

گزارش آزمایش 5

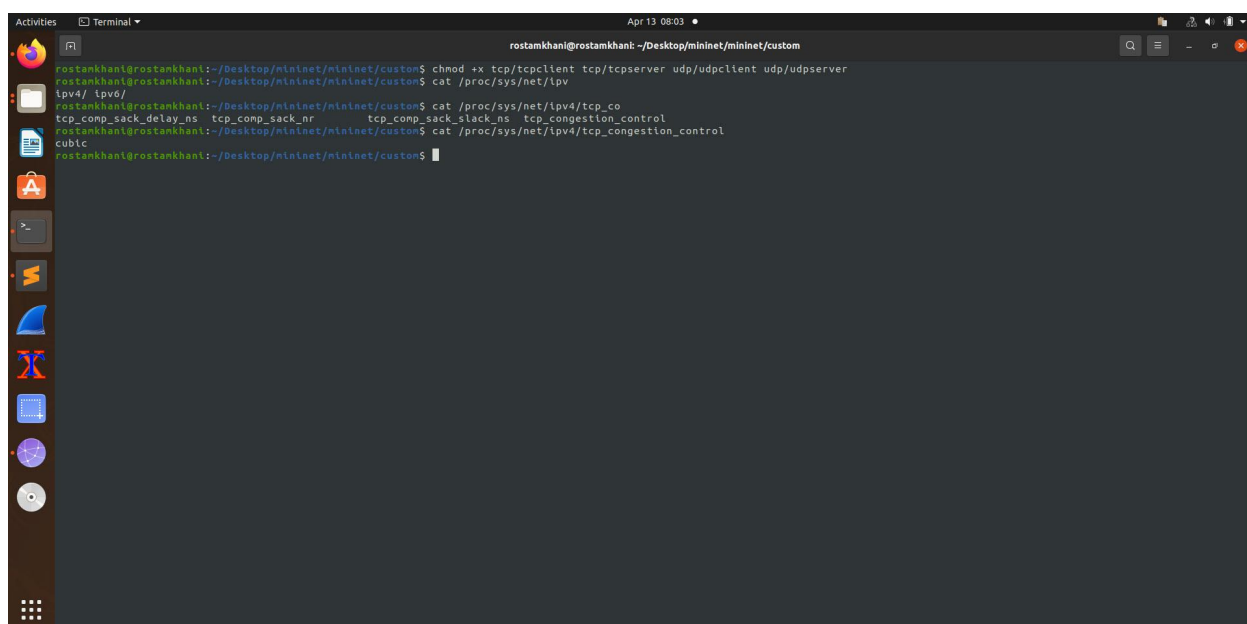
آرمین غلام پور - 97521414

محمد مصطفی رستم خانی - 97521306

سجاد رضانی

الف) ابتدا فایل ها را در مکان مناسب قرار داده و به آنها حالت اجرایی می دهیم:

```
chmod +x tcp/tcpclient tcp/tcpserver udp/udpclient udp/udpserver
```



```
rostandkhani@rostandkhani: ~/Desktop/mininet/mininet/custom
rostandkhani@rostandkhani:~/Desktop/mininet/mininet/custom$ chmod +x tcp/tcpclient tcp/tcpserver udp/udpclient udp/udpserver
rostandkhani@rostandkhani:~/Desktop/mininet/mininet/custom$ cat /proc/sys/net/ipv4/tcp_
tcp_
rostandkhani@rostandkhani:~/Desktop/mininet/mininet/custom$ cat /proc/sys/net/ipv4/tcp_co
tcp_co
rostandkhani@rostandkhani:~/Desktop/mininet/mininet/custom$ cat /proc/sys/net/ipv4/tcp_con
tcp_con
rostandkhani@rostandkhani:~/Desktop/mininet/mininet/custom$ cat /proc/sys/net/ipv4/tcp_con
cubic
```

سپس مکانیزم کنترل ازدحام در ماشین مجازی را بررسی می کنیم. می توانیم با تایپ دستور زیر بررسی کنیم که چه مکانیزمی مورد استفاده است:

```
cat /proc/sys/netip4/tcp_congestion_control
```

همانطور که در بالا مشاهده می کنیم، مکانیزم کنترل ازدحام در ابتدا tcp cubic بوده است.

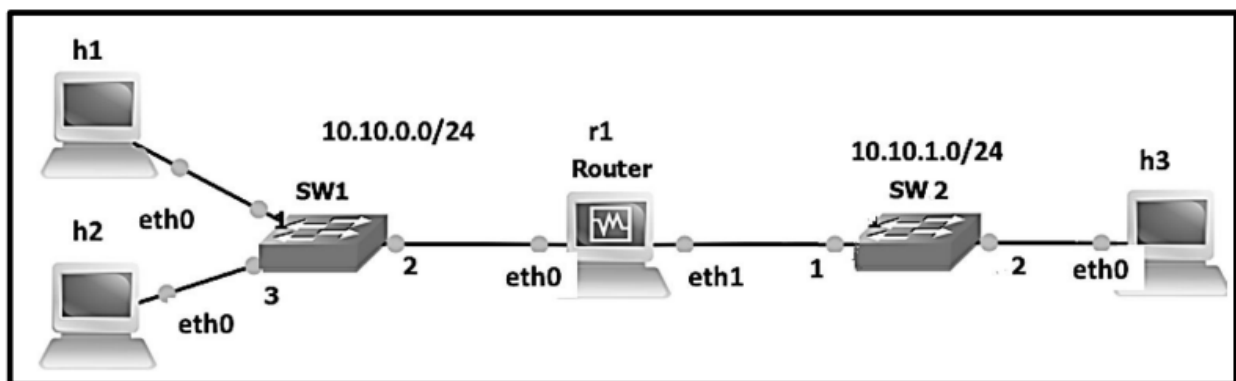
در این آزمایش، ما TCP را ملزم می کنیم که از الگوریتم کنترل ازدحام reno استفاده کند. در صورتی که پیشاپیش، الگوریتم مورد نظر از نوع reno نباشد، ما می توانیم با تایپ دستور زیر در پنجره ترمینال، آن را به reno تغییر دهیم (البته تا reboot بعدی):

```
sudo bash -c 'echo reno > /proc/sys/net/ipv4/tcp_congestion_control'
```

```
Activities Terminal Apr 13 08:04
rostamkhani@rostamkhani: ~/Desktop/mininet/mininet/custom
rostamkhani@rostamkhani:~/Desktop/mininet/mininet/custom$ chnod +x tcp/tcpclient tcp/tcpserver udp/udpclient udp/udpserver
rostamkhani@rostamkhani:~/Desktop/mininet/mininet/custom$ cat /proc/sys/net/ipv4/
tcp/ 1
udp/ 1
rosta...@rosta...:~/Desktop/mininet/mininet/custom$ cat /proc/sys/net/ipv4/tcp_co
tcp_comp_sack_delay_ns tcp_comp_sack_nr tcp_comp_sack_slack_ns tcp_congestion_control
rosta...@rosta...:~/Desktop/mininet/mininet/custom$ cat /proc/sys/net/ipv4/tcp_congestion_control
cubic
rosta...@rosta...:~/Desktop/mininet/mininet/custom$ sudo bash -c 'echo reno > /proc/sys/net/ipv4/tcp_congestion_control'
[sudo] password for rosta...:
rosta...@rosta...:~/Desktop/mininet/mininet/custom$ cat /proc/sys/net/ipv4/tcp_congestion_control
reno
rosta...@rosta...:~/Desktop/mininet/mininet/custom$
```

همانطور که مشاهده می کنیم بعد از اجرای دستور بالا، از cubic به reno تغییر یافته ایم.

(ب) سپس فایل lab5_network.py را به گونه ای تغییر میدهم که توپولوژی زیر را بسازد:



تغییرات انجام شده در کد در زیر نشان داده شده اند:

```

52 global net
53 global hosts
54
55 net = Mininet(intf=TCIntf)
56
57 info('\n** Adding Controller\n')
58 net.addController('c0')
59
60 info('\n** Adding Hosts\n')
61 h1 = net.addHost('h1', ip='10.10.0.1/24', hostname='h1', private
62 h2 = net.addHost('h2', ip='10.10.0.2/24', hostname='h2', private
63 h3 = net.addHost('h3', ip='10.10.1.3/24', hostname='h3', private
64 r1 = net.addHost('r1', ip='10.10.1.10/24', hostname='r1', private
65 # Space to add any commands for configuring the IP addresses
66 #
67 #
68 #
69 #
70
71 info('\n** Adding Switches\n')
72 # Adding switches to the network
73 sw1 = net.addSwitch('sw1')
74 sw2 = net.addSwitch('sw2')
75
76 info('\n** Creating Links\n')
77 link_h1sw1 = net.addLink(h1, sw1)
78 link_h2sw1 = net.addLink(h2, sw1)
79 link_h3sw2 = net.addLink(h3, sw2)
80 link_r1sw1 = net.addLink(r1, sw1, intfName1='r1-eth0')
81 link_r1sw2 = net.addLink(r1, sw2, intfName1='r1-eth1')
82
83 info('\n** Modifying Link Parameters\n')
84 """
85 Default parameters for links:
86 bw = None,
87 delay = None,
88 jitter = None,
89 loss = None,

```

```

91 disable_gro = True,
92 speedup = 0,
93 use_hfsc = False,
94 use_tbf = False,
95 latency_ms = None,
96 enable_ecn = False,
97 enable_red = False,
98 max_queue_size = None
99
100 link_r1sw2.intf1.config(bw=10, enable_red=True, enable_ecn=True)
101
102 net.start()
103
104 info('*** Configuring hosts\n')
105 r1.cmd('ifconfig r1-eth0 10.10.0.10 netmask 255.255.255.0')
106 r1.cmd('ifconfig r1-eth1 10.10.1.10 netmask 255.255.255.0')
107 r1.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')
108 # r1.cmd('ip route add default via 10.0.1.3')
109 # r1.cmd('ip route add default via 10.0.1.3')
110 # r1.cmd('ip route add default via 10.0.1.3')
111
112 h1.cmd('ip route add default via 10.10.0.10')
113 h2.cmd('ip route add default via 10.10.0.10')
114 h3.cmd('ip route add default via 10.10.1.10')
115 # Space to add commands for configuring routing tables and default
116 #
117 #
118 #
119 #
120
121 info('*** Executing custom commands\n')
122 output = net.nameToNode.keys
123 #Enable Xterm window for every host
124 info('*** Enabling xterm for hosts only\n')
125 # We check if the display is available
126 hosts = [h1, h2, h3, r1]
127 if 'DISPLAY' not in os.environ:
128     error("Error starting terms: Cannot connect to display\n")

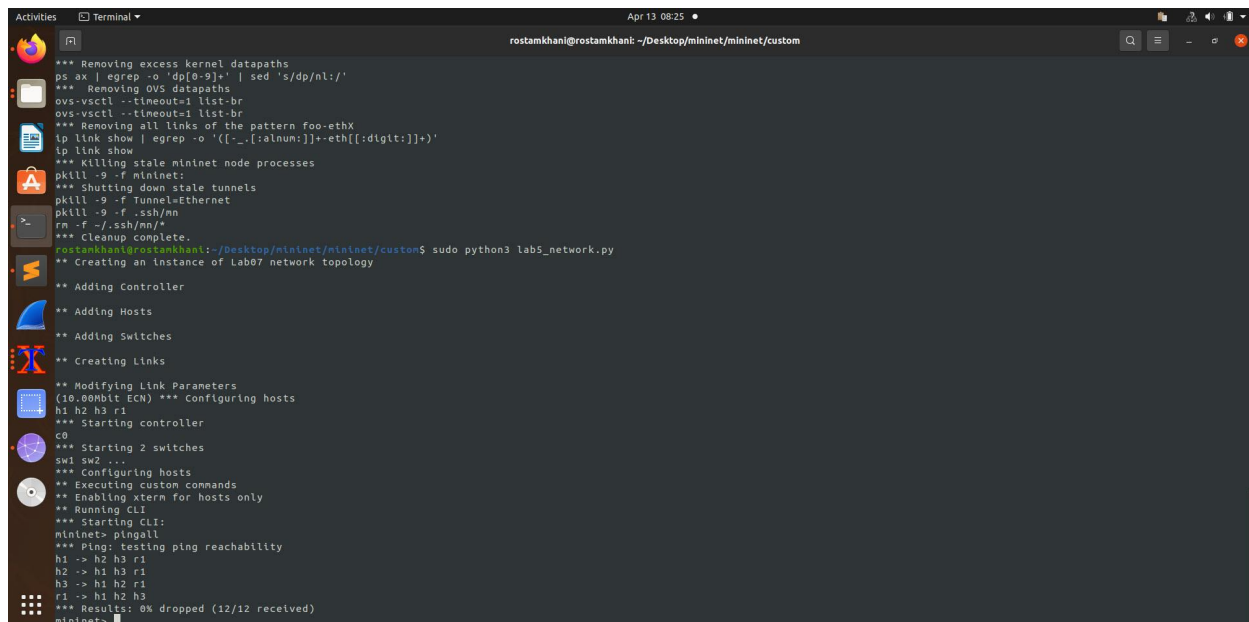
```

کد اصلی در فایل ضمیمه شده است.

- سابت ماشین های h1، h2 و روتر r1 دارای آدرس 10.10.0.0/24 است. آدرس های h1، h2 و روتر هم به ترتیب به صورت 10.10.0.1، 10.10.0.2 و 10.10.0.10 می باشد.
- سابت ماشین h3 و روتر دارای آدرس 10.10.1.0/24 است. آدرس های h3 و روتر هم به ترتیب، 10.10.1.3 و 10.10.1.10 می باشد.

یک ترمینال در ماشین خود باز کنید و اسکریپت lab5_network.py را اجرا نمایید. پس از اجرای توپولوژی، با استفاده از دستور pingall، پیکربندی خود را آزمایش نمایید.

سپس برای تست درستی شبکه ساخته شده، pingall می‌کنیم.



```
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]:' | sed 's/dp/nl:/'
*** Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([_[:alnum:]]+eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/nn
rm -f ~/.ssh/nn/*
*** Cleanup complete.
rostandkhan@rostandkhan: ~/Desktop/mininet/mininet/custom$ sudo python3 lab5_network.py
** Creating an instance of Lab07 network topology
** Adding Controller
** Adding Hosts
** Adding Switches
** Creating Links
** Modifying Link Parameters
(10.00Mbit ECN) *** Configuring hosts
h1 h2 h3 r1
*** Starting controller
c0
*** Starting 2 switches
sw1 sw2 ...
*** Configuring hosts
** Executing custom commands
** Enabling xterm for hosts only
** Running CLI
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 r1
h2 -> h1 h2 r1
h3 -> h1 h2 r1
r1 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

مشاهده می‌کنیم که با pingall همه device ها می‌توانند یکدیگر را ping کنند پس پیکره بندی درست است.