(inport=1) >> fwd(2)
```

```
orange.py
```

```
(match(switch=1,inport=1) >> fwd(4)) +  
( (match(switch=3,inport=1) | match(switch=8,inport=3)) >> fwd(2))  
+  
(match(switch=4,inport=4) >> fwd(3)) +  
(match(switch=7,inport=3) >> fwd(1))
```

Forwarding abstraction of Pyretic is not considered purely behavioral. In this task, we have to specify the port of a switch in each forwarding path. It is desired to specify only the switch of the designated route without specifying the exact port or lower layer connectivity.

It would be desired to only specify switch on routing path and leave compiler to match the topology with ports. For example, we, the user, only want to declare two flows, with special function in pyretic `CREATE_FLOW(switch=1, 2, 4, 6, 8)` and `CREATE_FLOW(switch=1, 3, 4, 7, 8)` and don't want to take care of the detail connection.

For pyretic programming, the space complexity is $O(N*M)$, where N is the number of total switches and M is the number of the port for each switch, because at worst case, we need to specify every in-out-port for every switches.

Space complexity of `blue.py` is lower than that of `orange.py` because there is only one action (`fwd(2)`) in the former case. However, since the inport and outport for each switch is different in the orange route, we have to define different match conditions and actions for different switches, so the policy is more complex and requires more memory space to store.