Project 3 Switches and Bridges Suraj Sureshkumar(ss7495)

** I installed a virtual box and added the VM image to the virtual box.
I am running this on a windows machine through the help of a virtual box.
Used the nano editor which is available in the mininet vm to code the python script.
I wrote the program in nano editor using python and ran it using the root access.
I have done a mix of the program and mininet vm terminal. So you might not get the exact Output if you just run the program.**

Program Code:

```
 GNU nano 2.5.3                File: proj3.py                      Modified

from mininet.cli import CLI
from mininet.link import TCLink, Link
from mininet.net import Mininet
import time

if __name__ == '__main__':
        net = Mininet(link=TCLink)

        h1 = net.addHost('h1', ip='50.0.0.1', mac='00:00:00:00:01:00')
        h2 = net.addHost('h2', ip='50.0.0.2', mac='00:00:00:00:02:00')
        h3 = net.addHost('h3', ip='50.0.0.3', mac='00:00:00:00:03:00')
        h4 = net.addHost('h4', ip='50.0.0.4', mac='00:00:00:00:04:00')
        bridge = net.addHost('bridge', ip = '50.0.0.8', mac='00:00:00:00:08:00')

        Link(h1, bridge)
        Link(h2, bridge)
        Link(h3, bridge)
        Link(h4, bridge)

        net.build()

        bridge.cmd('ifconfig bridge-eth0 0')
        bridge.cmd('ifconfig bridge-eth1 0')
        bridge.cmd('ifconfig bridge-eth2 0')
        bridge.cmd('ifconfig bridge-eth3 0')

        bridge.cmd('brctl addbr br0')


        bridge.cmd('brctl addif br0 bridge-eth0')
        bridge.cmd('brctl addif br0 bridge-eth1')
        bridge.cmd('brctl addif br0 bridge-eth2')

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  ^Y Prev Page
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Linter ^_ Go To Line ^V Next Page
```

```
        bridge.cmd('brctl addif br0 bridge-eth3')

        bridge.cmd('ifconfig br0 up')

        print(bridge.cmd('brctl showmacs br0'))
        net.pingAll()

        bridge.cmd('bridge monitor fdb >> bridge_logs.txt &')

        h1.cmd('tcpdump -n -e -i h1-eth0 >> h1logs.txt &')
        h2.cmd('tcpdump -n -e -i h2-eth1 >> h2logs.txt &')
        h3.cmd('tcpdump -n -e -i h3-eth2 >> h1logs.txt &')
        h4.cmd('tcpdump -n -e -i h4-eth3 >> h1logs.txt &')

        h1.cmd('iperf -s > h1.log &')
        h2.cmd('iperf -s > h2.log &')
```

```
        h3.cmd('iperf -c 50.0.0.1 -i 10 -t 10 > h1_h3_result.txt &')
        h4.cmd('iperf -c 50.0.0.2 -i 10 -t 10 > h2_h4_result.txt &')

        net.waitConnected()

        CLI(net)

        net.stop()
```

Pre Task 1:

Created 4 hosts named h1,h2,h3,h4 and another host name bridge.
Assigned each host an IP address.
Linked the h1,h2,h3,h4 to the bridge

Task 1: Configure the bridge

Step1: : Flush IP addresses from the four interfaces of the bridge using:

```
bridge.cmd("ifconfig bridge-eth0 0")
bridge.cmd("ifconfig bridge-eth1 0")
bridge.cmd("ifconfig bridge-eth2 0")
bridge.cmd("ifconfig bridge-eth3 0")
```

Step2: : Created a new interface called br0:

```
bridge.cmd("brctl addbr br0")
```

Step3: : Added all the 4 hosts to br0:

```
bridge.cmd('brctl addif br0 bridge-eth1')
bridge.cmd('brctl addif br0 bridge-eth2')
bridge.cmd('brctl addif br0 bridge-eth3')
bridge.cmd('brctl addif br0 bridge-eth4')
```

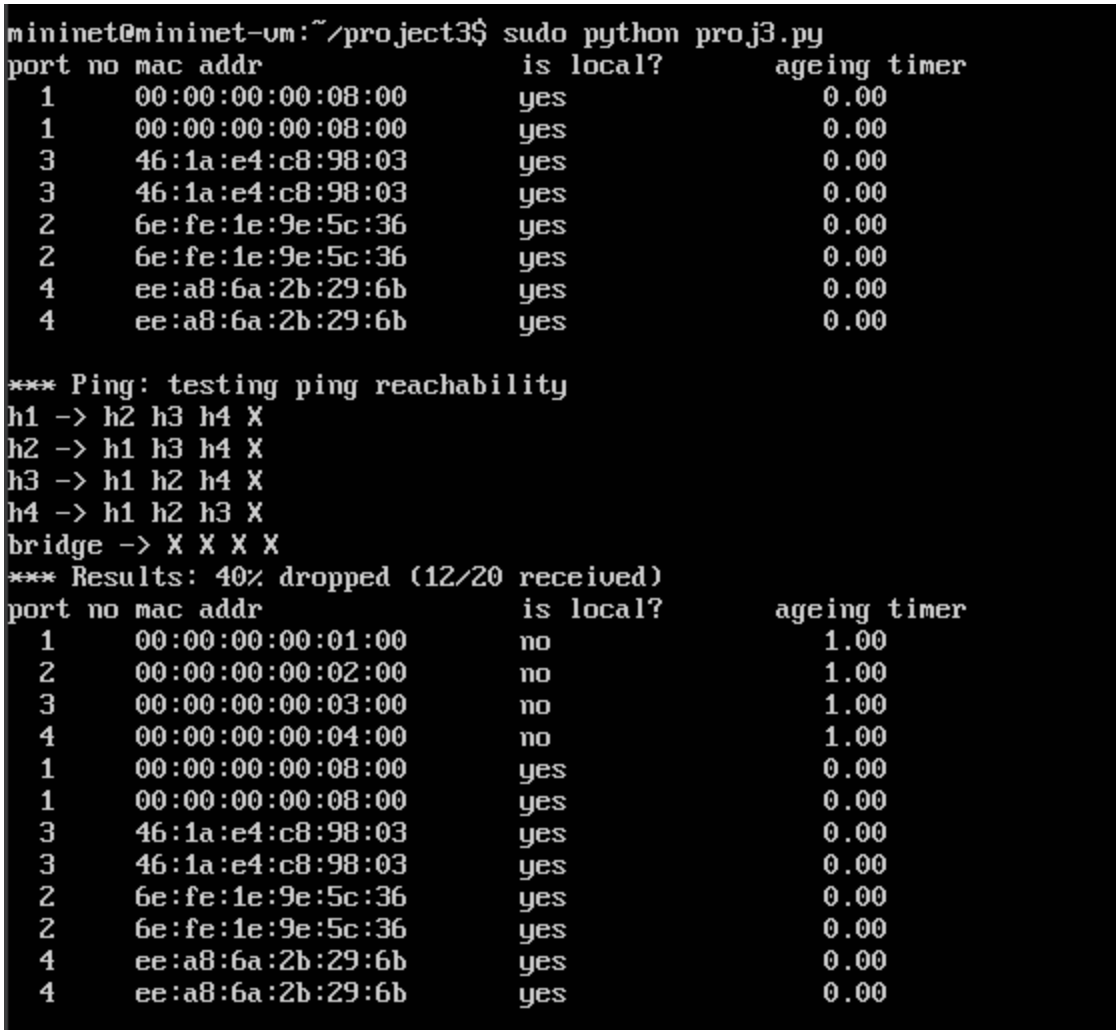
Step4: : Brought up all the interfaces, particularly the br0 since all the hosts are connected to br0.

```
bridge.cmd("ifconfig br0 up")
```

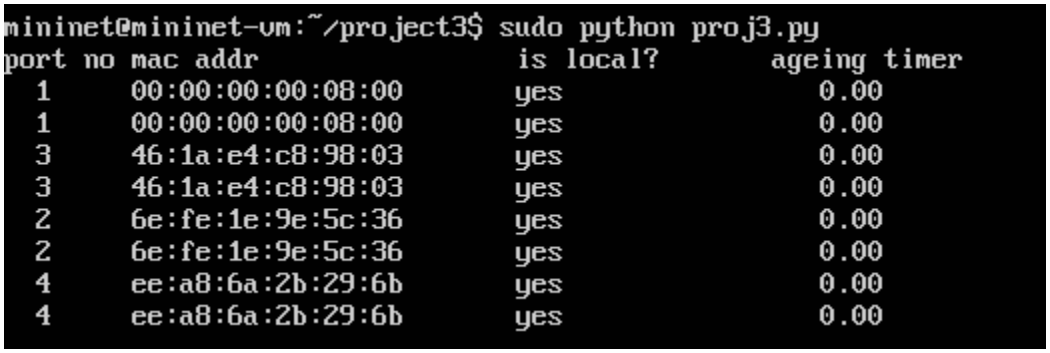
Step 5 and 6: :

Listing all the MAC addresses and then verifying that each host can reach others through the bridge by doing a pingall and the output is as below when the program is run:

Need to run the project with root access like



I will break the above image to chunks and explain:



I ran the program using root access and added a `print(bridge.cmd('brctl showmacs br0'))` to show the initial table before doing a pingall to check the connectivity.

```
*** Ping: testing ping reachability
h1 -> h2 h3 h4 X
h2 -> h1 h3 h4 X
h3 -> h1 h2 h4 X
h4 -> h1 h2 h3 X
bridge -> X X X X
*** Results: 40% dropped (12/20 received)
```

Shows us that all the hosts are connected and reachable.

```
port no mac addr                is local?       ageing timer
  1     00:00:00:00:01:00       no                 1.00
  2     00:00:00:00:02:00       no                 1.00
  3     00:00:00:00:03:00       no                 1.00
  4     00:00:00:00:04:00       no                 1.00
  1     00:00:00:00:08:00       yes                0.00
  1     00:00:00:00:08:00       yes                0.00
  3     46:1a:e4:c8:98:03       yes                0.00
  3     46:1a:e4:c8:98:03       yes                0.00
  2     6e:fe:1e:9e:5c:36       yes                0.00
  2     6e:fe:1e:9e:5c:36       yes                0.00
  4     ee:a8:6a:2b:29:6b       yes                0.00
  4     ee:a8:6a:2b:29:6b       yes                0.00
```

This shows us a change in the forwarding table as the bridge learns all the host addresses and adds it to the table.

Task 2: Bridge learns MAC addresses

Step1:

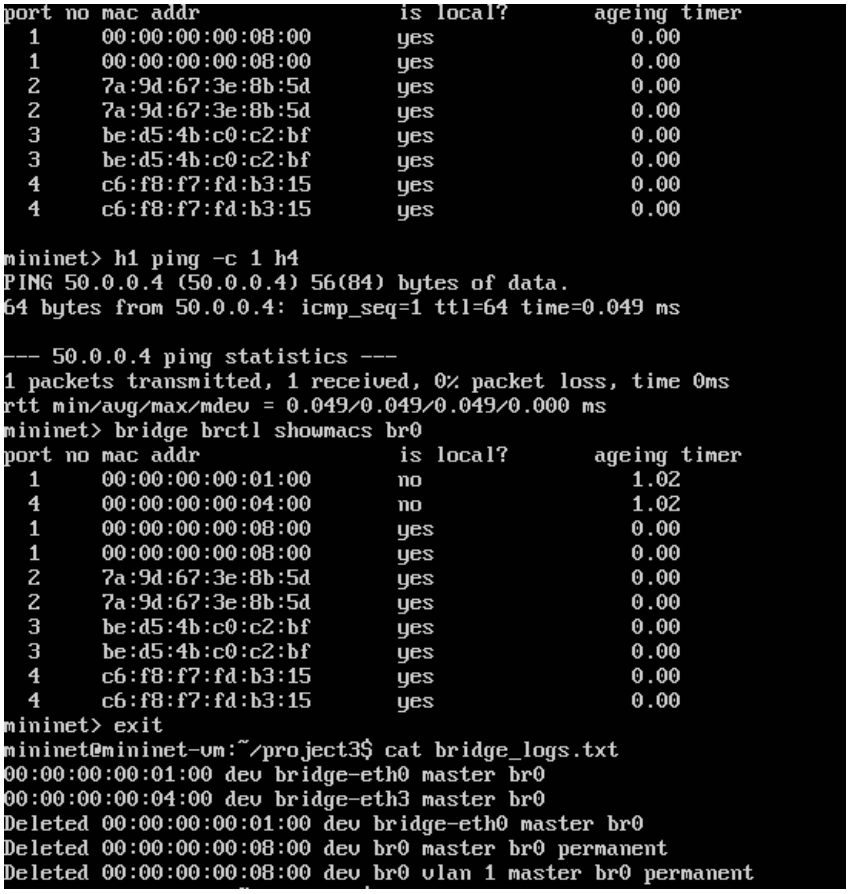
If I run the showmacs command again the bridge shows me only the local address.

```
mininet> bridge brctl showmacs br0
port no mac addr                is local?       ageing timer
  1     00:00:00:00:08:00       yes                0.00
  1     00:00:00:00:08:00       yes                0.00
  3     46:1a:e4:c8:98:03       yes                0.00
  3     46:1a:e4:c8:98:03       yes                0.00
  2     6e:fe:1e:9e:5c:36       yes                0.00
  2     6e:fe:1e:9e:5c:36       yes                0.00
  4     ee:a8:6a:2b:29:6b       yes                0.00
  4     ee:a8:6a:2b:29:6b       yes                0.00
```

Step2:

So the code for the logs has been added as

```
bridge.cmd('bridge monitor fdb >> bridge_logs.txt &')
```



```
port no mac addr                is local?       ageing timer
  1     00:00:00:00:08:00       yes                0.00
  1     00:00:00:00:08:00       yes                0.00
  2     7a:9d:67:3e:8b:5d       yes                0.00
  2     7a:9d:67:3e:8b:5d       yes                0.00
  3     be:d5:4b:c0:c2:bf       yes                0.00
  3     be:d5:4b:c0:c2:bf       yes                0.00
  4     c6:f8:f7:fd:b3:15       yes                0.00
  4     c6:f8:f7:fd:b3:15       yes                0.00

mininet> h1 ping -c 1 h4
PING 50.0.0.4 (50.0.0.4) 56(84) bytes of data.
64 bytes from 50.0.0.4: icmp_seq=1 ttl=64 time=0.049 ms

--- 50.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.049/0.049/0.049/0.000 ms
mininet> bridge brctl showmacs br0
port no mac addr                is local?       ageing timer
  1     00:00:00:00:01:00       no                 1.02
  4     00:00:00:00:04:00       no                 1.02
  1     00:00:00:00:08:00       yes                0.00
  1     00:00:00:00:08:00       yes                0.00
  2     7a:9d:67:3e:8b:5d       yes                0.00
  2     7a:9d:67:3e:8b:5d       yes                0.00
  3     be:d5:4b:c0:c2:bf       yes                0.00
  3     be:d5:4b:c0:c2:bf       yes                0.00
  4     c6:f8:f7:fd:b3:15       yes                0.00
  4     c6:f8:f7:fd:b3:15       yes                0.00
mininet> exit
mininet@mininet-vm:~/project3$ cat bridge_logs.txt
00:00:00:00:01:00 dev bridge-eth0 master br0
00:00:00:00:04:00 dev bridge-eth3 master br0
Deleted 00:00:00:00:01:00 dev bridge-eth0 master br0
Deleted 00:00:00:00:08:00 dev br0 master br0 permanent
Deleted 00:00:00:00:08:00 dev br0 vlan 1 master br0 permanent
```

First I ran the program which printed my initial table and after that I did a ping from h1 to h4 which sends one packet. After this if I do showmacs command in the mininet vm terminal it shows me the address of h1 and h4. My ageing timer never goes above 3 nor 5 the value is always within 2. Since the aging time got exceeded the entry got deleted and hence deleted is shown in the output.

Step3:

Host tcp dump logs for host 1 and host 2:

What is happening here is that the request is sent and then a response is received. Ping is sent from h1 to h2.

```
mininet@mininet-vm:~/project3$ cat h1logs.txt
17:39:54.148354 00:00:00:00:01:00 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request wh
o-has 50.0.0.2 tell 50.0.0.1, length 28
17:39:54.148374 00:00:00:00:02:00 > 00:00:00:00:01:00, ethertype ARP (0x0806), length 42: Reply 50.0
.0.2 is-at 00:00:00:00:02:00, length 28
17:39:54.148375 00:00:00:00:01:00 > 00:00:00:00:02:00, ethertype IPv4 (0x0800), length 98: 50.0.0.1
> 50.0.0.2: ICMP echo request, id 16681, seq 1, length 64
17:39:54.148383 00:00:00:00:02:00 > 00:00:00:00:01:00, ethertype IPv4 (0x0800), length 98: 50.0.0.2
> 50.0.0.1: ICMP echo reply, id 16681, seq 1, length 64
17:39:56.943926 00:00:00:00:01:00 > 00:00:00:00:02:00, ethertype IPv4 (0x0800), length 98: 50.0.0.1
> 50.0.0.2: ICMP echo request, id 16686, seq 1, length 64
17:39:56.943937 00:00:00:00:02:00 > 00:00:00:00:01:00, ethertype IPv4 (0x0800), length 98: 50.0.0.2
> 50.0.0.1: ICMP echo reply, id 16686, seq 1, length 64
```

Forwarding table:

```
mininet@mininet-vm:~/project3$ sudo python proj3.py
port no mac addr                is local?       ageing timer
  1     00:00:00:00:08:00       yes                0.00
  1     00:00:00:00:08:00       yes                0.00
  3     62:35:c8:e0:54:72       yes                0.00
  3     62:35:c8:e0:54:72       yes                0.00
  2     9a:d6:04:f4:1a:e5       yes                0.00
  2     9a:d6:04:f4:1a:e5       yes                0.00
  4     d2:a7:e1:04:cc:4b       yes                0.00
  4     d2:a7:e1:04:cc:4b       yes                0.00
```

Step4:

Sending traffic from host 1 to host 2

```
mininet> h1 ping -c 1 h2
PING 50.0.0.2 (50.0.0.2) 56(84) bytes of data.
64 bytes from 50.0.0.2: icmp_seq=1 ttl=64 time=0.046 ms

--- 50.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.046/0.046/0.046/0.000 ms
```

We sent 1 packets here and the above image is the output.

Step5:

Changes made to the forwarding table are as below as the bridge learns the address of host 1 and host 2

```
mininet> bridge brctl showmacs br0
port no mac addr                is local?       ageing timer
  1     00:00:00:00:01:00       no                 1.41
  2     00:00:00:00:02:00       no                 1.41
  1     00:00:00:00:08:00       yes                0.00
  1     00:00:00:00:08:00       yes                0.00
  3     62:b9:9b:c4:c9:b4       yes                0.00
  3     62:b9:9b:c4:c9:b4       yes                0.00
  2     ce:da:a1:f2:5d:20       yes                0.00
  2     ce:da:a1:f2:5d:20       yes                0.00
  4     ea:d1:f8:4d:d9:bf       yes                0.00
  4     ea:d1:f8:4d:d9:bf       yes                0.00
```

Step6:
So at host 1 and host 2, both will receive and send traffic without being aware of the switching that takes place. Just like a blackbox.

Step7:
Sending traffic again from host 1 to host 2

```
mininet> h1 ping -c 1 h2
PING 50.0.0.2 (50.0.0.2) 56(84) bytes of data.
64 bytes from 50.0.0.2: icmp_seq=1 ttl=64 time=0.028 ms

--- 50.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.028/0.028/0.028/0.000 ms
```

Step8:
The difference from step 4 is that the ping runs faster, the first time was 0.046ms and the second time was 0.028ms

Task 3: Why is learning at the bridge helpful?

Step1:
For the iperf traffic rate calculation:

(Note that I am not running my iperf commands in the terminal as they are in my code, I am just pinging my hosts)
Run the program
Run the ping from h1 to h3 and
Run the ping from h2 to h4
Exit the program quickly before ageing timer hits 0
Run the program again
Now set the ageing timer 0
Ping h1 to h3
Ping h2 to h4
Exit the program
The output for the program was shown using the data saved in the log file.

First Ping

```
mininet> h1 ping -c 1 h3
PING 50.0.0.3 (50.0.0.3) 56(84) bytes of data.
64 bytes from 50.0.0.3: icmp_seq=1 ttl=64 time=0.059 ms

--- 50.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.059/0.059/0.059/0.000 ms
mininet> h2 ping -c 1 h4
tcpdump: h2-eth1: SIOCETHTOOL(ETHTOOL_GET_TS_INFO) ioctl failed: No such device
PING 50.0.0.4 (50.0.0.4) 56(84) bytes of data.
64 bytes from 50.0.0.4: icmp_seq=1 ttl=64 time=0.048 ms

--- 50.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.048/0.048/0.048/0.000 ms
```

Setting ageing time to 0

```
mininet> bridge brctl setageing br0 0
```

Repeating ping again

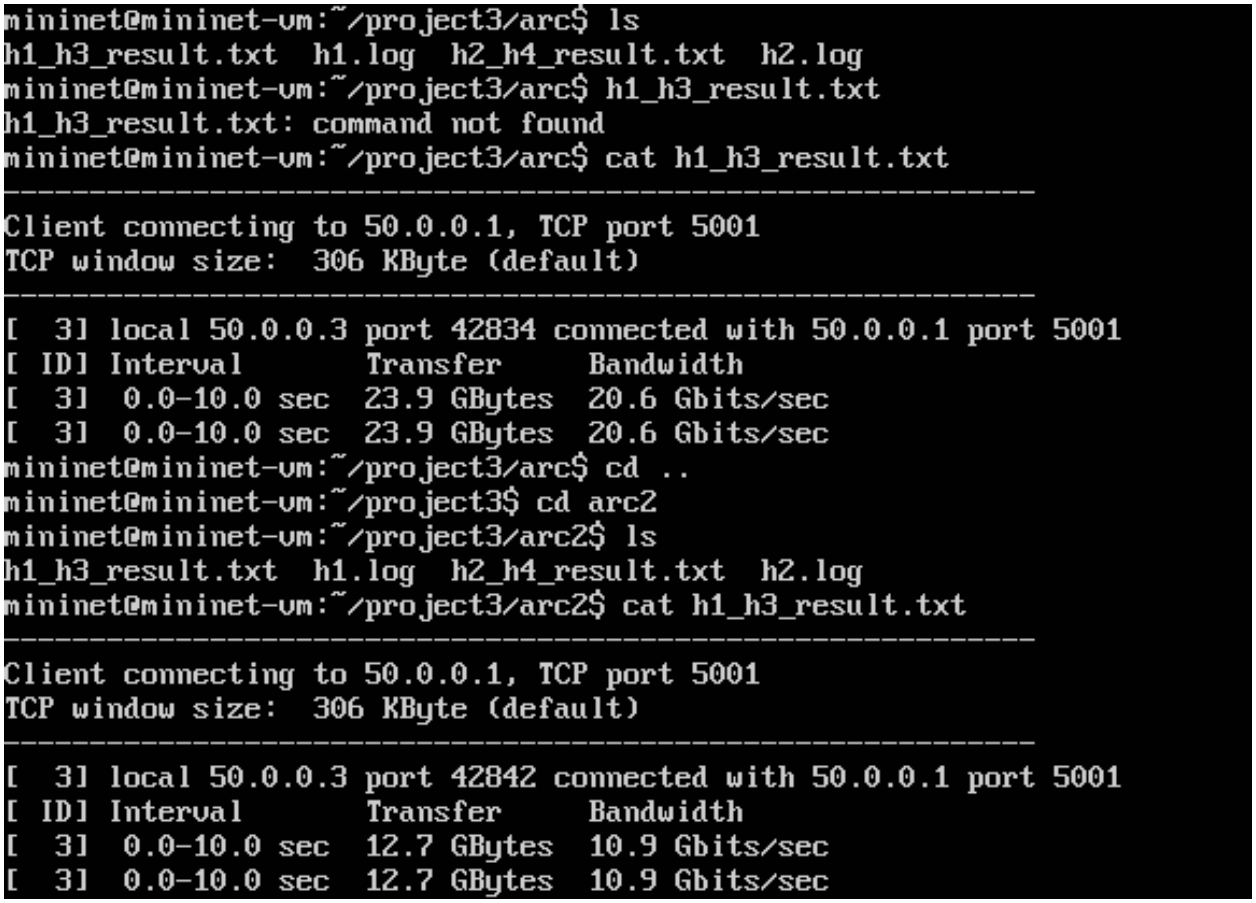
```
mininet> h1 ping -c 1 h3
PING 50.0.0.3 (50.0.0.3) 56(84) bytes of data.
64 bytes from 50.0.0.3: icmp_seq=1 ttl=64 time=0.042 ms

--- 50.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.042/0.042/0.042/0.000 ms
mininet> h2 ping -c 1 h4
tcpdump: h2-eth1: SIOCETHTOOL(ETHTOOL_GET_TS_INFO) ioctl failed: No such device
PING 50.0.0.4 (50.0.0.4) 56(84) bytes of data.
64 bytes from 50.0.0.4: icmp_seq=1 ttl=64 time=0.042 ms

--- 50.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.042/0.042/0.042/0.000 ms
```

Output from logfiles.

```
mininet@mininet-vm:~/project3/arc$ cat h1_h3_result.txt
------------------------------------------------------------
Client connecting to 50.0.0.1, TCP port 5001
TCP window size:  306 KByte (default)
------------------------------------------------------------
[  3] local 50.0.0.3 port 42834 connected with 50.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  23.9 GBytes  20.6 Gbits/sec
[  3]  0.0-10.0 sec  23.9 GBytes  20.6 Gbits/sec
mininet@mininet-vm:~/project3/arc$ cat h2_h4_result.txt
------------------------------------------------------------
Client connecting to 50.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 50.0.0.4 port 37060 connected with 50.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  25.0 GBytes  21.4 Gbits/sec
[  3]  0.0-10.0 sec  25.0 GBytes  21.4 Gbits/sec
mininet@mininet-vm:~/project3/arc$ cd ..
mininet@mininet-vm:~/project3$ cat h1_h3_result.txt
------------------------------------------------------------
Client connecting to 50.0.0.1, TCP port 5001
TCP window size:  306 KByte (default)
------------------------------------------------------------
[  3] local 50.0.0.3 port 42838 connected with 50.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  11.3 GBytes  9.70 Gbits/sec
mininet@mininet-vm:~/project3$ cat h2_h4_result.txt
------------------------------------------------------------
Client connecting to 50.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 50.0.0.4 port 37064 connected with 50.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  12.1 GBytes  10.4 Gbits/sec
[  3]  0.0-10.0 sec  12.1 GBytes  10.4 Gbits/sec
```

The traffic rate from h1 to h3 and h2 to h4 which shows that the bandwidth has been reduced in half approximately.

For a better understanding I reran the program but now only showing the h1 to h3.



I have two directory arc and arc2 in where arc contains the logs of h1 to h3 which is before setting the ageing timer to 0 and arc2 contains the logs of the h1 to h3 which is after setting the ageing timer to 0. As you can see in the above image the bandwidth is 20.6Gbits/sec before setting ageing time to 0. After setting the ageing time to 0 the bandwidth is 10.9Gbits/sec.