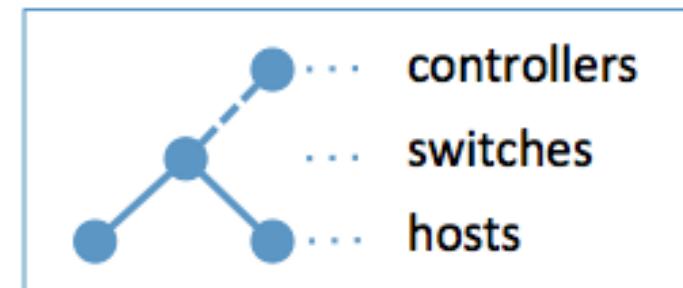


# Mininet

> sudo mn



# What is Mininet?

- A **virtual network environment** that can run on a single PC
- Runs real kernel, switch, and application code on a single machine
  - Command-line, UI, Python interfaces
- Many **OpenFlow features** are built-in
  - Useful: developing, deploying, and sharing

# Why Use Mininet?

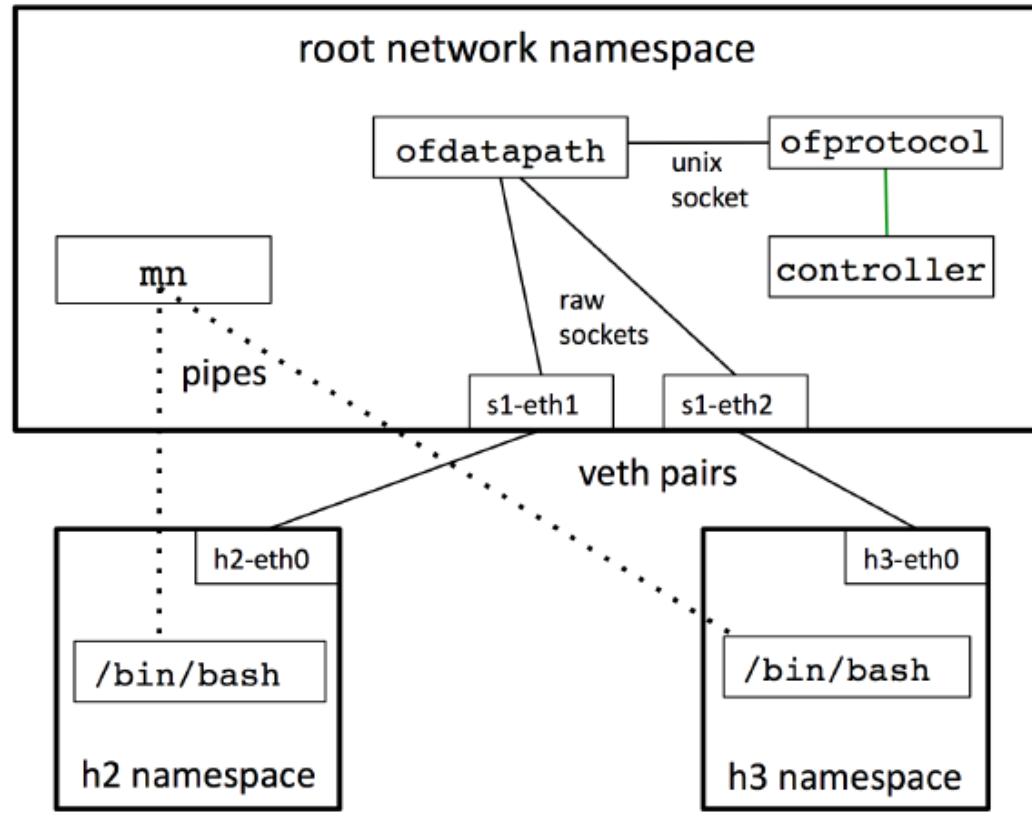
- Fast
- Possible to create custom topologies
- Can run real programs (anything that can run on Linux can run on a Mininet host)
- Programmable OpenFlow switches
- Easy to use
- Open source

# Alternatives

- **Real system:** Pain to configure
- **Networked VMs:** Scalability
- **Simulator:** No path to hardware deployment

# The Mininet VM in a Nutshell

## Virtual Machine

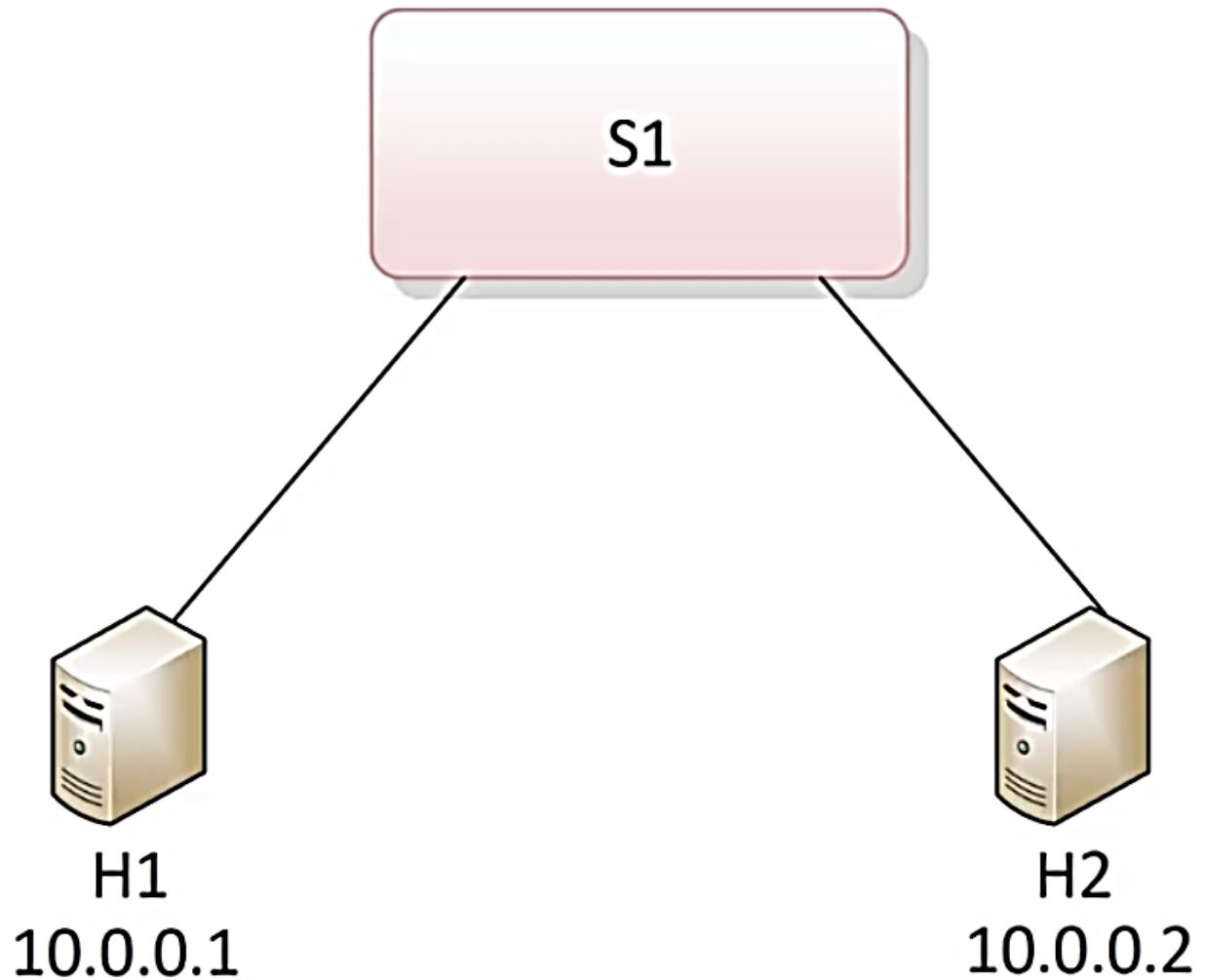


- Launch mininet process
- Per host
  - Bash process
  - Network namespace
- Create veth pairs and assign to namespaces
- Create OpenFlow switch to connect hosts
- Create OpenFlow controller

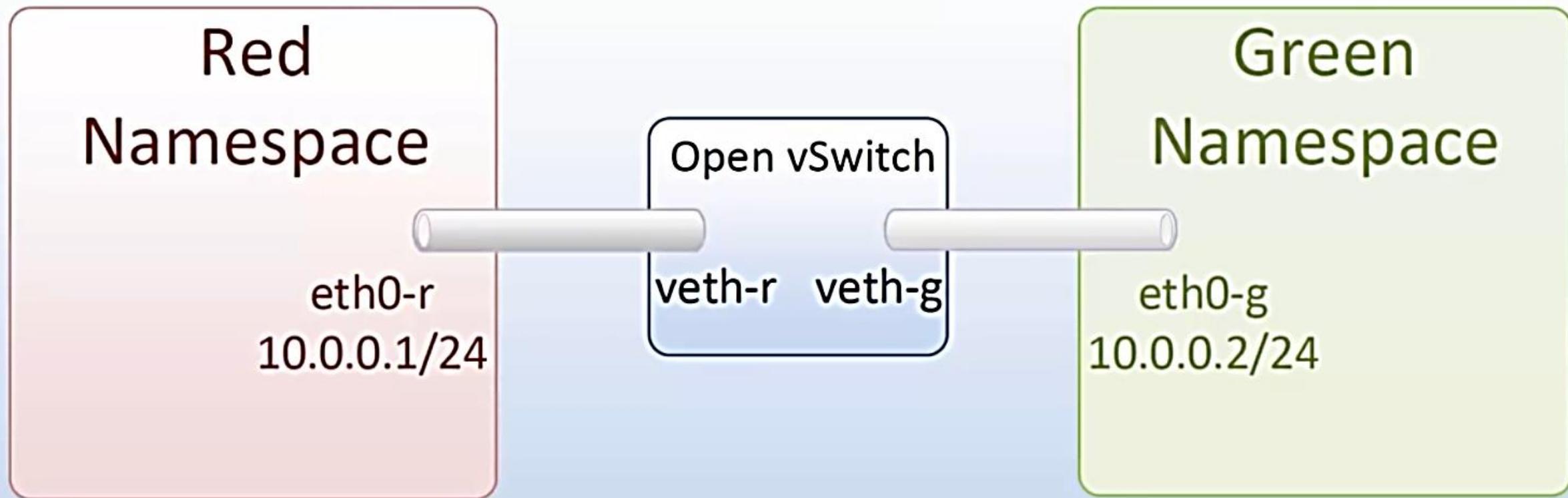
# What are Linux Network Namespaces?

- Multiple isolated networking environments running on a single physical host or VM
- Each network namespace has its own interfaces, routing tables and forwarding tables
- Processes can be dedicated to one network namespace
- Used in OpenStack, Mininet, Docker, more...

# Example



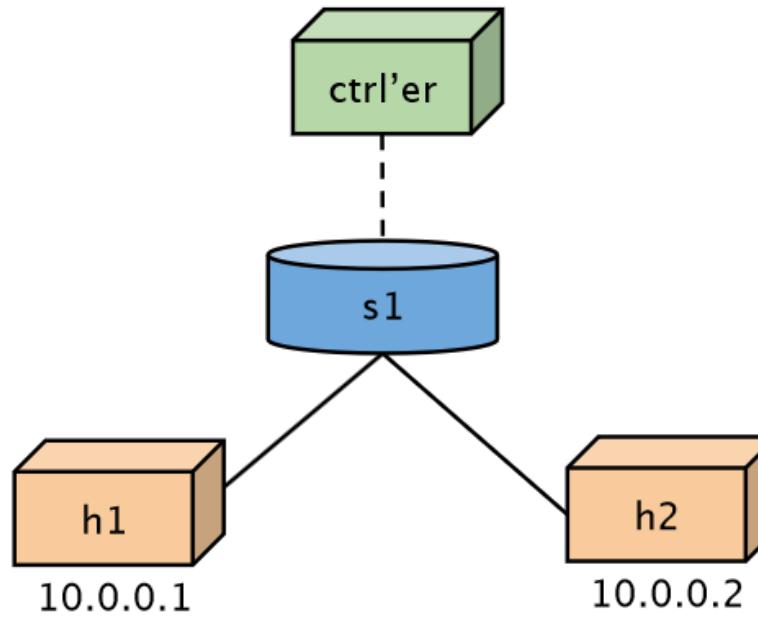
# Root Namespace



# Demo: basic network setup in Linux

```
sudo bash
# Create host namespaces
ip netns add h1
ip netns add h2
# Create switch
ovs-vsctl add-br s1
# Create links
ip link add h1-eth0 type veth peer name s1-eth1
ip link add h2-eth0 type veth peer name s1-eth2
ip link show
# Move host ports into namespaces
ip link set h1-eth0 netns h1
ip link set h2-eth0 netns h2
ip netns exec h1 ip link show
ip netns exec h2 ip link show
# Connect switch ports to OVS
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2
ovs-vsctl show
# Set up OpenFlow controller
ovs-vsctl set-controller s1 tcp:127.0.0.1
ovs-controller ptcp: &
ovs-vsctl show

# Configure network
ip netns exec h1 ifconfig h1-eth0 10.1
ip netns exec h1 ifconfig lo up
ip netns exec h2 ifconfig h2-eth0 10.2
ip netns exec h1 ifconfig lo up
ifconfig s1-eth1 up
ifconfig s1-eth2 up
# Test network
ip netns exec h1 ping -c1 10.2
```



# Testing a Simple Mininet Setup

- Try setting up a simple topology with three hosts connected to a single switch:
  - `sudo mn --test pingall --topo single,3`
- This setup uses a default switch controller and switch
  - Mininet also allows you to use custom remote controllers (and custom switches)

# Basic Mininet Command Line

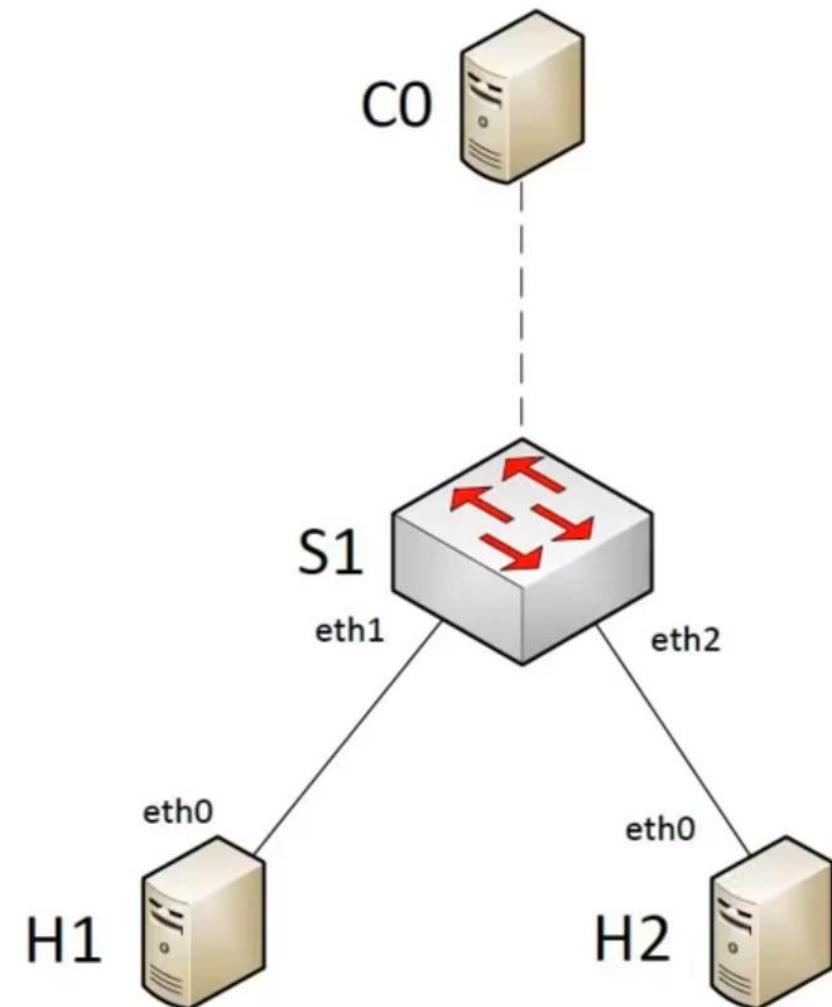
- **--topo** – defines a topology via command line upon mininet start-up.
- **--switch** – defines the switch to be used. By default the OVSK software switch is used.
- **--controller** – defines the controller to be used. If unspecified default controller is used with a default hub behavior.

# Trying Out Different Mininet Topologies

- Minimal network with two hosts, one (1) switch
  - sudo mn --topo minimal
- Example with 4 hosts and 4 switches
  - sudo mn --topo linear,4
- Example with 4 hosts all connected to one switch.
  - sudo mn --topo single,4
- Tree topology with defined depth and fan-out.
  - sudo mn --topo tree,depth=2,fanout=2

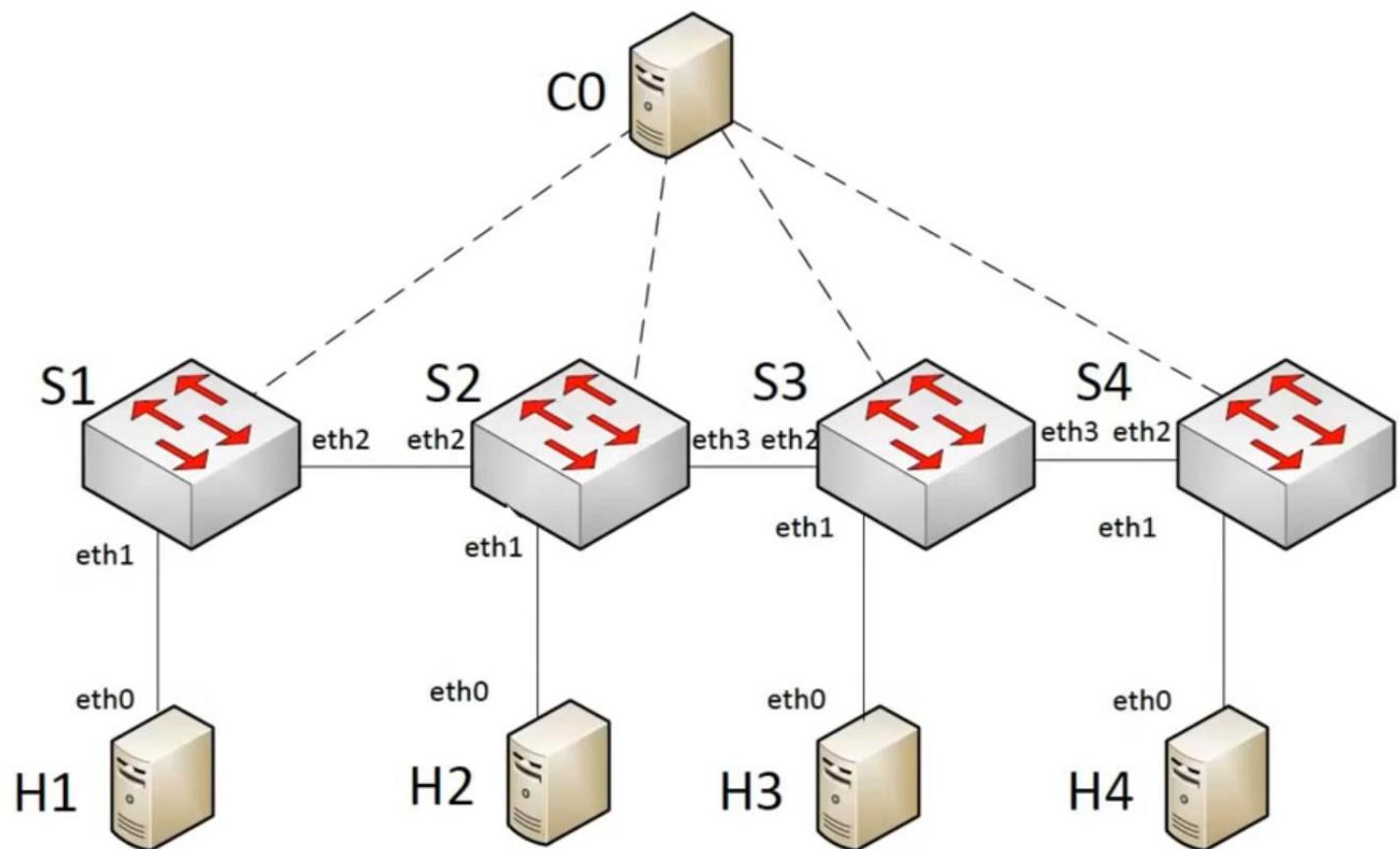
- Minimal network with two hosts, one (1) switch

- sudo mn --topo minimal



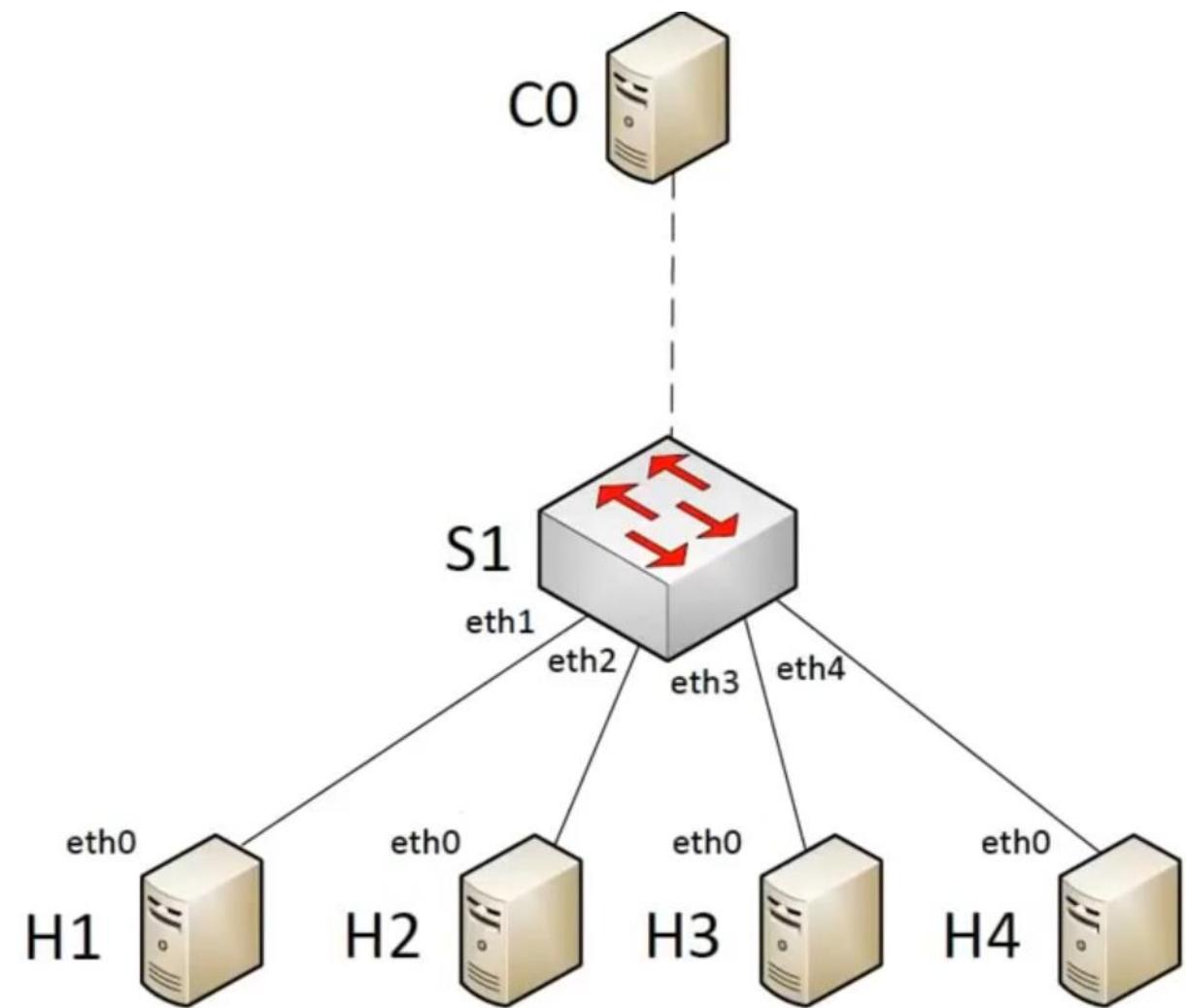
- Example with 4 hosts and 4 switches

- sudo mn --topo linear,4



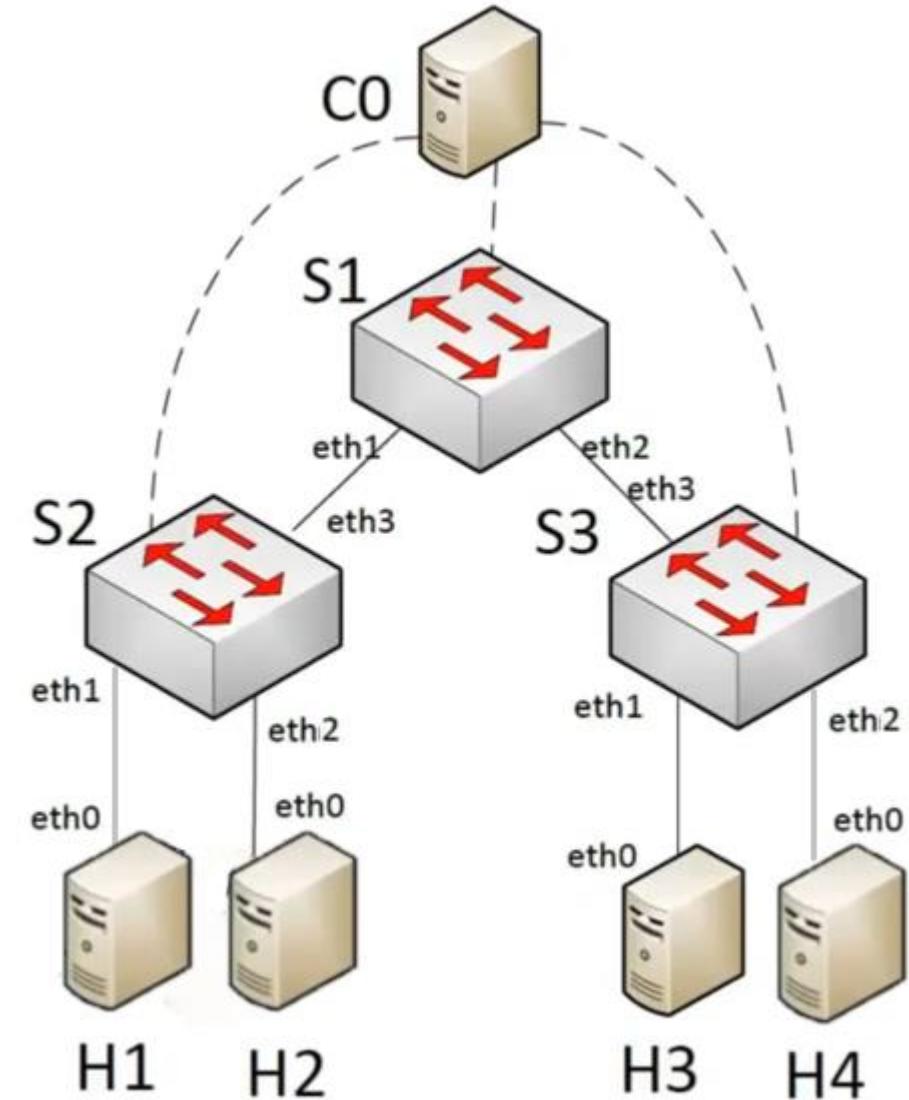
- Example with 4 hosts all connected to one switch.

- sudo mn --topo single,4



- Tree topology with defined depth and fan-out.

- sudo mn --topo tree,depth=2,fanout=2

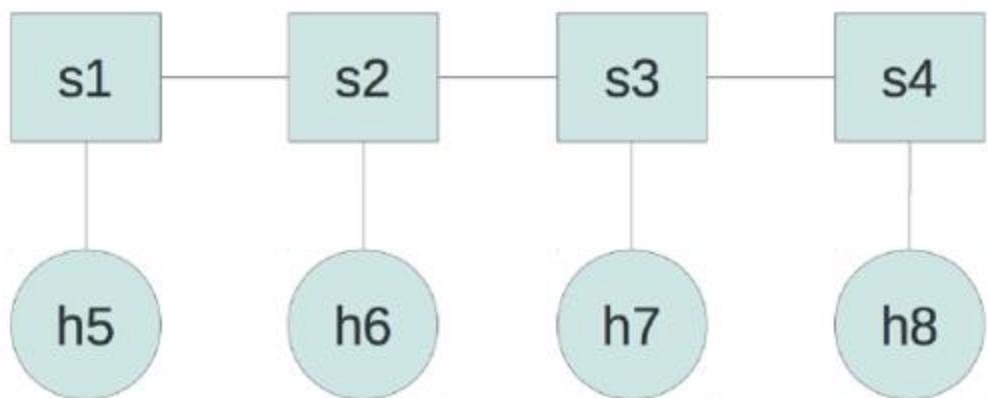


# How mn Works: mn executes Python

- “mn” is a launch script that executes Python
- Consider: “—topo linear,4”

```
from mininet.net import Mininet  
from mininet.topo import LinearTopo  
  
Linear = LinearTopo(k=4)  
  
net = Mininet(topo=Linear)
```

```
net.start()  
net.pingAll()  
net.stop()
```



# Writing Your Own Mininet Topologies

- Example: Two hosts, one switch
- **mininet.cli.CLI(net)** before `net.stop()` will escape to interactive CLI before script terminates
- **addLink** allows you to specify: Bandwidth (bw) in Mbps, Delay (delay), Maximum Queue Size (max\_queue\_size), Loss (loss) in percentage

```
from mininet.net import Mininet
from mininet.util import createLink
net = Mininet()

# Creating nodes in the network.
c0 = net.addController()
h0 = net.addHost('h0')
s0 = net.addSwitch('s0')
h1 = net.addHost('h1')

# Creating links between nodes in network (2-ways)
net.addLink(h0, s0)
net.addLink(h1, s0)

# Configuration of IP addresses in interfaces
h0.setIP('192.168.1.1', 24)
h1.setIP('192.168.1.2', 24)

net.start()
net.pingAll()
net.stop()
```

# More Complicated Topology Generation

```
#!/usr/bin/python
```

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel

class SingleSwitchTopo(Topo):
    "Single switch connected to n hosts."
    def __init__(self, n=2, **opts):
        # Initialize topology and default options
        Topo.__init__(self, **opts)
        switch = self.addSwitch('s1')

        # Python's range(N) generates 0..N-1
        for h in range(n):
            host = self.addHost('h%ss' % (h + 1))
            self.addLink(host, switch)
```

```
def simpleTest():
    "Create and test a simple network"
    topo = SingleSwitchTopo(n=4)
    net = Mininet(topo)
    net.start()
    print "Dumping host connections"
    dumpNodeConnections(net.hosts)
    print "Testing network connectivity"
    net.pingAll()
    net.stop()

if __name__ == '__main__':
    # Tell mininet to print useful information
    setLogLevel('info')
    simpleTest()
```

# Mininet Command Line Interface Usage

## ❖ Mininet Command Line Interface Usage

- *Interact with hosts and switches*

- **Start a minimal topology**

```
$ sudo mn
```

- **Start a minimal topology using a remote controller**

```
$ sudo mn --controller=remote,ip=[IP_ADDDR],port=[listening port]
```

- **Start a custom topology**

```
$ sudo mn --custom [topo_script_path] --topo=[topo_name]
```

- **Display nodes**

```
mininet> nodes
```

- **Display links**

```
mininet> net
```

- **Dump information about all nodes**

```
mininet> dump
```

# Mininet Command Line Interface Usage

## ❖ Mininet Command Line Interface Usage

### ▪ Interact with hosts and switches

- Check the IP address of a certain node

```
mininet> h1 ifconfig -a
```

- Print the process list from a host process

```
mininet> h1 ps -a
```

### ▪ Test connectivity between hosts

- Verify the connectivity by pinging from host1 to host2

```
mininet> h1 ping -c 1 h2
```

- Verify the connectivity between all hosts

```
mininet> pingall
```

# MiniNet commands

- (...) --link tc,bw=100,delay=1ms,loss=0,max\_queue\_size=1000,...
- ping (10 echo requests)
  - h1 ping h2 -c 10
- iperf
  - To perform a TCP bandwidth test between hosts
  - iperf h1 h2
- exit
  - Release resources

آزمایشگاه شبکه‌های کامپیوتری - ترم ۴۰۲۲

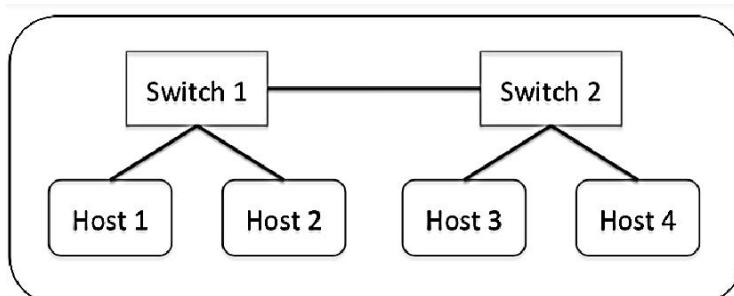
دستور کار آزمایش ۱: آشنایی با امولاتور Mininet

۴۰۲/۱۲/۸ مهلت ارسال:

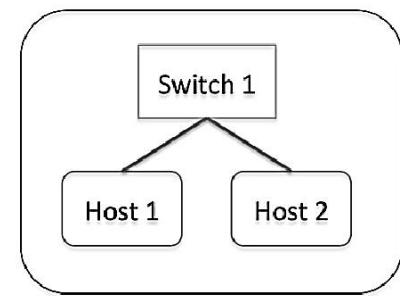
۱- در ماشین مجازی Mininet، سعی کنید حداقل ۵ مقصد مختلف را *ping* کنید (به عنوان مثال: وبسایت‌های مختلف). RTT اندازه‌گیری شده و نیز *ping* (TTL) حاصل از پاسخ *ping* را گزارش نمایید. به بیان خودتان توضیح دهید که این دو عدد نمایانگر چه هستند و اینکه آیا رابطه‌ای بین آنها وجود دارد؟

۲- دستور *ping* را از ماشین مجازی Mininet به مقصد کامپیوتر خود اجرا نمایید. پارامتر *ping* را طوری تنظیم کنید که دقیقاً ۵ بسته متوالی به مقصد ارسال نماید. هم‌زمان از Wireshark نیز استفاده نمایید تا درخواست‌ها و پاسخ‌های *ping* را capture کند (می‌توانید Wireshark را مستقیماً در کامپیوتر و یا در VM اجرا نمایید). به خاطر داشته باشید که از فیلترهای Wireshark استفاده کنید به نحوی که تنها درخواست‌ها و پاسخ‌های *ping* نمایش داده شوند و در مورد این فیلترهای مورد استفاده خود نیز توضیح دهید.

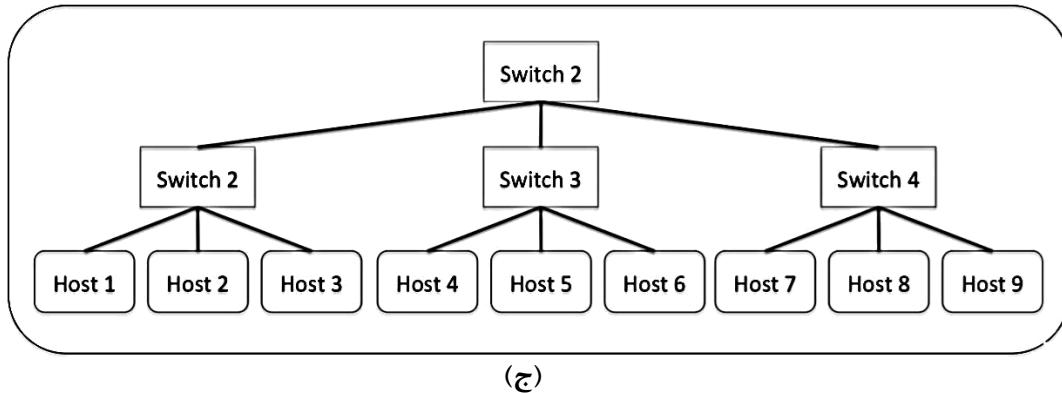
۳- در Mininet، چگونه از طریق تنظیم پارامترهای *topo*--، توپولوژی‌های نمایش‌داده شده در شکل زیر را تولید نماییم؟ دستور تک خطی مورد استفاده برای تولید این توپولوژی‌ها را بنویسید. همچنین، برای هریک از این توپولوژی‌ها از دستور *net* برای بررسی درستی توپولوژی تولیدی خود بهره بگیرید. پس از هر دستور، فهرست اتصالات را در گزارش خود بیان نمایید.



(ب)



(الف)



شکل ۱- توپولوژی‌های شبکه‌ای نمونه

۴- بار دیگر توپولوژی نشان‌داده شده در شکل ۱.(الف) را در نظر بگیرید. می‌خواهیم سناریوهایی با پارامترهای لینک (پهنهای باند، تأخیر) مختلف تولید نماییم. ابتدا، پهنهای باند را ثابت گرفته و تأخیر را با ۱۰ مقدار مختلف تنظیم نماییم؛ سپس، تأخیر را ثابت نگاه داشته و این بار، پهنهای باند را با ۱۰ مقدار متفاوت تنظیم کنید. برای هر سناریویی که تولید می‌کنید، از دستور *iperf* و *ping* برای اندازه‌گیری RTT و پهنهای باند بین دو host تحت پارامترهای لینک مختلف استفاده نمایید. نتایج اندازه‌گیری شده را نشان داده و ارتباط میان RTT، پهنهای باند و پارامترهای لینک را توضیح دهید.

پهنهای باند ثابت (bw=100Mbps)، تأخیر متغیر

Delay (ms)	RTT (ms)	Measured Bandwidth
0.01		
0.05		
0.1		
0.5		
1.0		
5.0		
10.0		
50.0		
100.0		
500.0		

تأخیر ثابت (delay=1ms)، پهنهای باند متغیر

Bandwidth (Mbits/sec)	RTT (ms)	Measured Bandwidth
0.01		
0.05		
0.1		
0.5		
1.0		
5.0		
10.0		
50.0		
100.0		
500.0		

## گزارشکار جلسه اول آزمایشگاه شبکه‌های کامپیوتری: آشنایی با Mininet

شکیبا انارکی – 99442047

بهاره کاووسی نژاد – 99431217

1 – در این سوال با استفاده از دستور ping، تعدادی request را ارسال می‌کنیم.

### RTT (Round-Trip Time)

RTT نشان‌دهنده زمان لازم برای ارسال یک بسته داده از یک دستگاه به دستگاه و دریافت پاسخ از آن دستگاه است. این زمان شامل تاخیر در ارسال داده‌ها، پردازش در مقصد و بازگشت اطلاعات به مبدأ است. RTT معیار مهمی برای ارزیابی عملکرد شبکه است، زیرا تأثیر مستقیم بر تجربه کاربر در هنگام استفاده از خدمات آنلاین دارد. مقادیر پایین‌تر RTT به معنای پاسخگویی سریع‌تر شبکه است.

### TTL (Time-To-Live)

TTL یک مقدار محدود‌کننده در بسته‌های داده‌ای است که تعیین می‌کند بسته برای چه مدت (یا چند هاپ) در شبکه زنده باقی می‌ماند تا به مقصد برسد. هر زمان که بسته‌ای از یک روتر به روتر دیگر منتقل می‌شود، مقدار TTL آن یک واحد کاهش می‌یابد. اگر TTL به صفر برسد و بسته هنوز به مقصد نرسیده باشد، بسته دور ریخته می‌شود. این مکانیزم جلوی حلقه‌های بی‌پایان در شبکه را می‌گیرد و اطمینان می‌دهد که بسته‌های گم شده یا خراب به طور بی‌پایان در شبکه گردش نکنند.

### رابطه بین TTL و RTT

در حالی که TTL و RTT هر دو مربوط به ارسال داده‌ها در شبکه‌ها هستند، آنها جنبه‌های مختلفی را نشان می‌دهند. RTT معیاری برای اندازه‌گیری سرعت و کارایی شبکه است، در حالی که TTL امنیت و قابلیت اطمینان شبکه را توسط جلوگیری از چرخش بی‌پایان بسته‌ها افزایش می‌دهد.

```
bahareh@bahareh:~$ ping www.google.com
PING www.google.com (216.239.38.120) 56(84) bytes of data.
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=1 ttl=49 time=305 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=2 ttl=49 time=187 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=3 ttl=49 time=113 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=4 ttl=49 time=132 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=5 ttl=49 time=155 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=6 ttl=49 time=179 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=7 ttl=49 time=109 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=8 ttl=49 time=110 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=9 ttl=49 time=110 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=10 ttl=49 time=113 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=11 ttl=49 time=109 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=12 ttl=49 time=110 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=13 ttl=49 time=109 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=14 ttl=49 time=114 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=15 ttl=49 time=110 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=16 ttl=49 time=111 ms
^C
--- www.google.com ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15787ms
rtt min/avg/max/mdev = 108.794/136.014/304.837/50.219 ms
```

```
bahareh@bahareh:~$ ping www.github.com
PING github.com (140.82.121.3) 56(84) bytes of data.
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=1 ttl=43 time=121 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=2 ttl=43 time=253 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=3 ttl=43 time=174 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=4 ttl=43 time=97.8 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=5 ttl=43 time=95.4 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=6 ttl=43 time=98.2 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=7 ttl=43 time=97.4 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=8 ttl=43 time=95.3 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=9 ttl=43 time=95.0 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=10 ttl=43 time=94.4 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=11 ttl=43 time=94.7 ms
^C
--- github.com ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 15626ms
rtt min/avg/max/mdev = 94.435/119.652/252.567/47.807 ms
bahareh@bahareh:~$
```

```
bahareh@bahareh:~$ ping www.quera.org
PING www.quera.org (185.143.233.61) 56(84) bytes of data.
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=1 ttl=50 time=102 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=2 ttl=50 time=9.03 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=3 ttl=50 time=7.43 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=4 ttl=50 time=48.9 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=5 ttl=50 time=6.94 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=6 ttl=50 time=12.0 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=7 ttl=50 time=7.63 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=8 ttl=50 time=11.8 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=9 ttl=50 time=7.53 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=10 ttl=50 time=8.10 ms
64 bytes from 185.143.233.61 (185.143.233.61): icmp_seq=11 ttl=50 time=11.5 ms
^C64 bytes from 185.143.233.61: icmp_seq=12 ttl=50 time=7.25 ms

--- www.quera.org ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 17716ms
rtt min/avg/max/mdev = 6.938/19.983/101.762/27.054 ms
bahareh@bahareh:~$
```

```

bahareh@bahareh:~$ ping -c 5 www.iust.ac.ir
PING www.iust.ac.ir (194.225.230.88) 56(84) bytes of data.
64 bytes from www.iust.ac.ir (194.225.230.88): icmp_seq=1 ttl=60 time=5.09 ms
64 bytes from www.iust.ac.ir (194.225.230.88): icmp_seq=2 ttl=60 time=11.8 ms
64 bytes from www.iust.ac.ir (194.225.230.88): icmp_seq=3 ttl=60 time=7.79 ms
64 bytes from www.iust.ac.ir (194.225.230.88): icmp_seq=4 ttl=60 time=10.1 ms
64 bytes from www.iust.ac.ir (194.225.230.88): icmp_seq=5 ttl=60 time=3.66 ms

--- www.iust.ac.ir ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 3.658/7.686/11.796/3.022 ms
bahareh@bahareh:~$
```

```

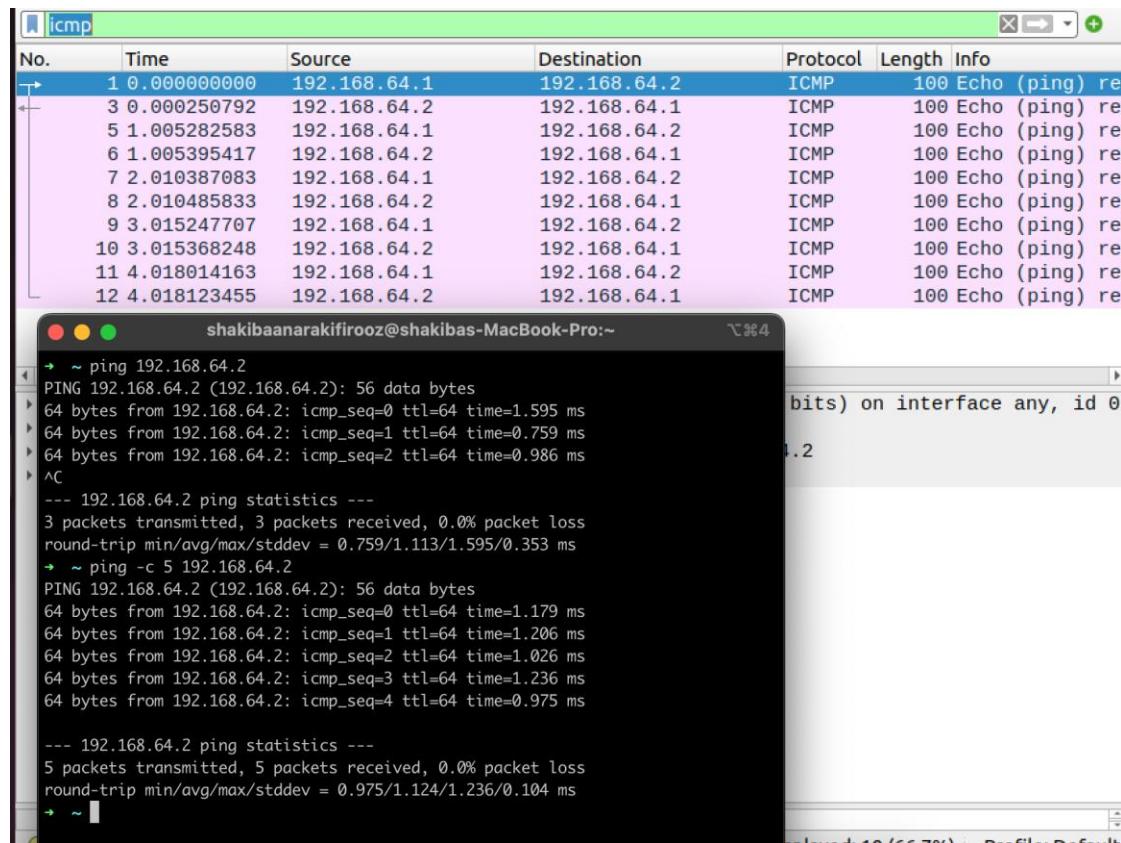
bahareh@bahareh:~$ ping -c 5 www.stackoverflow.com
PING www.stackoverflow.com (104.18.32.7) 56(84) bytes of data.
64 bytes from 104.18.32.7 (104.18.32.7): icmp_seq=1 ttl=47 time=267 ms
64 bytes from 104.18.32.7 (104.18.32.7): icmp_seq=2 ttl=47 time=384 ms
64 bytes from 104.18.32.7: icmp_seq=3 ttl=47 time=103 ms
64 bytes from 104.18.32.7 (104.18.32.7): icmp_seq=4 ttl=47 time=103 ms
64 bytes from 104.18.32.7: icmp_seq=5 ttl=47 time=169 ms

--- www.stackoverflow.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 19754ms
rtt min/avg/max/mdev = 103.017/205.330/383.946/107.561 ms
bahareh@bahareh:~$
```

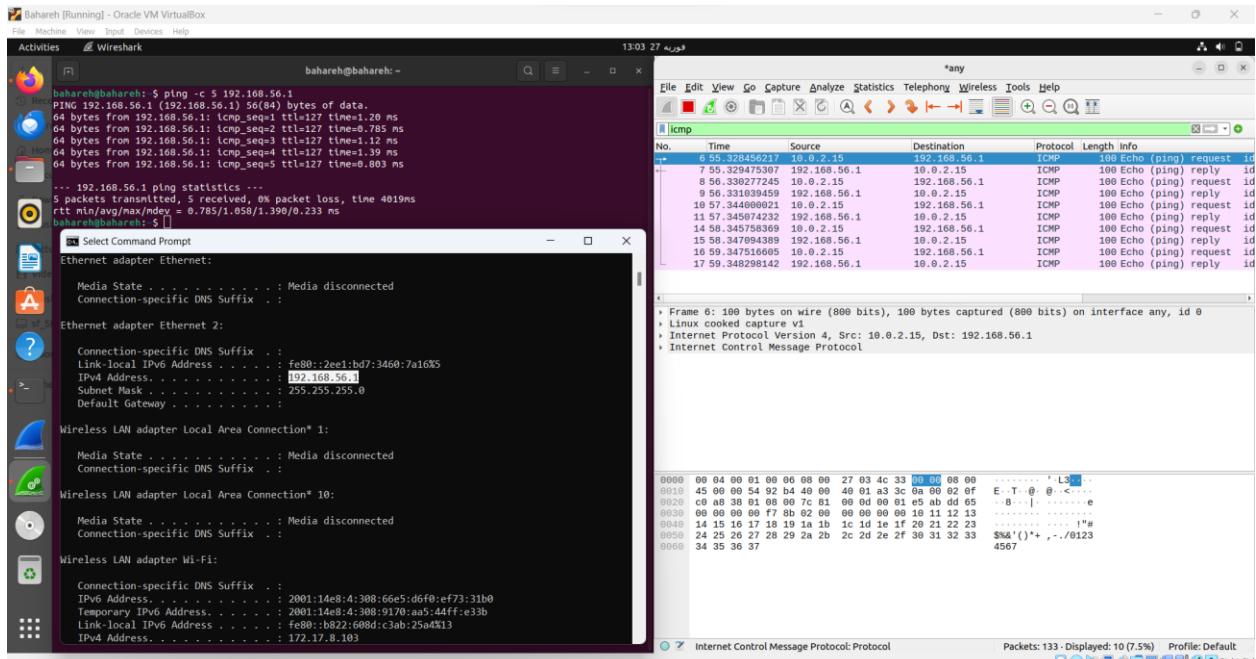
-2

- در مرحله اول ip خود را با دستور ipconfig پیدا می کنیم.
- با دستور [ip] ping -c 5 5 بسته را پشت سر هم ارسال می کنیم.
- در برنامه Wireshark پنل any را باز می کنیم و فیلتر icmp را اعمال می کنیم (زیرا ارتباط بین این دو دستگاه از نوع icmp می باشد) و capture کردن را آغاز می کنیم.

رسال بسته از VM به host



رسال بسته از Host به VM



(الف) 3

```
bahareh@bahareh:~$ sudo mn --topo single,2 --switch ovsk --controller ref
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

(ب)

```
bahareh@bahareh:~$ sudo mn --topo linear,2,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1s1 h1s2 h2s1 h2s2
*** Adding switches:
s1 s2
*** Adding links:
(h1s1, s1) (h1s2, s2) (h2s1, s1) (h2s2, s2) (s2, s1)
*** Configuring hosts
h1s1 h1s2 h2s1 h2s2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> net
h1s1 h1s1-eth0:s1-eth1
h1s2 h1s2-eth0:s2-eth1
h2s1 h2s1-eth0:s1-eth2
h2s2 h2s2-eth0:s2-eth2
s1 lo: s1-eth1:h1s1-eth0 s1-eth2:h2s1-eth0 s1-eth3:s2-eth3
s2 lo: s2-eth1:h1s2-eth0 s2-eth2:h2s2-eth0 s2-eth3:s1-eth3
c0
mininet> █
```

```
bahareh@bahareh:~$ sudo mn --topo tree,depth=2,fanout=3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s2, h1) (s2, h2) (s2, h3) (s3, h4) (s3, h5) (s3, h6) (s4, h7) (s4, h8) (s4, h9)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s2-eth3
h4 h4-eth0:s3-eth1
h5 h5-eth0:s3-eth2
h6 h6-eth0:s3-eth3
h7 h7-eth0:s4-eth1
h8 h8-eth0:s4-eth2
h9 h9-eth0:s4-eth3
s1 lo: s1-eth1:s2-eth4 s1-eth2:s3-eth4 s1-eth3:s4-eth4
s2 lo: s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:h3-eth0 s2-eth4:s1-eth1
s3 lo: s3-eth1:h4-eth0 s3-eth2:h5-eth0 s3-eth3:h6-eth0 s3-eth4:s1-eth2
s4 lo: s4-eth1:h7-eth0 s4-eth2:h8-eth0 s4-eth3:h9-eth0 s4-eth4:s1-eth3
c0
mininet>
```

#### 4-الف) پهنهای باند ثابت (100Mbps):

هنگامی که پهنهای باند ثابت است، تفاوت چندانی در ستون Measured Bandwidth مشاهده نمی‌شود (به جز سطر آخر). اما در مورد RTT می‌توان گفت که دارای یک رابطه مستقیم با Delay است و با افزایش تاخیر مقدار RTT نیز افزایش می‌یابد.

Delay (ms)	RTT(ms)	Measured Bandwidth
<b>0.01</b>	min/avg/max/mdev = 0.150/0.636/3.050/0.833	89.0 Mbits/sec
<b>0.05</b>	min/avg/max/mdev = 0.150/0.582/3.415/0.950	91.1 Mbits/sec
<b>0.1</b>	min/avg/max/mdev = 0.181/3.354/31.218/9.288	87.0 Mbits/sec
<b>0.5</b>	min/avg/max/mdev = 0.596/3.658/15.515/5.439	90.8 Mbits/sec
<b>1.0</b>	min/avg/max/mdev = 1.125/5.081/39.004/11.309	88.6 Mbits/sec
<b>5.0</b>	min/avg/max/mdev = 5.122/11.633/66.573/18.315	91.2 Mbits/sec
<b>10.0</b>	min/avg/max/mdev = 10.181/14.415/44.906/10.229	91.1 Mbits/sec
<b>50.0</b>	min/avg/max/mdev = 50.174/56.111/105.272/16.390	89.5 Mbits/sec
<b>100.0</b>	min/avg/max/mdev = 100.094/111.342/210.858/33.172	80.4 Mbits/sec
<b>500.0</b>	min/avg/max/mdev = 500.126/551.358/1009.572/152.738	7.92 Mbits/sec

## تصویر یک نمونه اجراء:

```
bahareh@bahareh:~$ sudo mn --topo single,2 --link tc,bw=100
[sudo] password for bahareh:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(100.00Mbit) *** Error: Warning: sch_htb: quantum of class 50001 is big. Consider r2q change.
(100.00Mbit) *** Error: Warning: sch_htb: quantum of class 50001 is big. Consider r2q change.
(h1, s1) (100.00Mbit) *** Error: Warning: sch_htb: quantum of class 50001 is big. Consider r2q change.
(100.00Mbit) *** Error: Warning: sch_htb: quantum of class 50001 is big. Consider r2q change.
(h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(100.00Mbit) *** Error: Warning: sch_htb: quantum of class 50001 is big. Consider r2q change.
(100.00Mbit) *** Error: Warning: sch_htb: quantum of class 50001 is big. Consider r2q change.

*** Starting CLI:
mininet> h1 tc qdisc add dev h1-eth0 root netem delay 0.01ms
mininet> h1 ping -c 10 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.05 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.479 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.223 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.150 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.153 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.173 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.758 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.314 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.301 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.761 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9198ms
rtt min/avg/max/mdev = 0.150/0.636/3.050/0.833 ms
mininet> h2 iperf -s &
mininet> h1 iperf -c h2 -t 10
-----
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[  1] local 10.0.0.1 port 42974 connected with 10.0.0.2 port 5001
[ ID] Interval      Transfer      Bandwidth
```

ب) تاخیر ثابت (1ms):

با افزایش پهنای باند، مقدار افزایش می‌یابد و مقدار میانگین RTT کاهش می‌یابد.

چند تصویر از مراحل انجام شده:

```
shakiba@shakiba-server:~$ sudo mn --topo single,2 --link tc,bw=0.01,delay=1ms
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(0.01Mbit 1ms delay) (0.01Mbit 1ms delay) (h1, s1) (0.01Mbit 1ms delay) (0.01Mbit 1ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...(0.01Mbit 1ms delay) (0.01Mbit 1ms delay)
*** Starting CLI:
mininet> h1 ping -c h2
ping: invalid argument: '10.0.0.2'
mininet> h1 ping -c 2 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=9.05 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=4.46 ms

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 4.458/6.753/9.049/2.295 ms
mininet> h2 iperf -s &
mininet> h1 iperf -c h2 -t 5
-----
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 1] local 10.0.0.1 port 41624 connected with 10.0.0.2 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 1] 0.0000-10.2884 sec   140 KBytes   112 Kbits/sec
mininet> 
```

```
shakiba@shakiba-server:~$ sudo mn --topo single,2 --link tc,bw=0.05,delay=1ms
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(0.05Mbit 1ms delay) (0.05Mbit 1ms delay) (h1, s1) (0.05Mbit 1ms delay) (0.05Mbit 1ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...(0.05Mbit 1ms delay) (0.05Mbit 1ms delay)
*** Starting CLI:
mininet> h1 ping -c 2 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=9.25 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=4.26 ms

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 4.257/6.755/9.253/2.498 ms
mininet> h2 iperf -s &
mininet> h1 iperf -c h2 -t 5
-----
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 1] local 10.0.0.1 port 37202 connected with 10.0.0.2 port 5001
[ ID] Interval Transfer Bandwidth
[ 1] 0.0000-10.3627 sec 246 KBytes 195 Kbits/sec
mininet> 
```

```
shakiba@shakiba-server:~$ sudo mn --topo single,2 --link tc,bw=0.1,delay=1ms
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(0.10Mbit 1ms delay) (0.10Mbit 1ms delay) (h1, s1) (0.10Mbit 1ms delay) (0.10Mbit 1ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...(0.10Mbit 1ms delay) (0.10Mbit 1ms delay)
*** Starting CLI:
mininet> h1 ping -c 2 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=10.1 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=4.33 ms

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 4.334/7.232/10.131/2.898 ms
mininet> h2 iperf -s &
mininet> h1 iperf -c h2 -t 2
-----
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 1] local 10.0.0.1 port 47316 connected with 10.0.0.2 port 5001
[ ID] Interval Transfer Bandwidth
[ 1] 0.0000-4.0686 sec 182 KBytes 367 Kbits/sec
mininet> █
```

<b>Bandwidth</b>	<b>RTT</b> min/avg/max/mdev ms	<b>Measured Bandwidth</b>
<b>0.01</b>	4.458/6.753/9.049/2.295	112 Kbits/sec
<b>0.05</b>	4.257/6.755/9.253/2.498	195 Kbits/sec
<b>0.1</b>	4.334/7.232/10.131/2.898	367 Kbits/sec
<b>0.5</b>	4.369/4.394/4.419/0.025	935 Kbits/sec
<b>1.0</b>	4.342/6.976/9.610/2.634	1.17Mbits/sec
<b>5.0</b>	4.267/6.505/8.744/2.238	4.77Mbits/sec
<b>10.0</b>	4.291/4.817/5.343/0.526	9.52 Mbits/sec
<b>50.0</b>	4.237/6.750/9.264/2.513	47.6 Mbits/sec
<b>100.0</b>	4.437/7.144/9.851/2.707	94.9 Mbits/sec
<b>500.0</b>	4.332/6.731/9.130/2.399	473 Mbits/sec

## گزارش ۱

بکتاش انصاری

پوریا رحیمی

سوال ۱ :

همانطور که در عکس ها پیداست برای هر ریکوئست، یک **time to live** ای وجود دارد برای مثال در عکس اول برای سایت [github.com](https://github.com) این مقدار برابر است با : 43 برای مقدار **rtt** نیز پس از پایان دستور پینگ، از بین ریکوئست های زده شده مقدار میانگین، بیشترین و کمترین مقدار گزارش شده است که برای مثال اول این مقادیر برابر است با:

$\text{min/avg/max/mdev} = 172.495/209.11/260.022/29.150 \text{ ms}$

به زمانی میگویند که یک **packet** درون شبکه وجود دارد (زنده است) تا زمانی که به وسیله‌ی یک **router** از بین برود. به طوری که هر بار پکت به یک **router** میرسد از این عدد یک مقدار کم میشود تا وقتی که درون یک **router** این عدد به صفر برسد و دیگر ارسال آن ادامه پیدا نخواهد کرد. این کار برای این است که پکت ها درون شبکه سرگردان نشوند و تا ابد داخل شبکه نمانند.

**Rtt** نیز که مخفف **round trip time** میباشد به زمانی میگویند که از ابتدای یک ریکوئست شروع شده و تا رسیدن **response** به مبدا ادامه دارد.

رابطه بین این دو مقدار بصورت غیرمستقیم میباشد به طوری که افزایش TTL میتواند به افزایش RTT منجر شود و بالعکس. این امر به دلیل تاثیر TTL بر مسیریابی و مسیر ارسالی پکتها است. در برخی مواقع، اگر TTL بسیار پایین باشد، پکتها ممکن است به دست مقصد نرسند و از مسیر حذف شوند. این ممکن است باعث افزایش RTT شود.

```
baktash@baktash:~$ ping github.com
PING github.com (140.82.121.3) 56(84) bytes of data.
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=1 ttl=43 time=172 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=2 ttl=43 time=195 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=3 ttl=43 time=217 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=4 ttl=43 time=202 ms
64 bytes from lb-140-82-121-3-fra.github.com (140.82.121.3): icmp_seq=5 ttl=43 time=260 ms
^C
--- github.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 172.495/209.211/260.022/29.150 ms
baktash@baktash:~$
```

```
baktash@baktash:~$ ping varzesh3.com
PING varzesh3.com (94.182.113.153) 56(84) bytes of data.
64 bytes from 94-182-113-153.shatel.ir (94.182.113.153): icmp_seq=1 ttl=53 time=40.6 ms
64 bytes from 94-182-113-153.shatel.ir (94.182.113.153): icmp_seq=2 ttl=53 time=23.0 ms
64 bytes from 94-182-113-153.shatel.ir (94.182.113.153): icmp_seq=3 ttl=53 time=32.8 ms
64 bytes from 94-182-113-153.shatel.ir (94.182.113.153): icmp_seq=4 ttl=53 time=24.9 ms
64 bytes from 94-182-113-153.shatel.ir (94.182.113.153): icmp_seq=5 ttl=53 time=30.9 ms
^C
--- varzesh3.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4011ms
rtt min/avg/max/mdev = 23.020/30.436/40.648/6.259 ms
baktash@baktash:~$
```

```
baktash@baktash:~$ ping yahoo.com
PING yahoo.com (98.137.11.164) 56(84) bytes of data.
64 bytes from media-router-fp73.prod.media.vip.gq1.yahoo.com (98.137.11.164): icmp_seq=1 ttl=32 time=290 ms
64 bytes from media-router-fp73.prod.media.vip.gq1.yahoo.com (98.137.11.164): icmp_seq=2 ttl=32 time=344 ms
64 bytes from media-router-fp73.prod.media.vip.gq1.yahoo.com (98.137.11.164): icmp_seq=3 ttl=32 time=411 ms
64 bytes from media-router-fp73.prod.media.vip.gq1.yahoo.com (98.137.11.164): icmp_seq=4 ttl=32 time=332 ms
64 bytes from media-router-fp73.prod.media.vip.gq1.yahoo.com (98.137.11.164): icmp_seq=5 ttl=32 time=311 ms
^C
--- yahoo.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 290.370/337.630/410.781/40.919 ms
baktash@baktash:~$
```

```
baktash@baktash:~$ ping time.ir
PING time.ir (185.13.228.162) 56(84) bytes of data.
64 bytes from 185.13.228.162.pol.ir (185.13.228.162): icmp_seq=1 ttl=114 time=25.6 ms
64 bytes from 185.13.228.162.pol.ir (185.13.228.162): icmp_seq=2 ttl=114 time=32.4 ms
64 bytes from 185.13.228.162.pol.ir (185.13.228.162): icmp_seq=3 ttl=114 time=26.8 ms
64 bytes from 185.13.228.162.pol.ir (185.13.228.162): icmp_seq=4 ttl=114 time=21.8 ms
64 bytes from 185.13.228.162.pol.ir (185.13.228.162): icmp_seq=5 ttl=114 time=37.1 ms
^C
--- time.ir ping statistics ---
6 packets transmitted, 5 received, 16.6667% packet loss, time 5008ms
rtt min/avg/max/mdev = 21.763/28.731/37.115/5.398 ms
baktash@baktash:~$
```

سوال ۲:

همانطور که در عکس ها مشخص است ابتدا به mininet متصل شدم و آن را در شبکه پیدا کردم و سپس ping زدم.

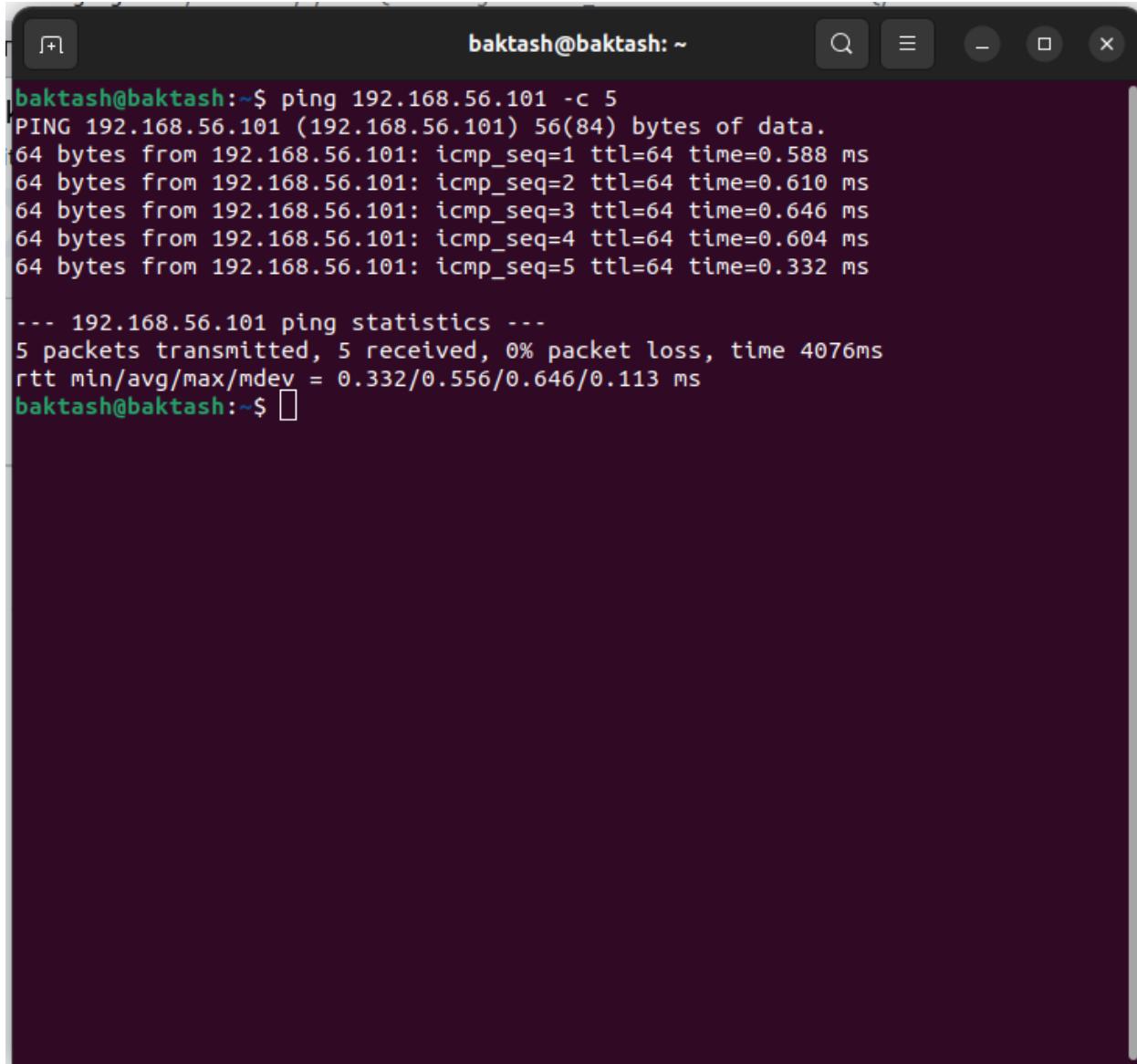
Mininet-VM [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

```
mininet@mininet-vm:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.101 netmask 255.255.255.0 broadcast 192.168.56.255
        ether 08:00:27:f7:79:cf txqueuelen 1000 (Ethernet)
        RX packets 2 bytes 1180 (1.1 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 2 bytes 684 (684.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 40 bytes 3064 (3.0 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 40 bytes 3064 (3.0 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet@mininet-vm:~$
```



A screenshot of a terminal window titled "baktash@baktash: ~". The window contains the following text output from the "ping" command:

```
baktash@baktash:~$ ping 192.168.56.101 -c 5
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=0.588 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=0.610 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=0.646 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=64 time=0.604 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=64 time=0.332 ms

--- 192.168.56.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4076ms
rtt min/avg/max/mdev = 0.332/0.556/0.646/0.113 ms
baktash@baktash:~$
```

سؤال ٣:

الف:

```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> [ ]
```

(ب)

```
mininet@mininet-vm:~$ sudo mn --topo linear,2,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1s1 h1s2 h2s1 h2s2
*** Adding switches:
s1 s2
*** Adding links:
(h1s1, s1) (h1s2, s2) (h2s1, s1) (h2s2, s2) (s2, s1)
*** Configuring hosts
h1s1 h1s2 h2s1 h2s2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> net
h1s1 h1s1-eth0:s1-eth1
h1s2 h1s2-eth0:s2-eth1
h2s1 h2s1-eth0:s1-eth2
h2s2 h2s2-eth0:s2-eth2
s1 lo:  s1-eth1:h1s1-eth0 s1-eth2:h2s1-eth0 s1-eth3:s2-eth3
s2 lo:  s2-eth1:h1s2-eth0 s2-eth2:h2s2-eth0 s2-eth3:s1-eth3
c0
mininet> 
```

(ج)

```
lp mininet@mininet-vm: ~
mininet@mininet-vm:~$ sudo mn --topo tree,depth=2,fanout=3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s2, h1) (s2, h2) (s2, h3) (s3, h4) (s3, h5) (s3, h6) (s4, h7) (s4, h8) (s4, h9)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s2-eth3
h4 h4-eth0:s3-eth1
h5 h5-eth0:s3-eth2
h6 h6-eth0:s3-eth3
h7 h7-eth0:s4-eth1
h8 h8-eth0:s4-eth2
h9 h9-eth0:s4-eth3
s1 lo: s1-eth1:s2-eth4 s1-eth2:s3-eth4 s1-eth3:s4-eth4
s2 lo: s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:h3-eth0 s2-eth4:s1-eth1
s3 lo: s3-eth1:h4-eth0 s3-eth2:h5-eth0 s3-eth3:h6-eth0 s3-eth4:s1-eth2
s4 lo: s4-eth1:h7-eth0 s4-eth2:h8-eth0 s4-eth3:h9-eth0 s4-eth4:s1-eth3
c0
mininet> 
```

سؤال ٤:

(الف)

Delay(ms)	RTT(ms)(avrg for 5) (ping h1 h2)	Measured Bandwidth (h1 h2)
0.01	0.518	91.3 - 109 Mbits/sec
0.05	0.918	91.3 - 108 Mbits/sec
0.1	1.041	92.7 - 109 Mbits/sec
0.5	3.051	93.4 - 110 Mbits/sec
1	7.570	92.5 - 109 Mbits/sec
5	25.041	91.6 - 108 Mbits/sec
10	49.843	89.4 - 105 Mbits/sec
50	242.205	72.3 - 82 Mbits/sec
100	482.778	45.3 - 49.6 Mbits/sec
500	2600.698	0.314 - 0.524 Mbits/sec 314 - 524 Kbits/sec

(ب)

Bandwidth Mbps / sec	RTT(ms)(avg for 5) (ping h1 h2)	Measured Bandwidth (h1 h2)
0.01	7.854	9.5 - 187 Kbps/sec
0.05	6.260	48 - 443 Kbps/sec
0.1	6.086	96.4 - 378 Kbps/sec
0.5	6.039	479 - 975 Kbps/sec
1	6.090	958 Kbps/sec - 1.58 Mbps/sec
5	5.790	4.75- 5.94 Mbps/sec
10	5.853	9.46 - 11.4 Mbps/sec
50	7.109	47.4 - 56.8 Mbps/sec
100	5.748	91.6 - 108 Mbps/sec
500	5.866	428 - 447 Mbps/sec

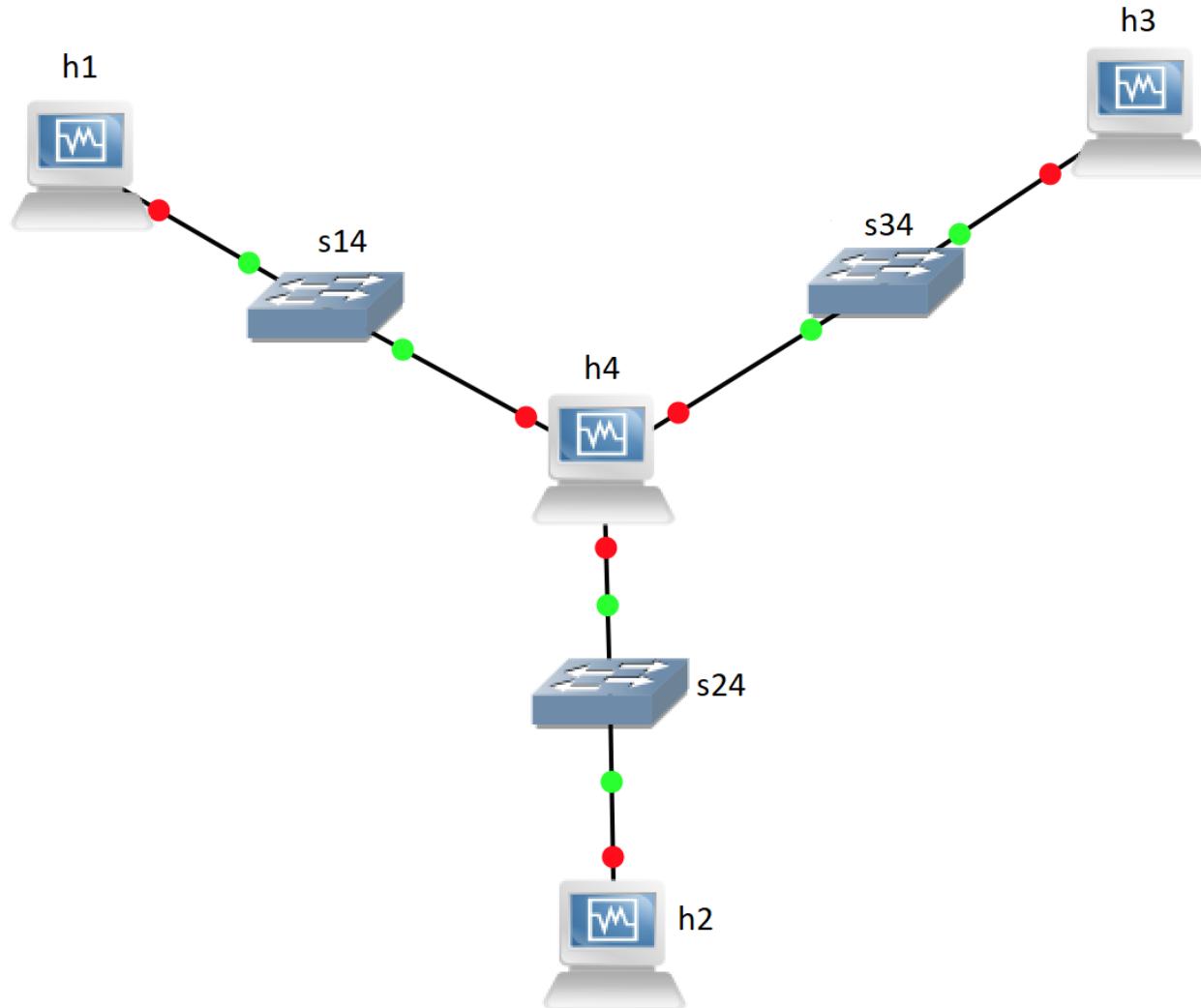
همانطور که در جدول مشاهده میکنید، در حالت اول که مقدار پهنهای باند ثابت است ما با افزایش تاخیر در شبکه پهنهای باند به مرور کمتر ولی RTT بیشتری را خواهیم داشت. البته باید توجه کنیم که در تاخیر های کوچک و نزدیک به هم مقدار پهنهای باند ها نیز تقریباً یکی است و آنچنان این افزایش تاخیر بر روی آنها تاثیر ندارد اما با تغییرات شدید تاخیر مقدار پهنهای باند هم تغییر شدیدی میکند و کاهش می یابد.

در حالت دوم طبق جدول با افزایش پهنهای باند و با ثابت ماندن تاخیر مقدار پهنهای باند افزایش داشته اما مقدار تاخیر آنچنان تغییر قابل توجهی نداشته است و مقادیر اندازه گیری شده نزدیک به هم هستند.

در نتیجه میتوانیم بگوییم در حالت اول تغییر تاخیر آنچنان تاثیری بر روی پهنهای باند نداشت (به جز مقادیر آخر) و در حالت دوم تغییر پهنهای باند آنچنان تاثیری بر روی تاخیر نداشت.

# LAN Configuration

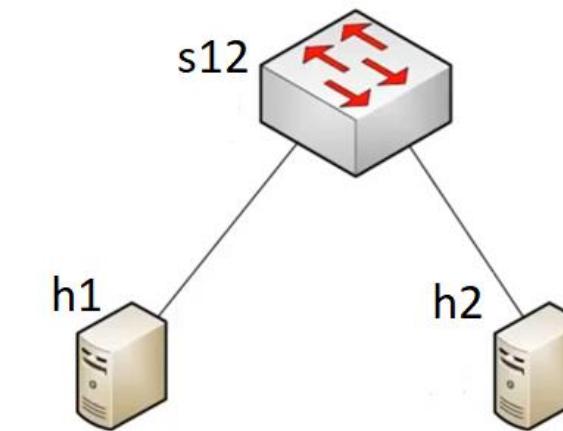
# Custom topology



```

1  #!/usr/bin/python
2  """
3  This example shows how to create a Mininet object and add nodes to it
4  """
5  #Importing Libraries
6  from mininet.net import Mininet
7  from mininet.node import Controller
8  from mininet.cli import CLI
9  from mininet.log import setLogLevel, info
10
11 #Function definition: This is called from the main function
12 def firstNetwork():
13     #Create an empty network and add nodes to it.
14     net = Mininet()
15     info( '*** Adding controller\n' )
16     net.addController( 'c0' )
17
18     info( '*** Adding hosts\n' )
19     h1 = net.addHost( 'h1', ip='10.0.0.1' )
20     h2 = net.addHost( 'h2' )
21
22     info( '*** Adding switch\n' )
23     s12 = net.addSwitch( 's12' )
24
25     info( '*** Creating links\n' )
26     net.addLink( h1, s12 )
27     net.addLink( h2, s12 )
28
29     info( '*** Starting network\n' )
30     net.start()
31
32     #This is used to run commands on the hosts
33
34     info( '*** Starting xterm on hosts\n' )
35     h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
36     h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')
37
38     info( '*** Running the command line interface\n' )
39     CLI( net )
40
41     info( '*** Closing the terminals on the hosts\n' )
42     h1.cmd("killall xterm")
43     h2.cmd("killall xterm")
44
45     info( '*** Stopping network' )
46     net.stop()

```



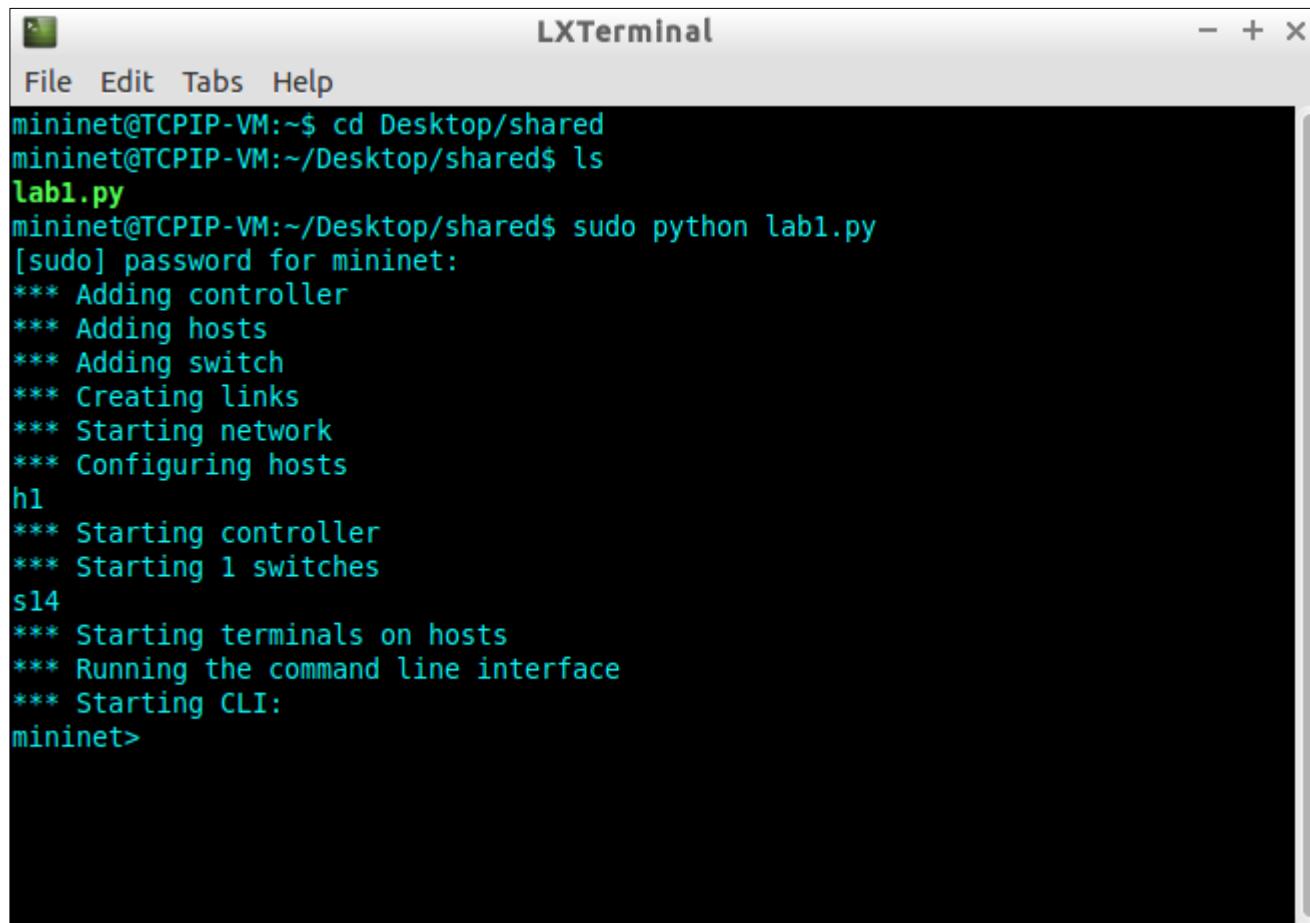
```

47
48     #main Function: This is called when the Python file is run
49     if __name__ == '__main__':
50         setLogLevel( 'info' )
51         firstNetwork()
52

```

# Custom topology

- Change directory to shared folder:
  - \$ cd Desktop/shared
- Edit a python file, e.g. lab1.py:
  - \$ sudo leafpad lab1.py
- Run topology:
  - \$ sudo python lab1.py
- Exit topology:
  - mininet> exit
- Clean up:
  - \$ sudo mn -c



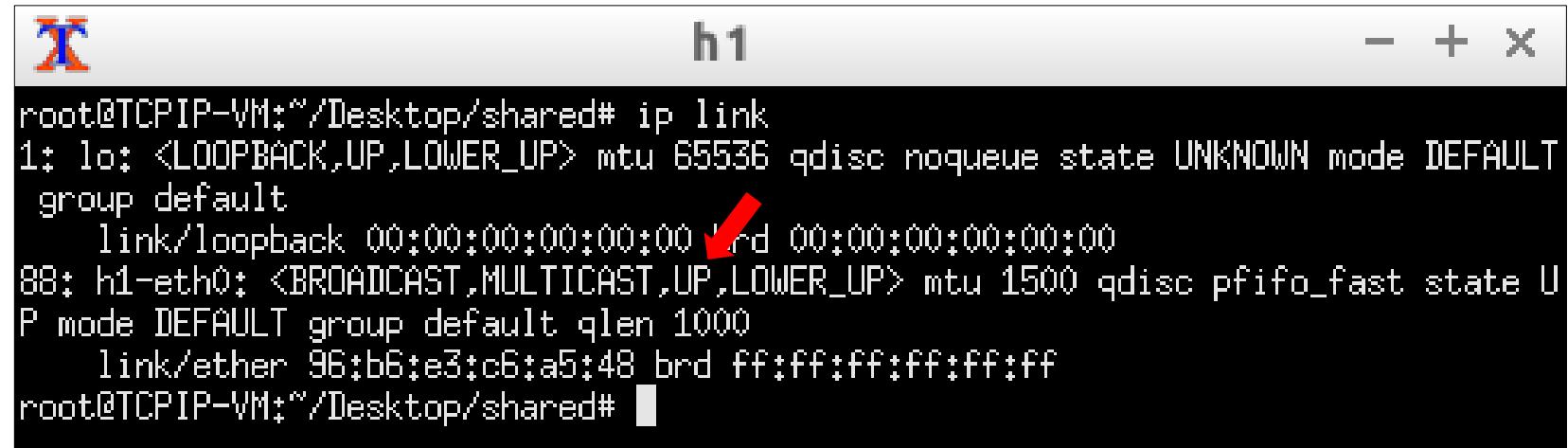
The screenshot shows an LXTerminal window with the following command-line session:

```
File Edit Tabs Help
mininet@TCPIP-VM:~$ cd Desktop/shared
mininet@TCPIP-VM:~/Desktop/shared$ ls
lab1.py
mininet@TCPIP-VM:~/Desktop/shared$ sudo python lab1.py
[sudo] password for mininet:
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1
*** Starting controller
*** Starting 1 switches
s14
*** Starting terminals on hosts
*** Running the command line interface
*** Starting CLI:
mininet>
```

# Interfaces

- Show the mode of a host interfaces:

- # ip link

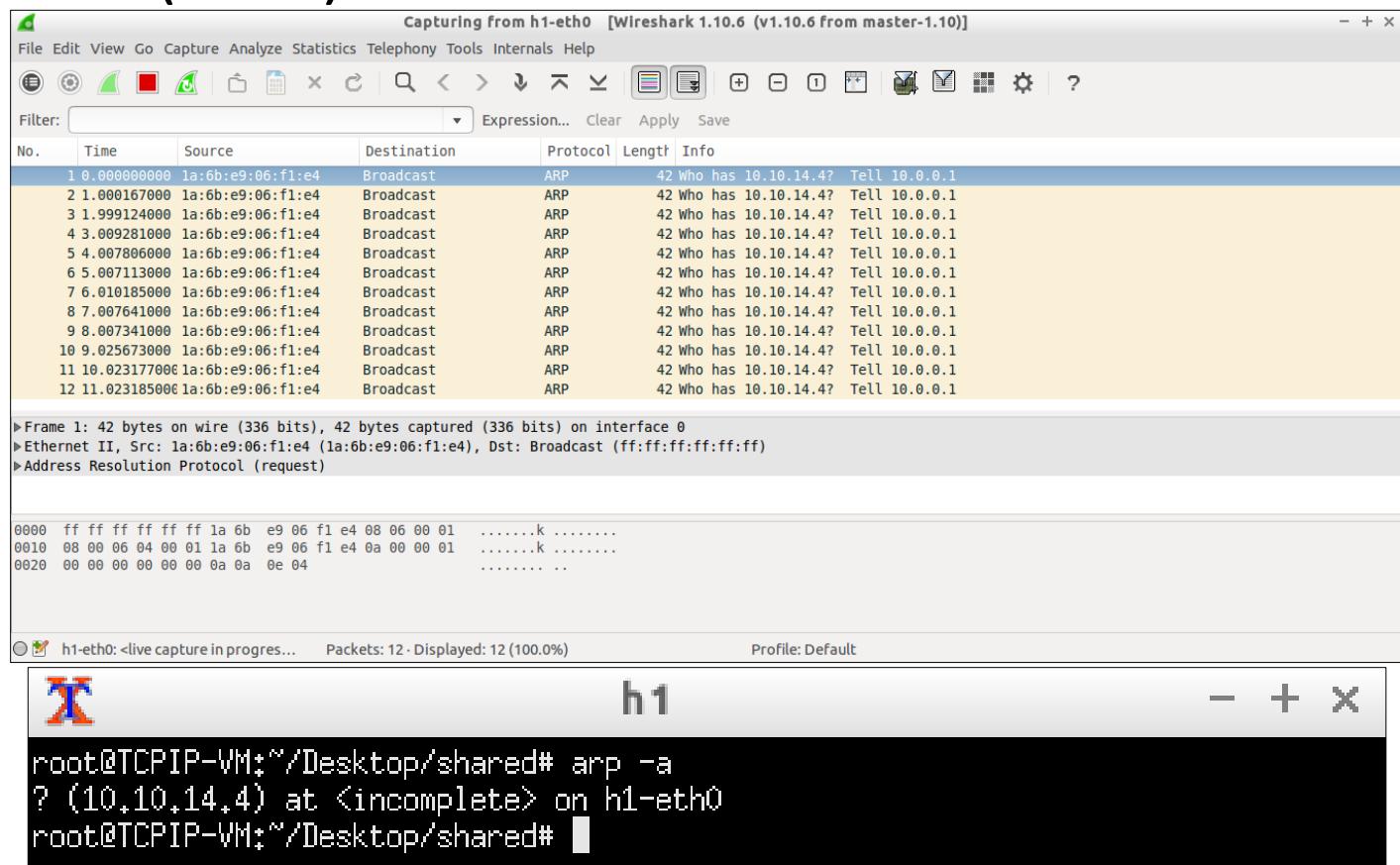


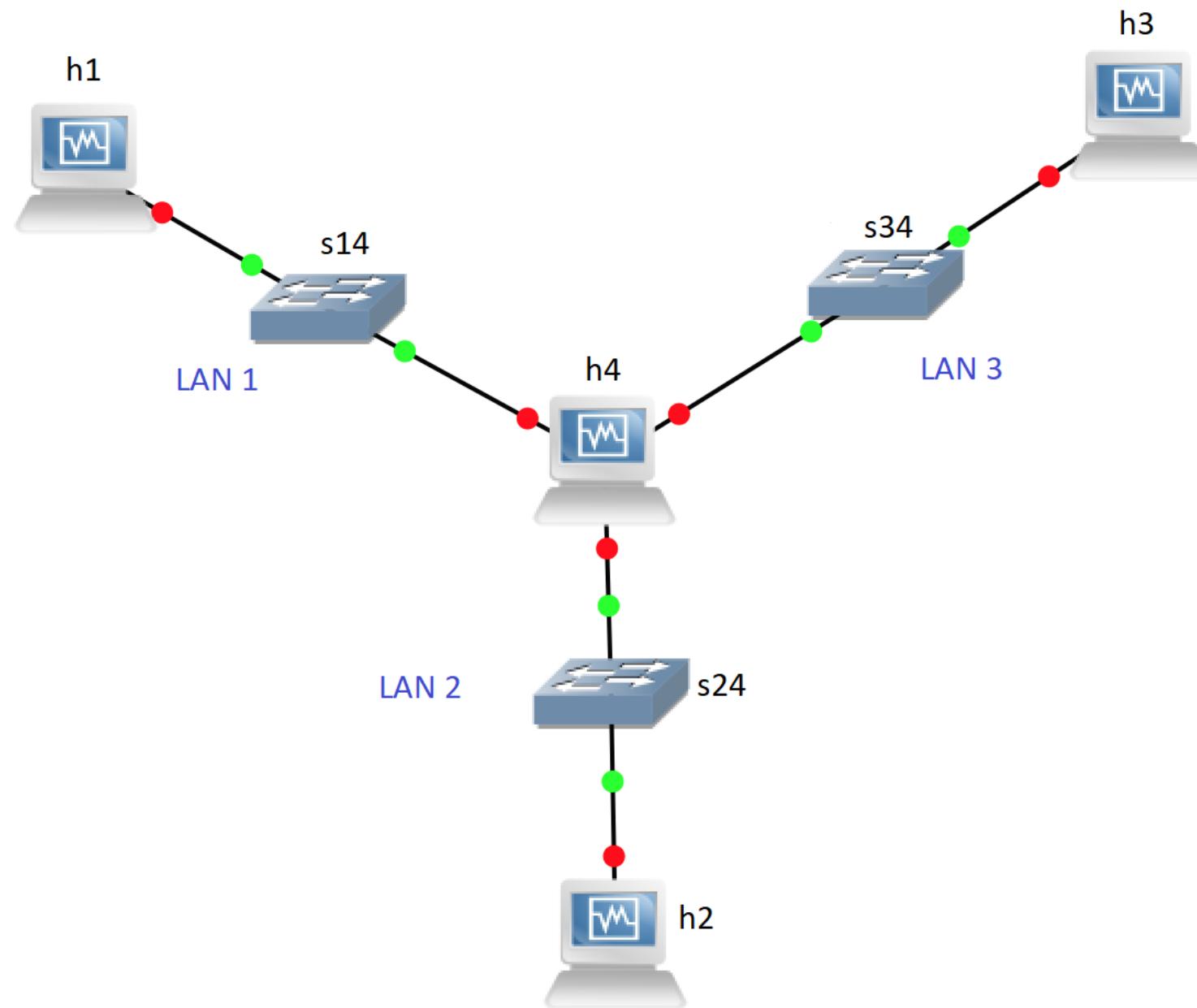
```
root@TCPIP-VM:~/Desktop/shared# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
88: h1-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    mode DEFAULT group default qlen 1000
        link/ether 96:b6:e3:c6:a5:48 brd ff:ff:ff:ff:ff:ff
root@TCPIP-VM:~/Desktop/shared#
```

- If an interface mode is DOWN, change it to UP, e.g. h1-eth0:
  - # ip link set h1-eth0 up
  - ping **x**

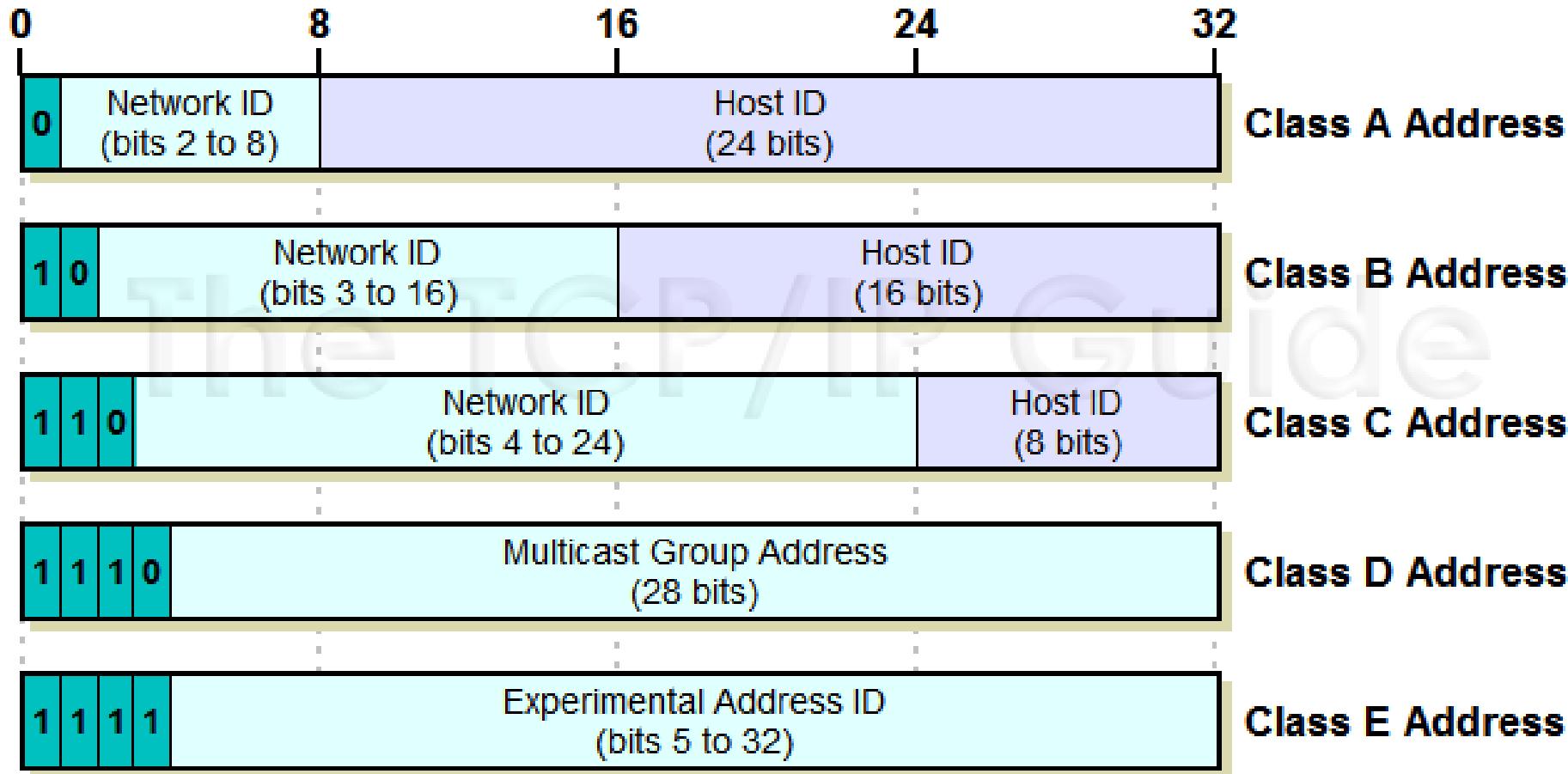
# ARP (Address Resolution Protocol)

- A procedure for mapping a dynamic IP address to a physical address, known as a media access control (MAC) address.
  - ARP request
  - ARP reply
- Open Wireshark on a host:
  - # wireshark &
- Show ARP table of a host:
  - # arp -a





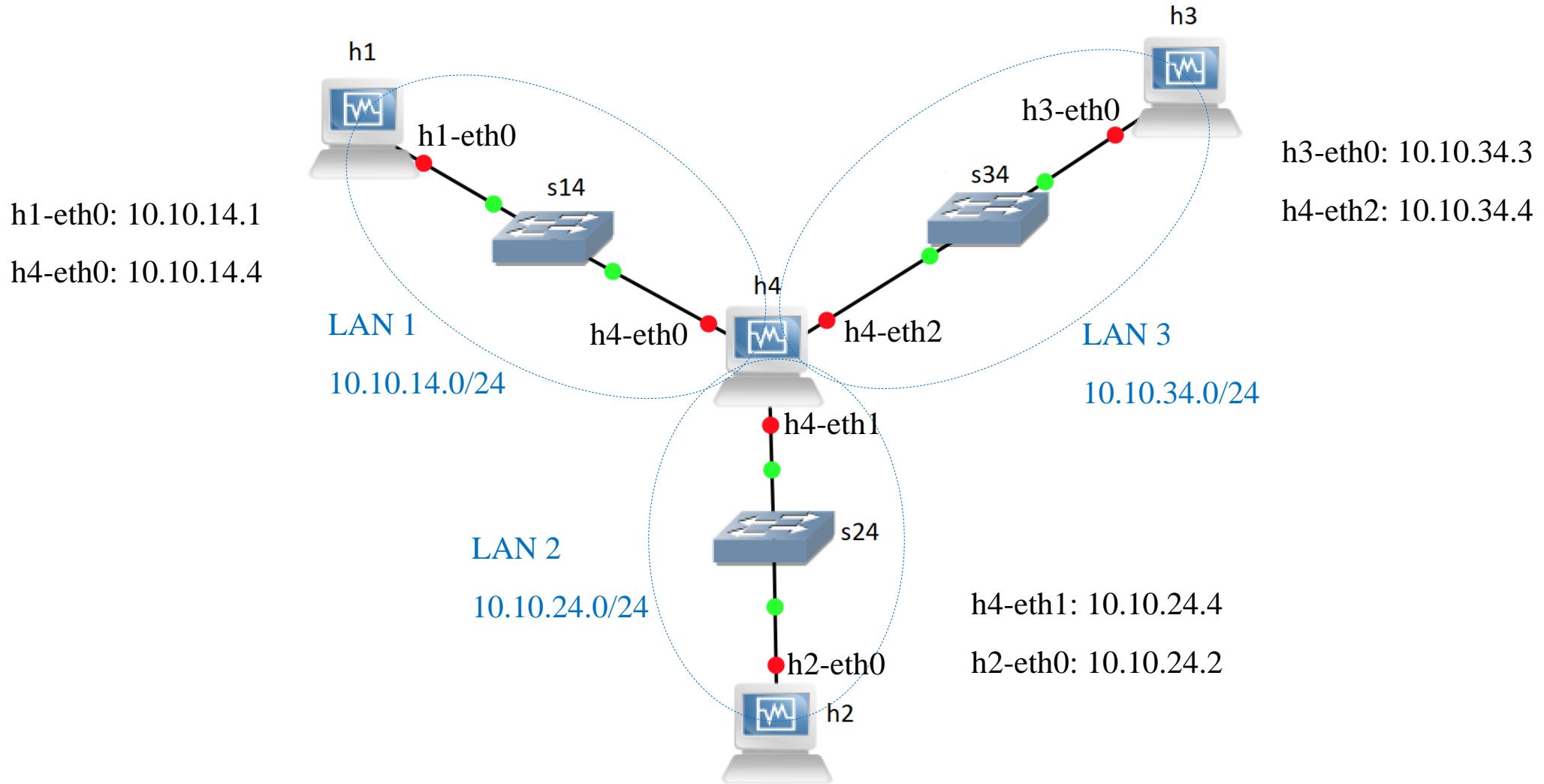
# IP Address Class Bit Assignments and Network/Host ID Sizes



# IP Address classes: Chart Representation

Address Classes	Range	Bit Pattern of 1 <sup>st</sup> byte	Decimal Range	Default Subnet Mask	Reserved for
A	1.0.0.0 to 127.255.255.255	0xxxxxx	1 to 127	255.0.0.0	Governments
B	128.0.0.0 to 191.255.255.255	10xxxxxx	128-191	255.255.0.0	Medium Companies
C	192.0.0.0 to 223.255.255.255	110xxxx	192-223	255.255.255.0	Small Companies
D	224.0.0.0 to 239.255.255.255	1110xxxx	224-239	Not Applicable	Reserved for Multicasting
E	240.0.0.0 to 255.255.255.255	11110xxx	240-255	Not Applicable	Experimental or future use

# 10.10.0.0/16



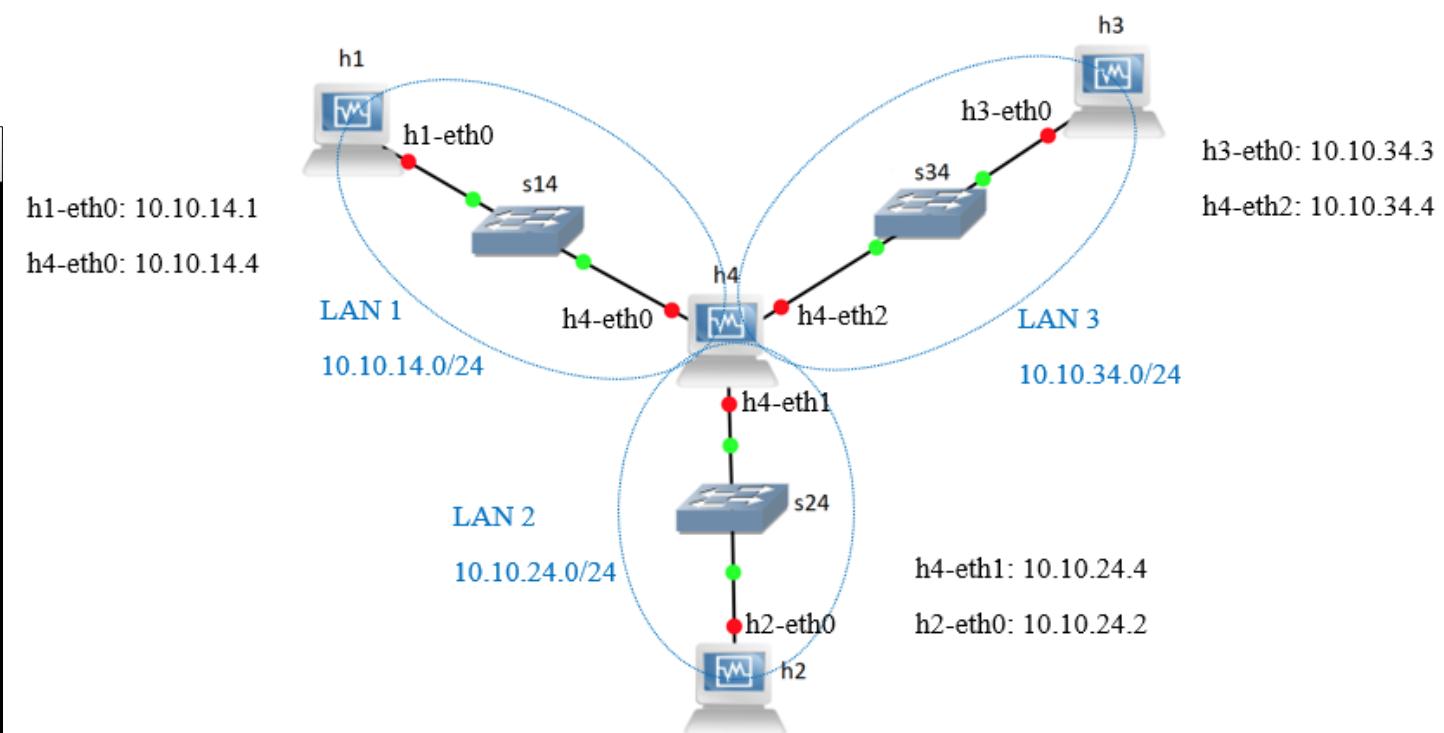
# Assign an IP address to an interface

- `# ip addr flush dev h1-eth0`
- `# ip addr add 10.10.14.1/24 dev h1-eth0`
- `# ifconfig -a`

```
h1
root@TCPiP-VM:~/Desktop/shared# ip addr flush dev h1-eth0
root@TCPiP-VM:~/Desktop/shared# ip addr add 10.10.14.1/24 dev h1-eth0
root@TCPiP-VM:~/Desktop/shared# ifconfig
h1-eth0  Link encap:Ethernet HWaddr 76:42:fd:85:98:43
          inet addr:10.10.14.1 Bcast:0.0.0.0 Mask:255.255.255.0
                        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                        RX packets:18 errors:0 dropped:0 overruns:0 frame:0
                        TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
                        collisions:0 txqueuelen:1000
                        RX bytes:1756 (1.7 KB) TX bytes:828 (828.0 B)

lo      Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
                        UP LOOPBACK RUNNING MTU:65536 Metric:1
                        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                        collisions:0 txqueuelen:0
                        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

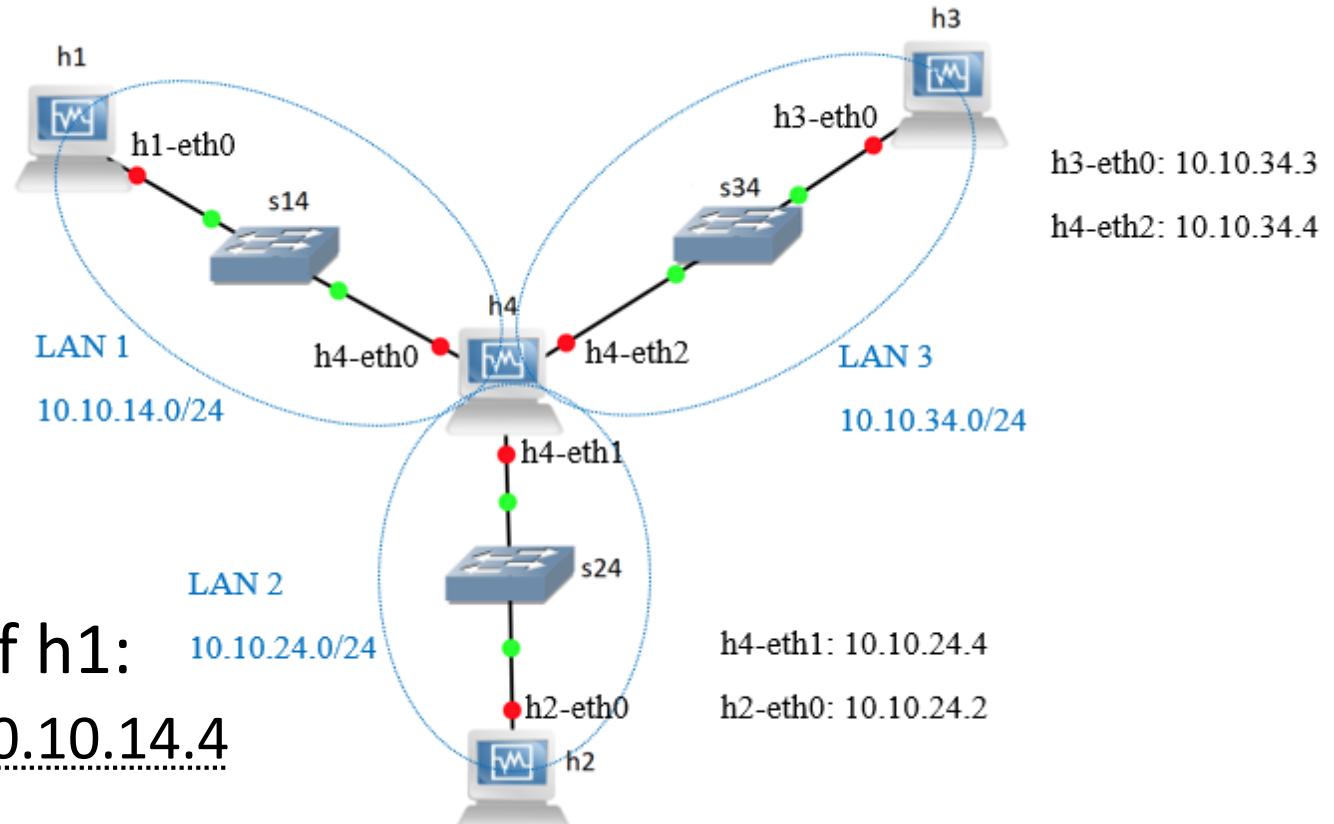
root@TCPiP-VM:~/Desktop/shared#
```



# Set gateway

h1-eth0: 10.10.14.1  
h4-eth0: 10.10.14.4

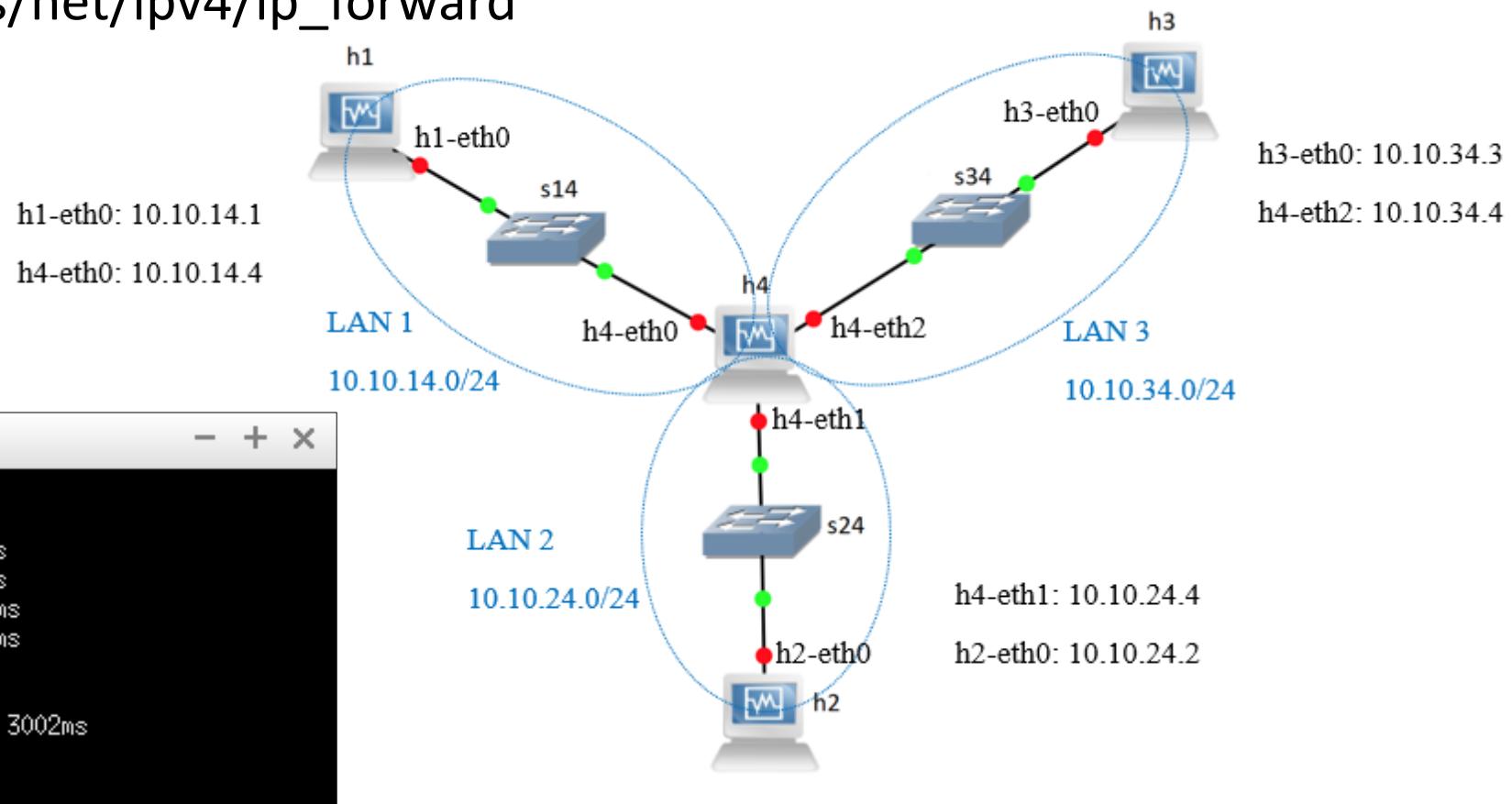
- Show routing table:
  - # ip route
- Add(Delete) default gateway of h1:
  - # ip route add(del) default via 10.10.14.4



```
h1
root@TCPiP-VM:~/Desktop/shared# ip route
10.10.14.0/24 dev h1-eth0 proto kernel scope link src 10.10.14.1
root@TCPiP-VM:~/Desktop/shared# ip route add default via 10.10.14.4
root@TCPiP-VM:~/Desktop/shared# ip route
default via 10.10.14.4 dev h1-eth0
10.10.14.0/24 dev h1-eth0 proto kernel scope link src 10.10.14.1
root@TCPiP-VM:~/Desktop/shared#
```

# Convert into router

- Convert h4 into a router:
  - `# echo 1 > /proc/sys/net/ipv4/ip_forward`



```
h1
root@TCPIP-VM:~/Desktop/shared# ping 10.10.24.2
PING 10.10.24.2 (10.10.24.2) 56(84) bytes of data.
64 bytes from 10.10.24.2: icmp_seq=1 ttl=63 time=7.85 ms
64 bytes from 10.10.24.2: icmp_seq=2 ttl=63 time=1.16 ms
64 bytes from 10.10.24.2: icmp_seq=3 ttl=63 time=0.116 ms
64 bytes from 10.10.24.2: icmp_seq=4 ttl=63 time=0.108 ms
^C
--- 10.10.24.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.108/2.310/7.850/3.227 ms
root@TCPIP-VM:~/Desktop/shared#
```

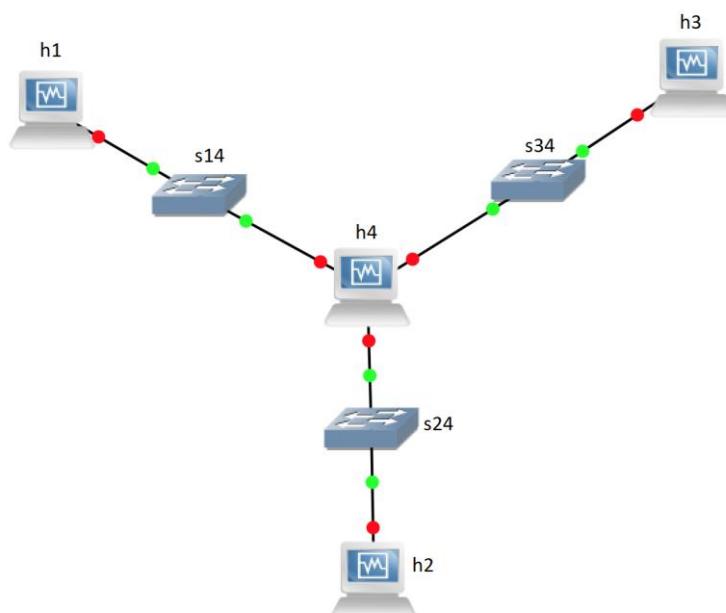
## آزمایشگاه شبکه

### آزمایش ۲: ایجاد ارتباطات در شبکه محلی (LAN)

#### الف) کابل‌کشی و برقراری اتصالات فیزیکی

مطابق شکل، در Mininet، یک پیکربندی متشكل از ۴ گره به نام‌های h1، h2، h3 و h4 تولید کنید که از طریق سه سوئیچ لایه ۲ به نام‌های s14، s24 و s34 بهم متصل شده‌اند. برای این منظور، شما می‌توانید از اسکریپت پایتون `lab2.py` به عنوان نقطه شروع استفاده کنید. بخش‌هایی از اسکریپت سازنده توپولوژی در این فایل از قبل نوشته شده است. پس از تکمیل، شما باید چهار پنجره ترمینال با عنوان‌ین h1، h2، h3 و h4 را ملاحظه نمایید. \*توجه! در این بخش، فعلاً فقط سوئیچ‌ها، گره‌ها و اینترفیس‌های آنها را ایجاد کنید. آدرس‌دهی به «رابطهای شبکه» (اینترفیس‌ها) در مراحل بعدی صورت خواهد گرفت.

- گره h1 دارای یک اینترفیس به نام h1-eth0 است که آن را به پورت s14-eth1 از سوئیچ s14 متصل کرده است.
- گره h2 دارای یک اینترفیس به نام h2-eth0 است که آن را به پورت s24-eth1 از سوئیچ s24 متصل کرده است.
- گره h3 دارای یک اینترفیس به نام h3-eth0 است که آن را به پورت s34-eth1 از سوئیچ s34 متصل کرده است.
- گره h4 دارای یک اینترفیس به نام h4-eth0 است که آن را به پورت s14-eth2 از سوئیچ s14 متصل کرده؛ هم‌چنین، یک اینترفیس به نام h4-eth1 دارد که آن را به پورت s24-eth2 از سوئیچ s24 متصل کرده و در نهایت اینکه، اینترفیس سوم گره h4 به نام h4-eth2 آن را به پورت s34-eth2 از سوئیچ s34 متصل کرده است.



## ب) بررسی برقراری اتصالات

- برای مشاهده اینترفیس‌های شبکه‌ای هر host، از دستور زیر استفاده کنید:

```
$ ip link
```

- اگر اینترفیس‌های یک host، در مُد UP نیست، با دستور زیر آن را بالا بیاورید (مثالاً برای h4):

```
# ip link set h4-eth0 up
# ip link set h4-eth1 up
# ip link set h4-eth2 up
```

- به طور مشابه، اینترفیس‌های سایر host‌ها را به مُد UP ببرید.

- همانطور که می‌دانید از برنامه WireShark می‌توانید برای capture کردن ترافیک ورودی و خروجی روی یک اینترفیس شبکه استفاده کنید. به همین منظور، یک برنامه WireShark باز کرده و روی اینترفیس مربوط به گوش دهید.

- ابتدا ملاحظه خواهیم کرد که هنگام ping کردن آدرس‌های منتبش‌نشده چه روی می‌دهد. برای تست کردن اتصال IPv4، یک دستور ping از h1 به مقصد h4 صادر کنید.

```
$ ping 10.10.14.4
```

سؤال 1- توضیح دهید چه اتفاقی می‌افتد (\*راهنمایی: آیا اساساً پیام ICMP request ارسال می‌شود؟ آیا جدول ARP برای آدرس MAC صحیح برای آدرس IP نظیر h4 یعنی 10.10.14.4 است؟ دستور arp-a محتوای جدول ARP reply را نشان می‌دهد. آیا ARP request ارسال می‌شود؟ ARP چطور؟).

---

## ج) پیکربندی و تست LAN

- اولین گام برای دستیابی به اتصال کامل، پیکربندی اینترفیس‌ها و انتساب آدرس IP به آنهاست.
- برای آدرس دهی subnet‌ها، ما از فضای آدرس 10.10.0.0/16 استفاده می‌نماییم.
- برای شبکه محلی متصل‌کننده h4 به h1 (یعنی LAN1)، ما از پیشوند آدرس 10.10.14.0/24 استفاده می‌کنیم. بایت چهارم شناسه IP ویژه h1 برابر با 1 و برای h4 برابر با 4 قرار داده می‌شود.
- برای شبکه محلی متصل‌کننده h4 به h2 (یعنی LAN2)، ما از پیشوند آدرس 10.10.24.0/24 استفاده می‌کنیم. بایت چهارم شناسه IP ویژه h2 برابر با 2 و برای h4 برابر با 4 قرار داده می‌شود.
- برای شبکه محلی متصل‌کننده h4 به h3 (یعنی LAN3)، ما از پیشوند آدرس 10.10.34.0/24 استفاده می‌کنیم. بایت چهارم شناسه IP ویژه h3 برابر با 3 و برای h4 برابر با 4 قرار داده می‌شود.

- بنابراین، به طور خلاصه، آدرس‌های IPv4 مورد استفاده برای هر اینترفیس (رابط شبکه‌ای) به شرح صفحه بعد خواهد بود:

**h1-eth0:** 10.10.14.1    **h2-eth0:** 10.10.24.2    **h3-eth0:** 10.10.34.3  
**h4-eth0:** 10.10.14.4    **h4-eth1:** 10.10.24.4    **h4-eth2:** 10.10.34.4

- به عنوان نمونه، جهت آدرس دهی به اینترفیس h1-eth0 از دستور زیر استفاده نمایید:

```
# ip addr add 10.10.14.1/24 dev h1-eth0
```

- \*نکته: برای اینترفیس h1-eth0 IP آدرس 10.0.0.1 پیش‌فرض قبلی (10.0.0.1) را با دستور ip addr flush حذف کنید.

- حال، مجدداً از h1، یک ping دیگر به مقصد اینترفیس h4 با IP 10.10.14.4 انجام دهید:

**سؤال ۲ - مشاهده خود را بیان کنید. آیا h1-eth0 روی WireShark بسته ARP reply را شنود کرده است؟ ICMP reply چطور؟**

- تا اینجا باید در هر LAN، اتصال IPv4 برقرار شده باشد.

#### ۵) مسیریابی بسته‌ها

- سعی کنید از h1 به هریک از اینترفیس‌های h4-eth1 و h4-eth2 برسید:

```
$ ping 10.10.24.4
$ ping 10.10.34.4
```

**سؤال ۳ - آیا کار می‌کند؟ چرا؟**

- برای مشاهده جدول مسیریابی IPv4، در h1 از دستور زیر استفاده نمایید:

```
$ ip route
```

**سؤال ۴ - مسیریابی مندرج در جدول مسیریابی را نوشته و آنها را توضیح دهید.**

- در گره h1، با تعریف default gateway، یک مسیر پیش‌فرض IPv4 اضافه نمایید:

```
# ip route add default via 10.10.14.4
```

**سؤال ۵ - مجدداً تست کنید که آیا اینترفیس h4-eth2 از h1 از طریق h4 قابل دسترسی است؟!**

- در h4، آن گوش کنید تا ترافیک لینک بین hostهای LAN3 را مشاهده نمایید.

- از h1، سعی کنید تا h3 را ping کرده و ترافیک مربوطه را روی h4 مشاهده کنید:

```
§ ping 10.10.34.3
```

- ملاحظه خواهید نمود که همچنان اتصال برقرار نیست! پیام‌های ICMP توسط h4 به بیرون ارسال نمی‌شوند.
- در واقع، بطور پیش‌فرض، h4 ترافیک IP را از یک LAN به دیگری forward نمی‌کند. برای این منظور، باید قابلیت IP forwarding را با تایپ دستور زیر روی h4 فعال‌سازی نمایید:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- مجدداً تلاش کنید تا h3 را از طریق h1 پینگ نمایید. باز هم کار نمی‌کند، اما حالا می‌دانید که چگونه این مشکل را رفع کنید (راهنمایی: مجدداً ترافیک روی این دو لینک را از طریق Wireshark ملاحظه کنید و ببینید چه بسته‌هایی به بیرون ارسال نمی‌شوند).

## سؤال ۶- برای رفع مشکل، از چه دستوراتی استفاده می‌کنید؟ روی کدام host

- به عنوان گام آخر، شبکه خود را طوری پیکربندی نمایید تا قادر باشد h1، h2 و h3 را از طریق هم‌دیگر ping کنید.

## سؤال ۷- این ping‌ها را انجام دهید:

h1 از طریق h2

h3 از طریق h2 و

h1 از طریق h3

نظر خود را در ارتباط با مقادیر RTT مورد مشاهده بیان نمایید.

به نام خدا

## گزارشکار آزمایش دوم آزمایشگاه شبکه های کامپیوتری: ایجاد ارتباطات در شبکه محلی (LAN)

سیده شکیبا انارکی فیروز - 99442047

بهاره کاووسی نژاد - 99431217

1 - کد پس از تکمیل:

```
#!/usr/bin/python

"""
This example shows how to create a Mininet object and add nodes to it manually.
"""

"Importing Libraries"
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

#Function definition: This is called from the main function
def firstNetwork():

    "Create an empty network and add nodes to it."
    net = Mininet()
    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1' )
    h2 = net.addHost( 'h2' )
    h3 = net.addHost( 'h3' )
    h4 = net.addHost( 'h4' )

    info( '*** Adding switch\n' )
    s14 = net.addSwitch( 's14' )
    s24 = net.addSwitch( 's24' )
    s34 = net.addSwitch( 's34' )

    info( '*** Creating links\n' )
    net.addLink( h1, s14 )
    net.addLink( h2, s24 )
    net.addLink( h3, s34 )
    net.addLink( h4, s14 )
    net.addLink( h4, s24 )
    net.addLink( h4, s34 )

    info( '*** Starting network\n' )
    net.start()

    #This is used to run commands on the hosts
    info( '*** Starting terminals on hosts\n' )
    h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
    h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')
    h3.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h3 &')
    h4.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h4 &')

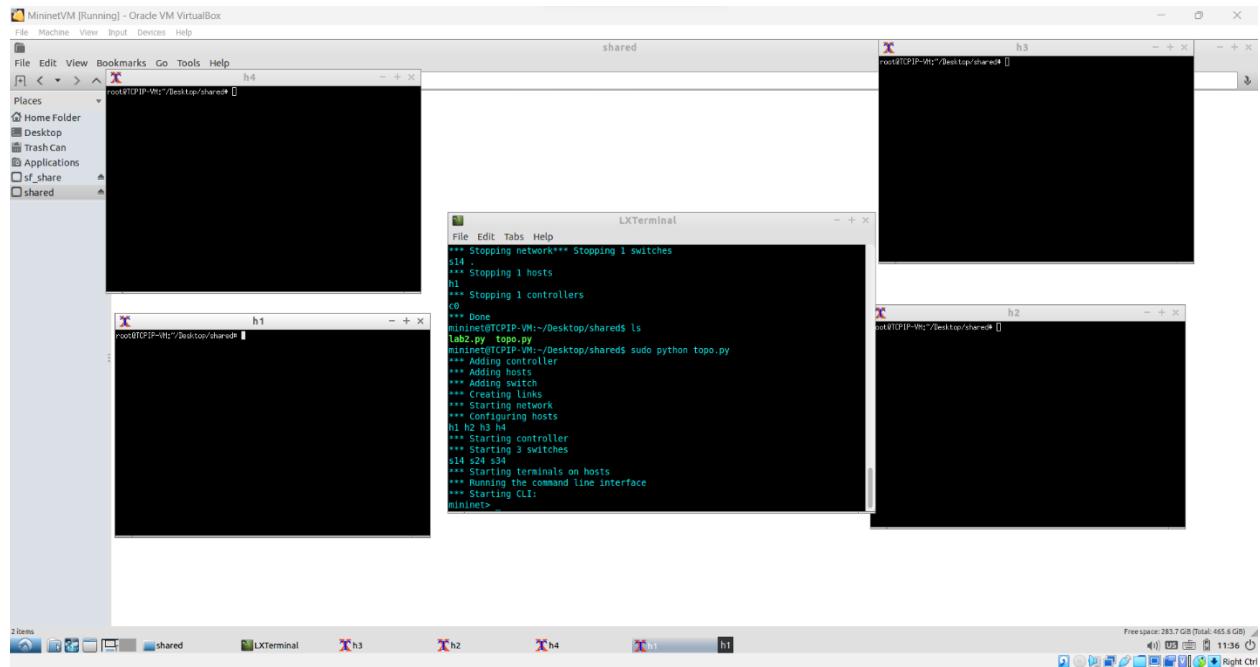
    info( '*** Running the command line interface\n' )
    CLI( net )

    info( '*** Closing the terminals on the hosts\n' )
    h1.cmd("killall xterm")
    h2.cmd("killall xterm")
    h3.cmd("killall xterm")
    h4.cmd("killall xterm")

    info( '*** Stopping network' )
    net.stop()

#main Function: This is called when the Python file is run
if __name__ == '__main__':
    setLogLevel( 'info' )
    firstNetwork()
```

پنجره های ترمینال باز شده پس از ران کردن کد:

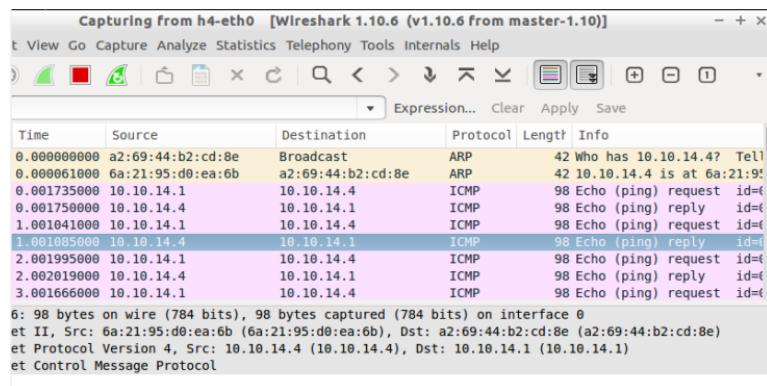


بدون دادن IP به Interface h1 نمی‌توان بسته‌ای توسط این لینک ارسال نمود؛ زیرا h1 باید آدرس مبدا خود را در بسته قرار دهد و همچنین h1 با بررسی آدرس مقصد آن را با subnet خود تطبیق دهد (که بسته باید برای Gateway ارسال شود یا مقصد بسته در همین h1 است).

پیام‌های ARP درخواستی ارسال می‌شوند اما پاسخی برای آنها دریافت نمی‌شود. زیرا سویچ این IP را نمی‌شناسد. در نتیجه جدول‌های ARP در h1 مقدار مقدار IP mac address وارد شده را به عنوان incomplete در نظر می‌گیرد.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
2	1.000046000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
3	1.999823000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
4	2.999866000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
5	4.000010000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
6	4.999968000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1

– هم بسته ICMP reply و هم بسته ARP reply شنود می‌شوند.



3 – خیر، به هیچکدام دسترسی نداریم زیرا **h4** باید برای ICMP forward کردن پیام‌های **reachable** تنظیم شود و از آنجایی که دسترسی امکان‌پذیر نیست، خطای **unreachable** را نمایش می‌دهد. در نتیجه ابتدا باید **h4** تنظیم شوند تا **h1** بتواند با آنها ارتباط برقرار کند و پس از آن دستور **ip route** را اجرا می‌کنیم.

entry – 4

- از این **config** برای ارسال هر IP‌ای که **entry** برای آن وجود نداشت استفاده می‌شود. به عنوان مثال از طریف **interface eth0** انتقال و برای IP **10.10.14.4** برود.
- از این **config** برای ارسال هر IP‌ای که در شبکه‌ی گفته شده وجود دارد استفاده می‌شود. به عنوان مثال از **interface eth0** ارسال شود.

5 – تمامی پیام‌هایی که باید به بیرون از **subnet** هدایت شوند به سمت **h4** فرستاده می‌شوند؛ زیرا **h1** host می‌داند که **Gateway** آن **h4** است بنابراین این پیام به مقصد رسیده و پاسخ آن باز می‌گردد. به عبارت دیگر، این بار **interface eth2** از طریق **h1** قابل دسترسی می‌شوند.

No.	Time	Source	Destination	Protocol	Length	Info
13	556.33476408	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=1/256, ttl=64 (reply in 14)
14	556.33567308	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=1/256, ttl=64 (request in 13)
15	557.33632508	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=2/512, ttl=64
16	557.33708408	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=2/512, ttl=64 (request in 15)
17	558.33857608	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=3/768, ttl=64 (reply in 18)
18	558.33685908	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=3/768, ttl=64 (request in 17)
19	559.33659008	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=4/1024, ttl=64
20	559.33663408	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=4/1024, ttl=64 (request in 19)
21	560.33666008	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=5/1280, ttl=64 (reply in 22)
22	560.33670908	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=5/1280, ttl=64 (request in 21)

6 – با توجه به اینکه پیام‌ها نمی‌توانند از **LAN3** خارج شوند، بنابراین باید یک **gateway** در IP **10.10.34.4** در **host h3** تعیین کنیم تا بتوانیم بسته‌های پاسخ را به خارج از **LAN3** بفرستیم. برای رفع این مشکل، باید بر بروی **host h3** دستور زیر را اجرا کنیم:

**ip route add default via 10.10.34.4**

همچنین مانند **h1** باید پیشفرض قبلی با دستور زیر حذف شود:

**ip addr del 10.0.0.3/8 dev h3-eth0**

7 – باید **h2** را برای **LAN2** در **h2** مشخص کنیم و سپس دستور **ping** می‌تواند برای هر سه **host** اجرا شود. از آنجایی که این شبکه متقاضن می‌باشد، اختلاف RTT‌ها بسیار کم است و RTT‌ها تقریباً مشابه به دست می‌آیند. تمامی مقادیر در حدود 0.2ms تا حدود 1 ثانیه هستند و نتیجه حدودی این است که فاصله این نودها از هم تقریباً برابر است.

```
PING 10.10.14.1 (10.10.14.1) 56(84) bytes of data.
64 bytes from 10.10.14.1: icmp_seq=1 ttl=63 time=8.82 ms
64 bytes from 10.10.14.1: icmp_seq=2 ttl=63 time=3.11 ms
64 bytes from 10.10.14.1: icmp_seq=3 ttl=63 time=0.153 ms
64 bytes from 10.10.14.1: icmp_seq=4 ttl=63 time=0.150 ms
64 bytes from 10.10.14.1: icmp_seq=5 ttl=63 time=0.150 ms

PING 10.10.24.2 (10.10.24.2) 56(84) bytes of data.
64 bytes from 10.10.24.2: icmp_seq=1 ttl=63 time=12.0 ms
64 bytes from 10.10.24.2: icmp_seq=2 ttl=63 time=4.54 ms
64 bytes from 10.10.24.2: icmp_seq=3 ttl=63 time=0.665 ms
64 bytes from 10.10.24.2: icmp_seq=4 ttl=63 time=0.335 ms
64 bytes from 10.10.24.2: icmp_seq=5 ttl=63 time=0.153 ms
```

```
PING 10.10.34.3 (10.10.34.3) 56(84) bytes of data.  
64 bytes from 10.10.34.3: icmp_seq=1 ttl=63 time=2.19 ms  
64 bytes from 10.10.34.3: icmp_seq=2 ttl=63 time=0.145 ms  
64 bytes from 10.10.34.3: icmp_seq=3 ttl=63 time=0.127 ms  
64 bytes from 10.10.34.3: icmp_seq=4 ttl=63 time=0.144 ms  
64 bytes from 10.10.34.3: icmp_seq=5 ttl=63 time=0.161 ms
```



دانشگاه علم و صنعت ایران

دانشکده مهندسی کامپیوتر  
آزمایشگاه شبکه های کامپیوتری

---

تمرین سری دوم

---

پوریا رحیمی (۹۹۵۲۱۲۸۹)  
بکتاش انصاری (۹۹۵۲۱۰۸۲)

نیم سال اول  
۱۴۰۳-۱۴۰۲

الف) کابل کشی و برقراری اتصالات فیزیکی :

```
1 #!/usr/bin/python
2
3 """
4 This example shows how to create a Mininet object and add nodes to it manually.
5 """
6 "Importing Libraries"
7 from mininet.net import Mininet
8 from mininet.node import Controller
9 from mininet.cli import CLI
10 from mininet.log import setLogLevel, info
11
12 #Function definition: This is called from the main function
13 def firstNetwork():
14
15     "Create an empty network and add nodes to it."
16     net = Mininet()
17     info( '*** Adding controller\n' )
18     net.addController( 'c0' )
19
20     info( '*** Adding hosts\n' )
21     h1 = net.addHost( 'h1' )
22     h2 = net.addHost( 'h2' )
23     h3 = net.addHost( 'h3' )
24     h4 = net.addHost( 'h4' )
25
26     info( '*** Adding switch\n' )
27     s14 = net.addSwitch( 's14' )
28     s24 = net.addSwitch( 's24' )
29     s34 = net.addSwitch( 's34' )
30
31     info( '*** Creating links\n' )
32     net.addLink( h1, s14 )
33     net.addLink( h2, s24 )
34     net.addLink( h3, s34 )
35     net.addLink( h4, s14 )
36     net.addLink( h4, s24 )
37     net.addLink( h4, s34 )
```

## الف) کابل کشی و برقراری اتصالات فیزیکی :

```
39     info( '*** Starting network\n' )
40     net.start()
41
42
43     #This is used to run commands on the hosts
44     info( '*** Starting terminals on hosts\n' )
45     h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
46     h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')
47     h3.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h3 &')
48     h4.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h4 &')
49
50     info( '*** Running the command line interface\n' )
51     CLI( net )
52
53     info( '*** Closing the terminals on the hosts\n' )
54     h1.cmd("killall xterm")
55     h2.cmd("killall xterm")
56     h3.cmd("killall xterm")
57     h4.cmd("killall xterm")
58
59     info( '*** Stopping network' )
60     net.stop()
61
62 #main Function: This is called when the Python file is run
63 if __name__ == '__main__':
64     setLogLevel( 'info' )
65     firstNetwork()
66
```

سوال 1- توضیح دهد چه اتفاقی می افتد (\*راهنمایی: آیا اساساً پیام ICMP request ارسال می شود؟ آیا جدول ARP برای h1 دارای آدرس MAC صحیح برای آدرس IP نظیر h4 یعنی 10.10.14.4 است؟ دستور arp-a محتوای جدول ARP را نشان می دهد. آیا ARP request ارسال می شود؟ ARP reply چطور؟).

پاسخ :

از آن جایی که ما آدرس فیزیکی 10.10.14.4 را نمی دانیم ، هیچ پیام ICMP ارسال نمی شود. جدول arp در h1 نشان می دهد که نتوانستیم آدرس فیزیکی مناسب به 10.10.14.4 را به دست بیاوریم. در ابتدا ARP request فرستاده می شود و از آنجایی که چنین IP ای در شبکه موجود نیست، کسی پاسخی نمی دهد. یعنی ARP reply ارسال نمی شود و دریافت نیز نمی شود.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
2	1.000046000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
3	1.999823000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
4	2.999866000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
5	4.000010000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1
6	4.999968000	ae:8c:42:e0:dd:98	Broadcast	ARP	42	Who has 10.10.14.4? Tell 10.0.0.1

سوال 2- مشاهده خود را بیان کنید. آیا WireShark روی h1-eth0، در رابطه با اجرای دستور ping فوق، بسته ARP reply را شنود کرده است؟ ICMP reply چطور؟

پاسخ :

بله ، هر دو بسته شنود خواهند شد.

ID.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.14.1	10.10.14.4	ICMP	98	Echo (ping) request id=0x1255, seq=1/256, ttl=64
2	0.001189000	10.10.14.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1255, seq=1/256, ttl=64 (request in 1)
3	1.001628000	10.10.14.1	10.10.14.4	ICMP	98	Echo (ping) request id=0x1255, seq=2/512, ttl=64 (reply in 4)
4	1.002392000	10.10.14.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1255, seq=2/512, ttl=64 (request in 3)
5	2.000862000	10.10.14.1	10.10.14.4	ICMP	98	Echo (ping) request id=0x1255, seq=3/768, ttl=64
6	2.001077000	10.10.14.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1255, seq=3/768, ttl=64 (request in 5)
7	3.000449000	10.10.14.1	10.10.14.4	ICMP	98	Echo (ping) request id=0x1255, seq=4/1024, ttl=64
8	3.000488000	10.10.14.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1255, seq=4/1024, ttl=64 (request in 7)
9	4.000648000	10.10.14.1	10.10.14.4	ICMP	98	Echo (ping) request id=0x1255, seq=5/1280, ttl=64
10	4.000695000	10.10.14.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1255, seq=5/1280, ttl=64 (request in 9)
11	5.017470000	32:0c:b7:98:1a:15	4a:55:c5:f6:69:70	ARP	42	Who has 10.10.14.1? Tell 10.10.14.4
12	5.017499000	4a:55:c5:f6:69:70	32:0c:b7:98:1a:15	ARP	42	10.10.14.1 is at 4a:55:c5:f6:69:70

پاسخ :

خیر، به هیچکدام از آن ها دسترسی نداریم زیرا همانطور که مشاهده می کنیم این دستور کار نمی کند زیرا h4 باید تنظیم شود که پیام های ICMP را forward کند و از آن جایی که به آن ها دسترسی نداریم به ما خطای network in unreachable را نمایش می دهد.

در نتیجه ما باید در ابتدا اینترفیس های h4 را تنظیم کنیم تا h1 بتواند با آن ها ارتباط برقرار کند. سپس دستور ip route را اجرا می کنیم.

## سوال 4

پاسخ :

در اینجا دو entry داریم، یک default دیگری 10.10.14.0/24 هر کدام از این entry ها به معنای زیر می باشند :

به این معنی می باشد که هر IP ای که برای آن entry ای وجود نداشت از این کانفیگ برای ارسالش استفاده شود. ( مثلا از طریق اینترفیس eth0 انتقال و برای IP، 10.10.14.4 برود).

10.10.14.0/24 : به این معنی می باشد که هر IP ای که در شبکه ی گفته شده وجود دارد از این کانفیگ برای ارسالش استفاده شود.(مثلا از اینترفیس eth0 ارسال بشود).

## سوال 5-

پاسخ :

همانطور که مشاهده می شود این بار اینترفیس h1 از طریق h4 از طریق eth2 قابل دسترسی می باشد در نتیجه تمامی پکت ها به درستی ارسال می شوند.

No.	Time	Source	Destination	Protocol	Length	Info
13	556.33476408	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=1/256, ttl=64 (reply in 14)
14	556.33567308	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=1/256, ttl=64 (request in 13)
15	557.33632508	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=2/512, ttl=64
16	557.33708408	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=2/512, ttl=64 (request in 15)
17	558.33657608	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=3/768, ttl=64 (reply in 18)
18	558.33685908	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=3/768, ttl=64 (request in 17)
19	559.33659008	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=4/1024, ttl=64
20	559.33663408	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=4/1024, ttl=64 (request in 19)
21	560.33666008	10.10.14.1	10.10.24.4	ICMP	98	Echo (ping) request id=0x1269, seq=5/1280, ttl=64 (reply in 22)
22	560.33670908	10.10.24.4	10.10.14.1	ICMP	98	Echo (ping) reply id=0x1269, seq=5/1280, ttl=64 (request in 21)

## سوال 6-

پاسخ :

با توجه با اینکه پیام ها نمی توانند از LAN3 خارج بشوند بنابراین باید یک gateway در 10.10.34.4 در h3 تعریف کنیم تا بتوانیم بسته های ریپلای را به خارج از LAN3 بفرستیم. برای رفع این مشکل، باید بر روی host h3 دستور ip route add default via 10.10.34.4 را اجرا کنیم.

## سوال 7-

پاسخ :

کافی است gateway را برای LAN2 در h2 مشخص کنیم و سپس هر 3 host ما می توانند همدیگر را ping کنند.

و از آن جایی که شبکه ما متقارن می باشد ، اختلاف RTT ها بسیار کم می باشد و می توان گفت RTT تقریبا مشابهی بدست می آوریم.

```
PING 10.10.14.1 (10.10.14.1) 56(84) bytes of data.  
64 bytes from 10.10.14.1: icmp_seq=1 ttl=63 time=8.82 ms  
64 bytes from 10.10.14.1: icmp_seq=2 ttl=63 time=3.11 ms  
64 bytes from 10.10.14.1: icmp_seq=3 ttl=63 time=0.153 ms  
64 bytes from 10.10.14.1: icmp_seq=4 ttl=63 time=0.150 ms  
64 bytes from 10.10.14.1: icmp_seq=5 ttl=63 time=0.150 ms
```

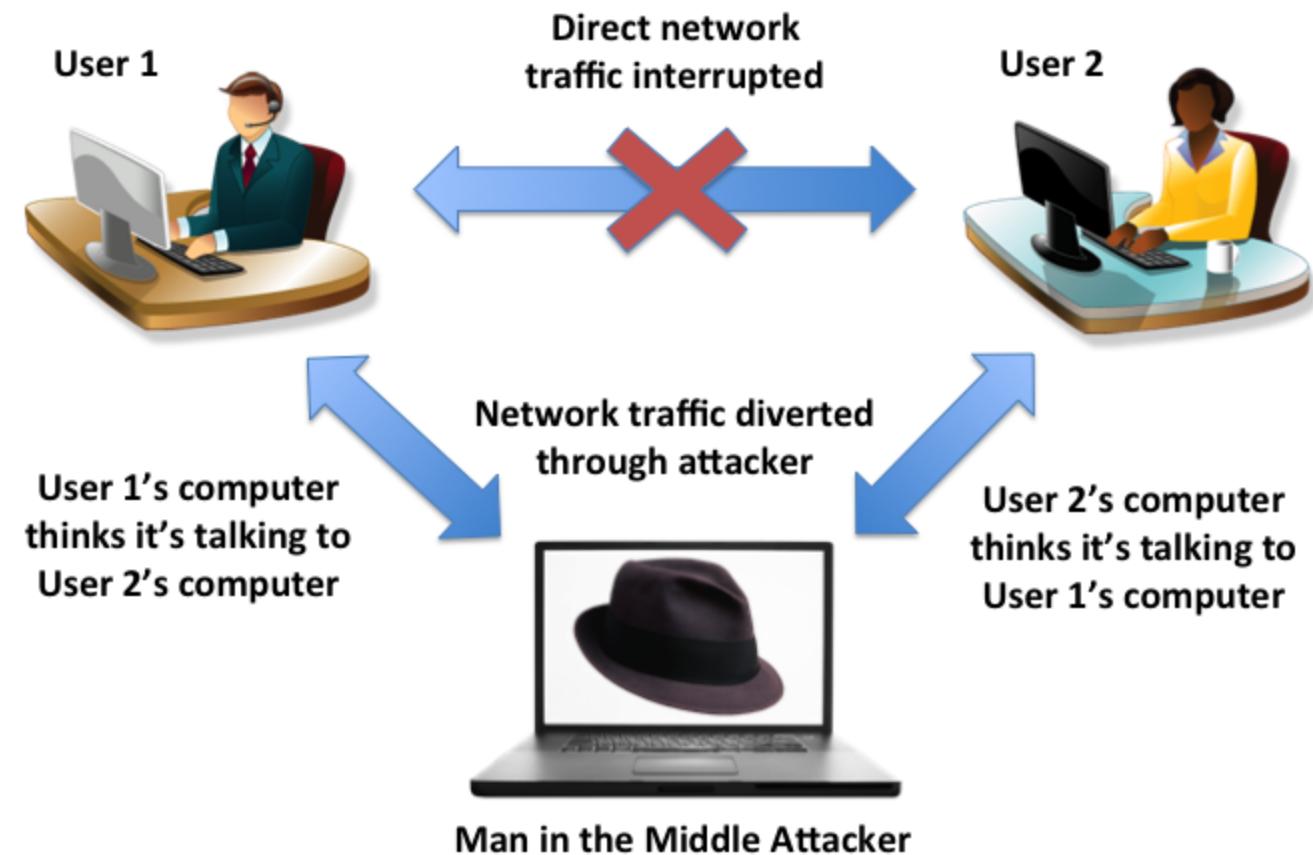
```
PING 10.10.24.2 (10.10.24.2) 56(84) bytes of data.  
64 bytes from 10.10.24.2: icmp_seq=1 ttl=63 time=12.0 ms  
64 bytes from 10.10.24.2: icmp_seq=2 ttl=63 time=4.54 ms  
64 bytes from 10.10.24.2: icmp_seq=3 ttl=63 time=0.665 ms  
64 bytes from 10.10.24.2: icmp_seq=4 ttl=63 time=0.335 ms  
64 bytes from 10.10.24.2: icmp_seq=5 ttl=63 time=0.153 ms
```

```
PING 10.10.34.3 (10.10.34.3) 56(84) bytes of data.  
64 bytes from 10.10.34.3: icmp_seq=1 ttl=63 time=2.19 ms  
64 bytes from 10.10.34.3: icmp_seq=2 ttl=63 time=0.145 ms  
64 bytes from 10.10.34.3: icmp_seq=3 ttl=63 time=0.127 ms  
64 bytes from 10.10.34.3: icmp_seq=4 ttl=63 time=0.144 ms  
64 bytes from 10.10.34.3: icmp_seq=5 ttl=63 time=0.161 ms
```

# Man-in-the-Middle Attack

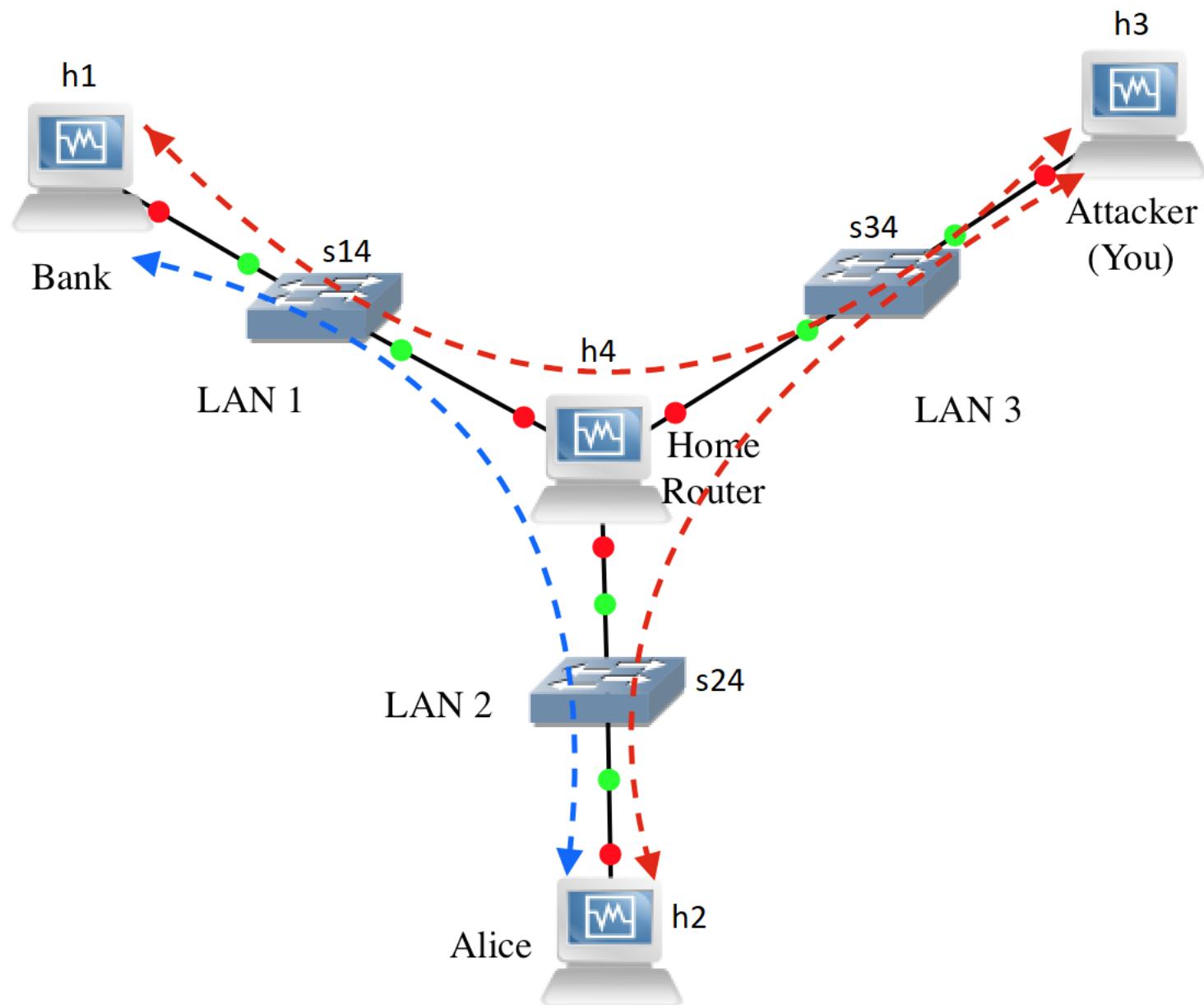
# Man-in-the-middle (MITM) Attack

Man-in-the-middle attack (MITM) are a common type of cybersecurity attacks that allow attackers to eavesdrop on the communication between two targets. The attack takes place in between two legitimately communicating hosts, allowing the attacker to “listen” to a conversation they should normally not be able to listen to, hence the name “man-in-the-middle.”



# IP spoofing

- IP spoofing is a method adopted by attackers to send forged address in their attack traffic:
  - i.e., they can send an IP packet with an IP address of their wish!
- Most of the times, spoofing is used by an attacker mainly for the following reasons:
  - To conduct a DDoS (Distributed Denial of Service) attack, and he does not want the response from the target machine to reach him.
  - To compromise source-based authentication.

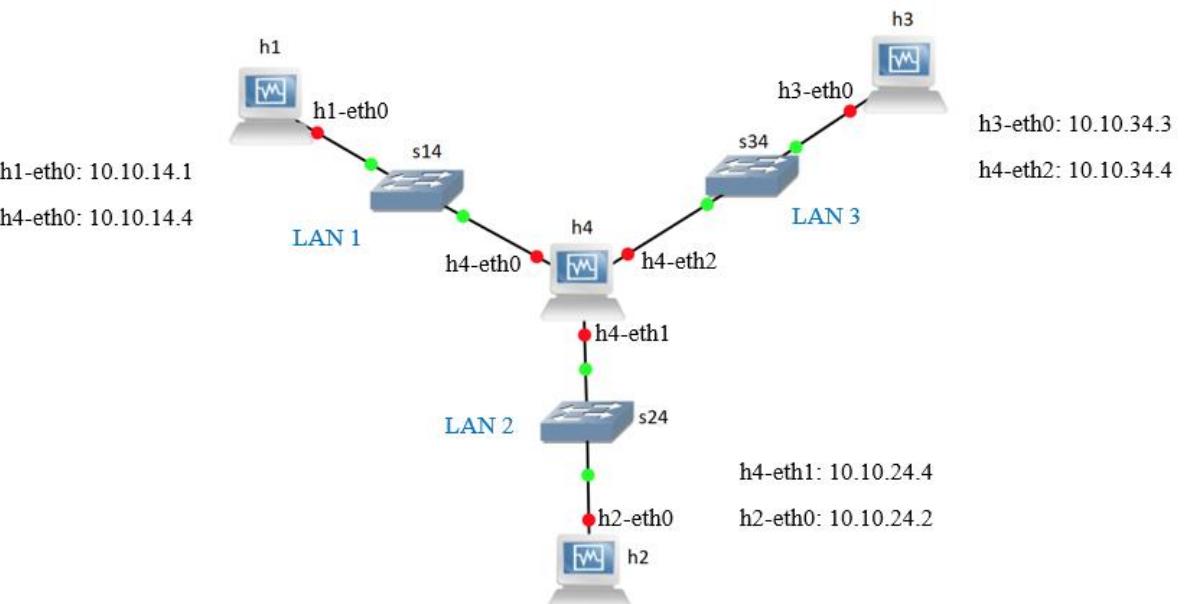


```

12 "Function definition: This is called from the main function"
13
14
15 "Create an empty network and add nodes to it."
16 net = Mininet()
17 info( '*** Adding controller\n' )
18 net.addController( 'c0' )
19
20 info( '*** Adding hosts\n' )
21 h1 = net.addHost( 'h1', ip='10.10.14.1/24' )
22 h2 = net.addHost( 'h2', ip='10.10.24.2/24' )
23 h3 = net.addHost( 'h3', ip='10.10.34.3/24' )
24 h4 = net.addHost( 'h4', ip='10.10.14.4/24' )
25
26 info( '*** Adding switch\n' )
27 s14 = net.addSwitch( 's14' )
28 s24 = net.addSwitch( 's24' )
29 s34 = net.addSwitch( 's34' )
30
31 info( '*** Creating links\n' )
32 net.addLink( h1, s14 )
33 net.addLink( h4, s14 )
34
35 net.addLink( h2, s24 )
36 net.addLink( h4, s24 )
37
38 net.addLink( h3, s34 )
39 net.addLink( h4, s34 )
40
41 h4.cmd('ip addr add 10.10.24.4/24 dev h4-eth1')
42 h4.cmd('ip addr add 10.10.34.4/24 dev h4-eth2')
43 h4.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')
44 h3.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')
45
46 info( '*** Starting network\n' )
47 net.start()
48 h1.cmd('ip route add default via 10.10.14.4')
49 h2.cmd('ip route add default via 10.10.24.4')
50 h3.cmd('ip route add default via 10.10.34.4')
51

```

attacker



```

51
52 "This is used to run commands on the hosts"
53
54 info( '*** Starting terminals on hosts\n' )
55 h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
56 h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')
57 h3.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h3 &')
58 h4.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h4 &')
59
60 info( '*** Running the command line interface\n' )
61 CLI( net )
62
63 info( '*** Closing the terminals on the hosts\n' )
64 h1.cmd("killall xterm")
65 h2.cmd("killall xterm")
66 h3.cmd("killall xterm")
67 h4.cmd("killall xterm")
68
69 info( '*** Stopping network' )
70 net.stop()
71
72 "main Function: This is called when the Python file is run"
73 if __name__ == '__main__':
74     setLogLevel( 'info' )
75     firstNetwork()
76

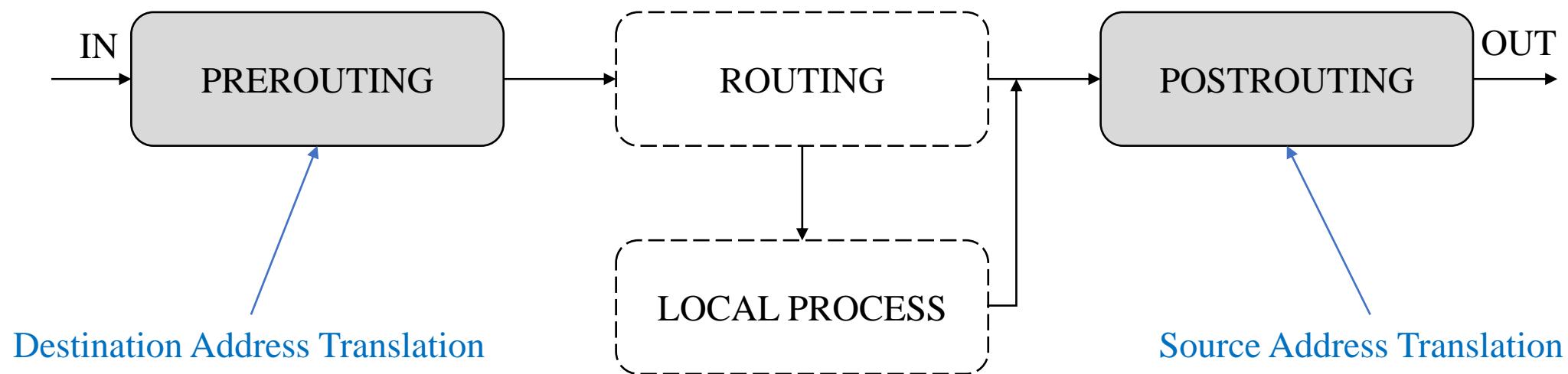
```

# iptables

- The Linux kernel contains a packet filter framework called **netfilter** which enables a Linux machine to use rule chains and configure the IP packets. When a connection tries to establish itself on your system, iptables looks for a rule in its list to match it to. If it doesn't find one, it resorts to the default action.
- The three types of iptables:
  1. Mangle: to manage class-based queuing, modify QoS, TTL, ...
  2. NAT: to change the IP addresses of the packets
  3. Filter: to accept or drop packets
- iptables command:
  - `$ iptables -t [table] [...]`

# NAT table

- This table has two types of rule chains:
  1. PREROUTING: to modify packets as soon as they arrive at the computer
  2. POSTROUTING: to modify packets that are ready to leave the computer



# Destination NAT

- PREROUTING rules are used for Destination NAT
- # iptables -t nat -A PREROUTING [match pattern] -j [action]
  - -A: Append a rule at the end of the PREROUTING chain
  - [match pattern]:
    - -p [protocol]: -p icmp, -p tcp, -p udp, ...
    - -s [source\_ip]: -s 192.168.1.1
    - -d [destination\_ip]: -d 192.168.2.2
    - -i [incoming\_interface\_name]: -i h1-eth0, -i h4-eth2
    - (Only for tcp & udp:) --dport [destination\_port\_number]: --dport 80
  - [action]:
    - DNAT --to [desired\_destination\_ip]

# DNAT examples

- *Change destination of TCP packets from 1.1.1.1 into 3.3.3.3:*

```
# iptables -t nat -A PREROUTING -p tcp -s 1.1.1.1 -j DNAT --to 3.3.3.3
```

- *Change destination of TCP packets to 2.2.2.2 into 3.3.3.3:*

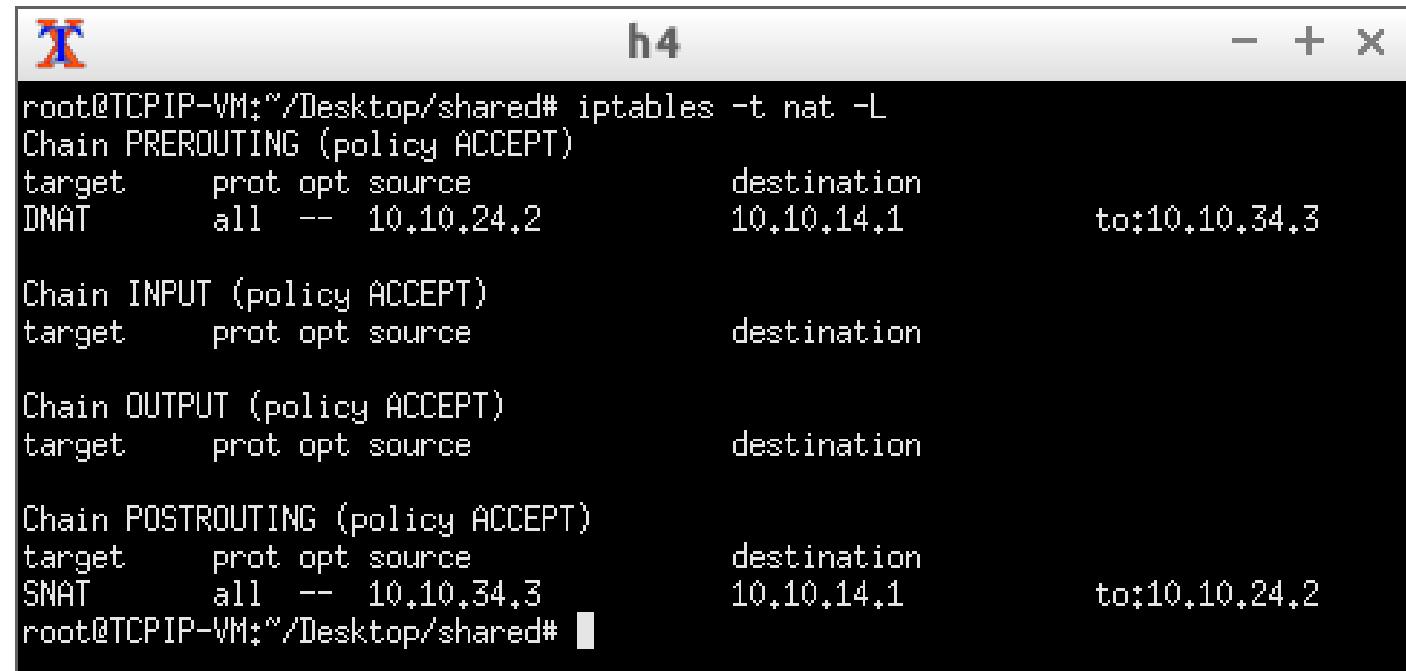
```
# iptables -t nat -A PREROUTING -p tcp -d 2.2.2.2 -j DNAT --to 3.3.3.3
```

- *Change destination of packets from 1.1.1.1 to 2.2.2.2 into 3.3.3.3:*

```
# iptables -t nat -A PREROUTING -s 1.1.1.1 -d 2.2.2.2 -j DNAT --to 3.3.3.3
```

# Rule chains

- Flush (remove) all rules in a chain:
  - # iptables -t nat -F PREROUTING (POSTROUTING)
- Flush (remove) all rules:
  - # iptables -t nat -F
- List rules:
  - # iptables -t nat -L



```
h4
root@TCPiP-VM:~/Desktop/shared# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target    prot opt source          destination
DNAT     all  --  10.10.24.2    10.10.14.1      to:10.10.34.3

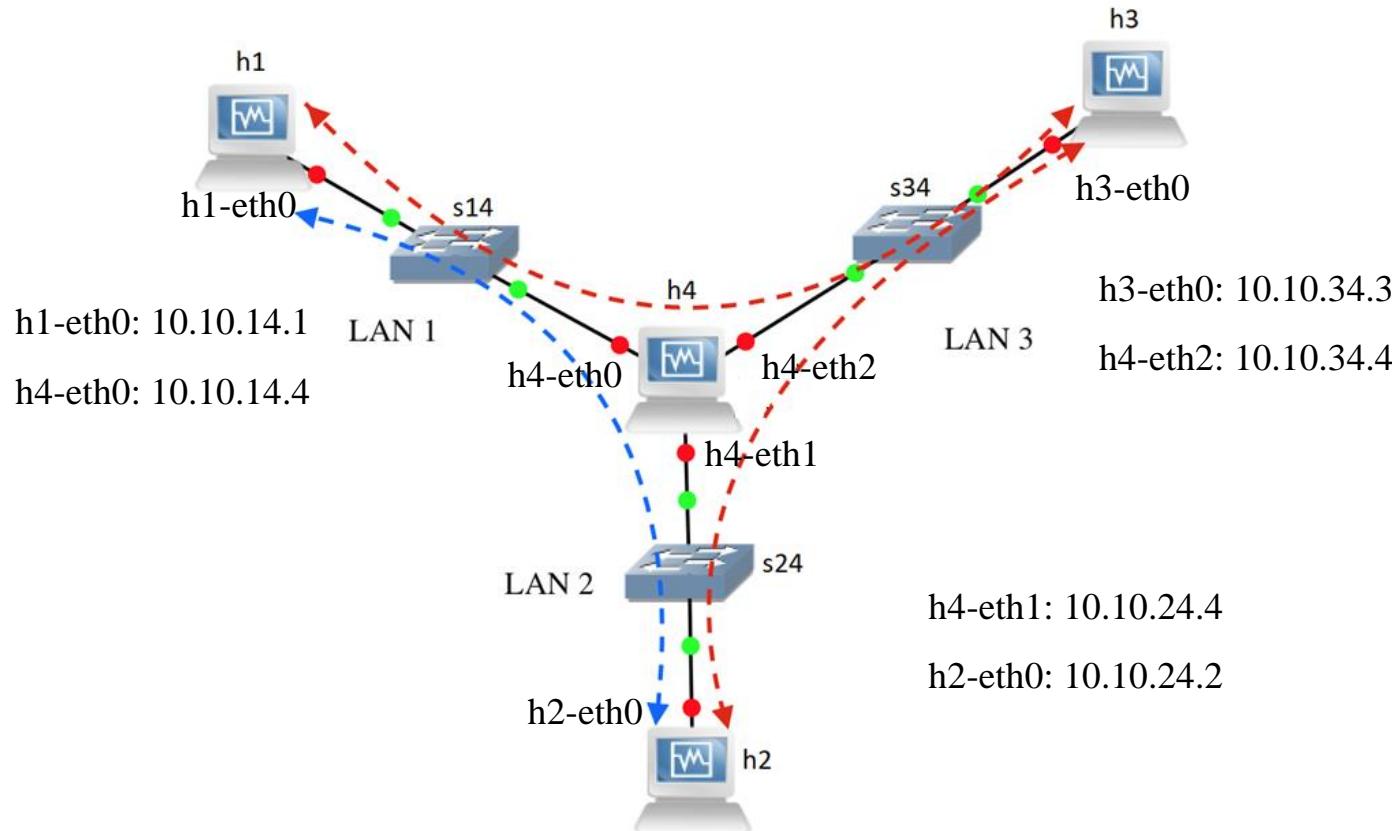
Chain INPUT (policy ACCEPT)
target    prot opt source          destination

Chain OUTPUT (policy ACCEPT)
target    prot opt source          destination

Chain POSTROUTING (policy ACCEPT)
target    prot opt source          destination
SNAT     all  --  10.10.34.3    10.10.14.1      to:10.10.24.2
root@TCPiP-VM:~/Desktop/shared#
```

# Man-in-the-middle Attack

- `# iptables -t nat -A PREROUTING -s [source_ip] -d [destination_ip] -j DNAT --to [desired_destination_ip]`



# Reverse Path Filtering (RPF)

- Reverse path filtering is a mechanism adopted by the Linux kernel, as well as most of the networking devices out there to check whether a receiving packet source address is routable.
- So in other words, when a machine with reverse path filtering enabled receives a packet, the machine will first check whether the source of the received packet is reachable through the interface it came in.
  - If it is routable through the interface which it came, then the machine will accept the packet.
  - If it is not routable through the interface which it came, then the machine will drop that packet.
- Basically, if the reply to this packet wouldn't go out the interface this packet came in, then this is a bogus packet and should be ignored.

# Source NAT

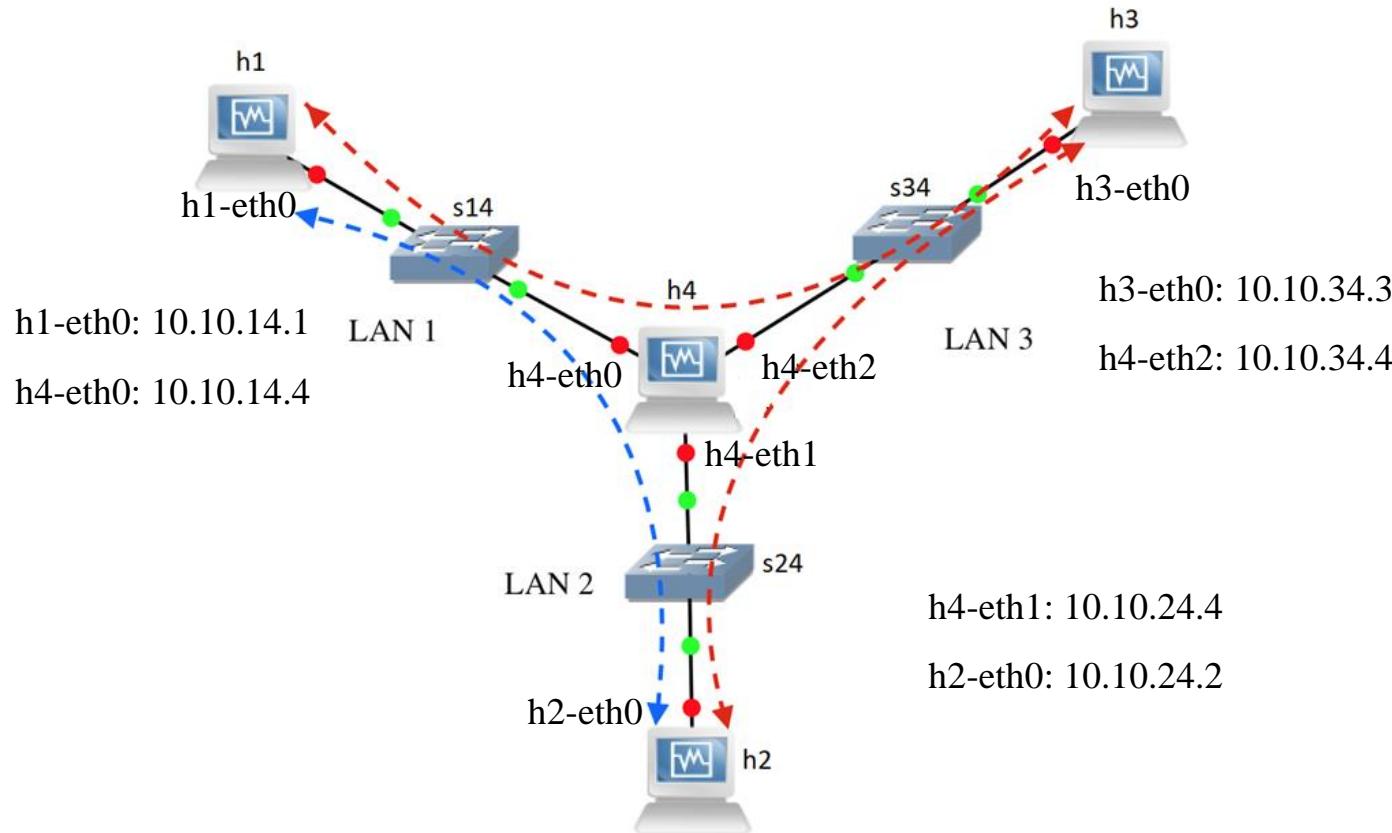
- POSTROUTING rules are used for Source NAT
- # iptables -t nat -A POSTROUTING [match pattern] -j [action]
  - [match pattern]:
    - -p [protocol]: -p icmp, -p tcp, -p udp, ...
    - -s [source\_ip]: -s 192.168.1.1
    - -d [destination\_ip]: -d 192.168.2.2
    - -o [outgoing\_interface\_name]: -o h1-eth0, -o h4-eth2
    - (Only for tcp & udp:) --sport [source\_port\_number]: --sport 80
  - [action]:
    - SNAT --to [desired\_source\_ip]
    - MASQUERADE
      - Source IP is replaced with the ip of the host outgoing interface
      - MASQUERADE  $\equiv$  SNAT --to [outgoing\_interface\_ip]

# SNAT examples

- *Change source of packets from 1.1.1.1 leaving at h4-eth0 into 3.3.3.3:*  
# iptables -t nat -A POSTROUTING -o h4-eth0 -s 1.1.1.1 -j SNAT --to 3.3.3.3
- *Change source of packets to 2.2.2.2 leaving at h4-eth0 into 3.3.3.3:*  
# iptables -t nat -A POSTROUTING -o h4-eth0 -d 2.2.2.2 -j SNAT --to 3.3.3.3
- *Change source of packets from 1.1.1.1 to 2.2.2.2 into 3.3.3.3:*  
# iptables -t nat -A POSTROUTING -s 1.1.1.1 -d 2.2.2.2 -j SNAT --to 3.3.3.3

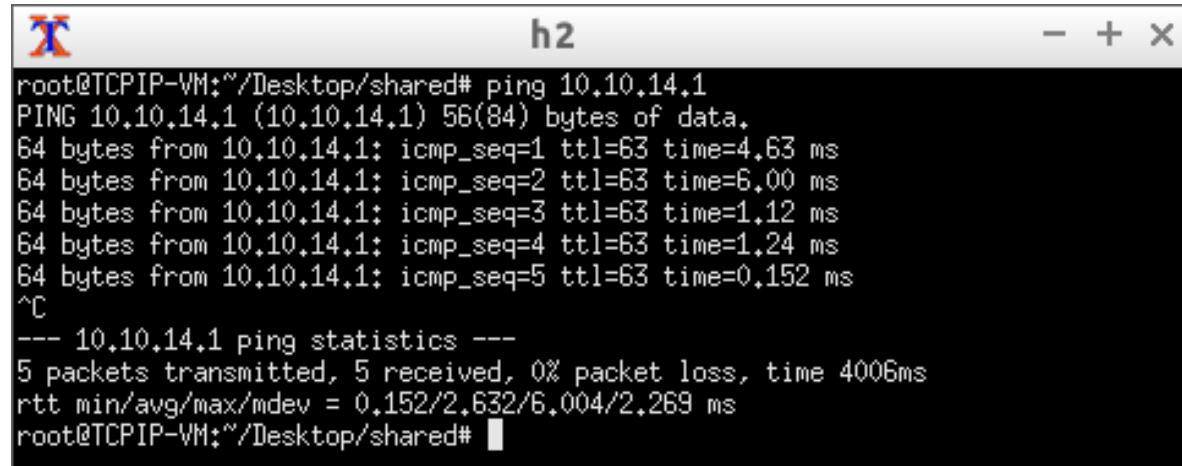
# Man-in-the-middle Attack

- `# iptables -t nat -A POSTROUTING -s [source_ip] -d [destination_ip] -j SNAT --to [desired_source_ip]`



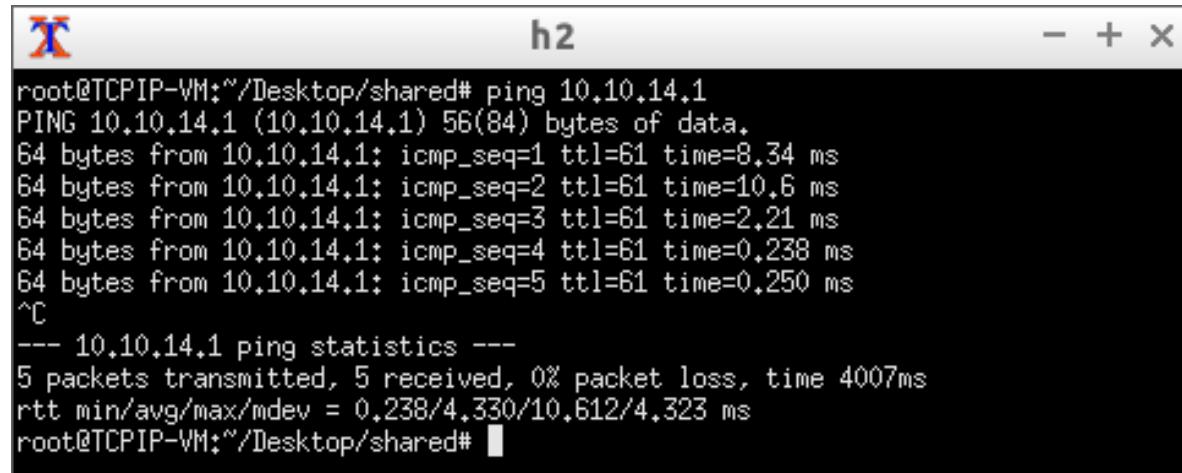
# How can Alice notice the attack?

- Before attack:



```
root@TCPiP-VM:~/Desktop/shared# ping 10.10.14.1
PING 10.10.14.1 (10.10.14.1) 56(84) bytes of data.
64 bytes from 10.10.14.1: icmp_seq=1 ttl=63 time=4.63 ms
64 bytes from 10.10.14.1: icmp_seq=2 ttl=63 time=6.00 ms
64 bytes from 10.10.14.1: icmp_seq=3 ttl=63 time=1.12 ms
64 bytes from 10.10.14.1: icmp_seq=4 ttl=63 time=1.24 ms
64 bytes from 10.10.14.1: icmp_seq=5 ttl=63 time=0.152 ms
^C
--- 10.10.14.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.152/2.632/6.004/2.269 ms
root@TCPiP-VM:~/Desktop/shared#
```

- After attack:



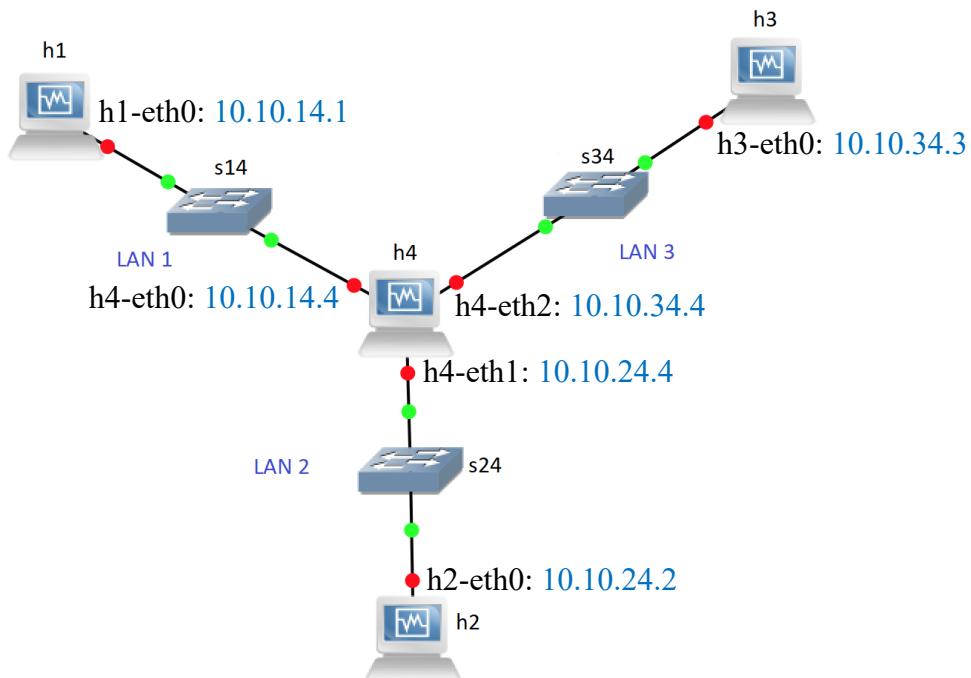
```
root@TCPiP-VM:~/Desktop/shared# ping 10.10.14.1
PING 10.10.14.1 (10.10.14.1) 56(84) bytes of data.
64 bytes from 10.10.14.1: icmp_seq=1 ttl=61 time=8.34 ms
64 bytes from 10.10.14.1: icmp_seq=2 ttl=61 time=10.6 ms
64 bytes from 10.10.14.1: icmp_seq=3 ttl=61 time=2.21 ms
64 bytes from 10.10.14.1: icmp_seq=4 ttl=61 time=0.238 ms
64 bytes from 10.10.14.1: icmp_seq=5 ttl=61 time=0.250 ms
^C
--- 10.10.14.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 0.238/4.330/10.612/4.323 ms
root@TCPiP-VM:~/Desktop/shared#
```

## آزمایشگاه شبکه

### آزمایش ۳: شبیه‌سازی حمله فرد میانه (Man-in-the-Middle)

#### الف) پیکربندی توپولوژی شبکه محلی

پیش‌شرط انجام آزمایش جاری، پیکربندی توپولوژی شکل ۱ است که قبلاً در قالب آزمایش ۱ انجام داده‌اید. اسکریپت پایتون lab3.py را ملاحظه کرده و در صورت نیاز، آن را اصلاح نمایید تا کلیه پیکربندی‌های مورد نظر روی کلیه ماشین‌ها و لینک‌ها به درستی انجام شده باشد.



شکل ۱- توپولوژی شبکه محلی پیکربندی شده در آزمایش ۱

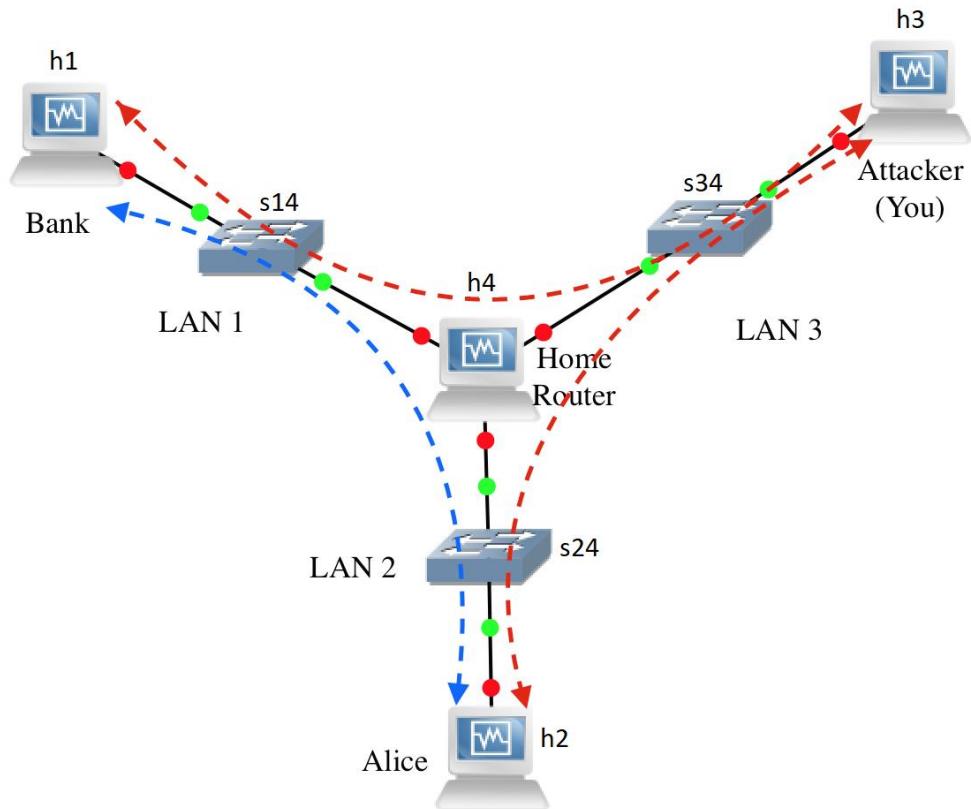
#### ب) تعریف نقش گره‌های شبکه برای پیاده‌سازی سناریوی حمله

به منظور پیاده‌سازی حمله «فرد میانه»، گره‌های توپولوژی شکل ۱ را بر اساس نقش جدیدشان برچسب می‌زنیم  
• شکل ۲ را ملاحظه نمایید.

- نقش سرور بانک را بازی می‌کند.
- h2 را کاربر Alice در نظر می‌گیریم که در پی دسترسی به حساب بانکی اش است.
- h3 سرور مورد استفاده مهاجم است و هدف، تغییر مسیر تعاملات بانکی Alice به مقصد این سرور و سرقت پول وی می‌باشد.
- h4 نیز روتر مورد استفاده Alice برای ارتباط گرفتن با بانک است. جلوتر، فرض خواهیم کرد که این روتر برای پیاده‌سازی حمله، دستخوش دستکاری‌هایی خواهد شد.

- فرض می‌کنیم که یک ping موفق بین Alice و بانک معادل با یک عملیات انتقال پول موفق بین آنهاست. هدف مهاجم این است که به Alice و بانک القاء کند که دارای یک ارتباط موفق هستند (که در شکل ۲ با پیکان آبی نشان داده شده) در حالی که در واقعیت، هر محاوره‌ای میان این دو از سرور مهاجم (h3) گذر داده می‌شود (پیکان‌های قرمز در شکل ۲).

- فرض براین است که برای پیاده‌سازی حمله تنها می‌توان روی روتر h4 و نیز سرور مهاجم (h3) تغییر ایجاد کرد.



شکل ۲- توبولوژی شبکه محلی تحت حمله «فرد میانه»

#### ج) مکانیزم «فیلترسازی بر مبنای مسیر معکوس» در روتر h4

- جلوتر خواهیم دید که به منظور پیاده‌سازی حمله «فرد میانه»، در مرحله‌ای نیازمند جعل آدرس IP کاربر Alice خواهیم بود (IP spoofing). یکی از راهکارهای پیشگیری از چنین حملاتی در روترها، مکانیزمی است تحت عنوان «فیلترسازی بر مبنای مسیر معکوس» (RPF) (Reverse Path Filtering).

- مکانیزم RPF ابتدا آدرس فرستنده بسته را در نظر گرفته و بررسی می‌کند که بسته‌های ارسالی از آن آدرس، به طور قانونی باید از کدام اینترفیس دریافت شوند. سپس اینترفیس بدست آمده را با اینترفیسی که بسته از آن وارد شده، مقایسه می‌نماید و در صورت مغایرت، بسته را دور می‌ریزد. جهت آشنایی بیشتر با RPF، به اسلایدهای ضمیمه این آزمایش با نام Man-in-the-Middle.pdf مراجعه کرده و آنها را مطالعه نمایید.

- با توجه به اینکه برای پیاده‌سازی حمله، سرور مهاجم باید قادر به فوروارد کردن بسته‌ها نیز باشد، این قابلیت را برای h3 فعال می‌نماییم (با اجرای دستوری نظیر آنچه در آزمایش ۱ فرا گرفتید).

## د) پیکربندی حمله «فرد میانه»

- به منظور پیکربندی حمله، از قابلیت ویژه‌ای بهره می‌گیریم که کرنل لینوکس برای پردازش بسته‌ها در اختیار admin‌های شبکه می‌گذارد. در واقع، یک برنامه سمت کاربر به نام iptables از سوی کرنل لینوکس پشتیبانی می‌شود که از طریق آن، ما می‌توانیم با تعریف زنجیره‌ای از قوانین، پردازش‌های مورد نظر خود را روی بسته‌های IPv4 انجام دهیم.

- با استفاده از برنامه iptables، ابتدا روتر h4 را طوری دستکاری می‌کنیم تا ترافیک به مقصد بانک را برای سرور مهاجم بفرستد. البته، از آنجایی که Alice ممکن است دارای حجم زیادی فعالیت‌های شبکه‌ای غیرمرتبط با مصرف پهنه‌ای باند بالا داشته باشد (مثلًا: بازی آنلاین و غیره)، باید به طریقی بتوانیم فقط نوع ترافیکی را که به آن علاوه‌مندیم (بسته‌های ICMP) و از سوی آدرس IP کاربر Alice هم منشأ می‌گیرد (10.10.24.2) برگزینیم.

### سوال ۱- دستورات لازم برای تحقیق هدف فوق را بنویسید.

- حال، با راهاندازی برنامه WireShark روی h3 بررسی کنید که آیا هدایت ترافیک بانکی از روتر h4 به سوی سرور مهاجم موفق بوده است یا خیر.
- در گام بعدی، باید سرور مهاجم (h3) را طوری پیکربندی نماییم که ترافیک واردہ از سوی سیستم Alice را دریافت کرده، آن را دستکاری نموده و در نهایت برای بانک بفرستد. برای این منظور، باید آدرس IP مقصد بسته را با آدرس IP سرور بانک (10.10.14.1) جایگزین نماییم. اما، هنگام خروج از h3، آدرس IP مبدأ بسته، باید برابر با آدرس سیستم Alice قرار داده شود (به دلیل همان مکانیزم فیلترسازی بر مبنای مسیر معکوس).
- تا اینجا، روی سرور مهاجم (h3)، باید بسته‌هایی از مبدأ 10.10.24.2 به مقصد 10.10.34.3 (و برعکس) را ملاحظه نمایید و همینطور از مبدأ 10.10.34.3 به مقصد 10.10.14.1 (و برعکس).
- با راهاندازی WireShark روی سرور بانک (h1)، وضعیت بسته‌های دریافتی توسط بانک را بررسی کنید.
- روی سرور بانک (h1) هم باید بسته‌هایی از مبدأ 10.10.34.3 ملاحظه کنید. اما به این ترتیب، بانک به سادگی متوجه غیرخودی بودن این بسته‌ها خواهد شد (مثلًا: سرور بانک ممکن است دارای یک لیست «کنترل دسترسی» باشد که صرفاً به ارتباطات واردہ از سوی آدرس‌های IP مشتریانش اجازه دسترسی می‌دهد). در این گام، دستورات iptables‌ای پیشنهاد دهید که با استفاده از آنها بتوانید روتر h4 را طوری پیکربندی کنید که کاربر Alice را به جای سرور مهاجم جا بزند (تعویض آدرس مبدأ بسته‌های h3 به h1).

### سوال ۲- دستورات لازم برای تحقیق هدف فوق را بنویسید.

- به این ترتیب، کار پیاده‌سازی حمله «فرد میانه» تمام می‌شود. فقط به دو سوال دیگر پاسخ دهید:
- سوال ۳- آیا این حمله را می‌توانستیم صرفاً با دستکاری جداول مسیریابی روتر h4 محقق کنیم؟
- سوال ۴- آیا در محیط LAN مورد مثال ما، کاربر Alice راهکاری برای تشخیص اینکه تحت حمله قرار گرفته دارد (البته به جز اینکه متوجه خالی شدن حساب بانکی اش بشود)؟!

به نام خدا

## گزارشکار آزمایش سوم آزمایشگاه شبکه های کامپیوتری: شبیه سازی حمله فرد میانه (Man-in-the-Middle)

سیده شکیبا انارکی فیروز - 99442047

بهاره کاووسی نژاد - 99431217

1 و 2 - کد IanConfig.py پس از تکمیل:



```
#!/usr/bin/python

"""
This example shows how to create a Mininet object and add nodes to it manually.
"""

"Importing Libraries"
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

"Function definition: This is called from the main function"
def firstNetwork():
    "Create an empty network and add nodes to it."
    net = Mininet()

    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.10.14.1/24' )
    h2 = net.addHost( 'h2', ip='10.10.24.2/24' )
    h3 = net.addHost( 'h3', ip='10.10.34.2/24' )
    h4 = net.addHost( 'h4', ip='10.10.14.4/24' )

    info( '*** Adding switch\n' )
    s14 = net.addSwitch( 's14' )
    s24 = net.addSwitch( 's24' )
    s34 = net.addSwitch( 's34' )

    info( '*** Creating links\n' )
    net.addLink( h1, s14 )
    net.addLink( h4, s14 )
    net.addLink( h2, s24 )
    net.addLink( h4, s24 )
    net.addLink( h3, s34 )
    net.addLink( h4, s34 )

    h4.cmd('ip addr add 10.10.24.4/24 dev h4-eth1')
    h4.cmd('ip addr add 10.10.34.4/24 dev h4-eth2')
    h4.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')
    h4.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')

    info( '*** Starting network\n' )
    net.start()

    # info( '*** Adding Gateways\n' )
    h1.cmd('ip route add default via 10.10.14.4')
    h2.cmd('ip route add default via 10.10.24.4')
    h3.cmd('ip route add default via 10.10.34.4')

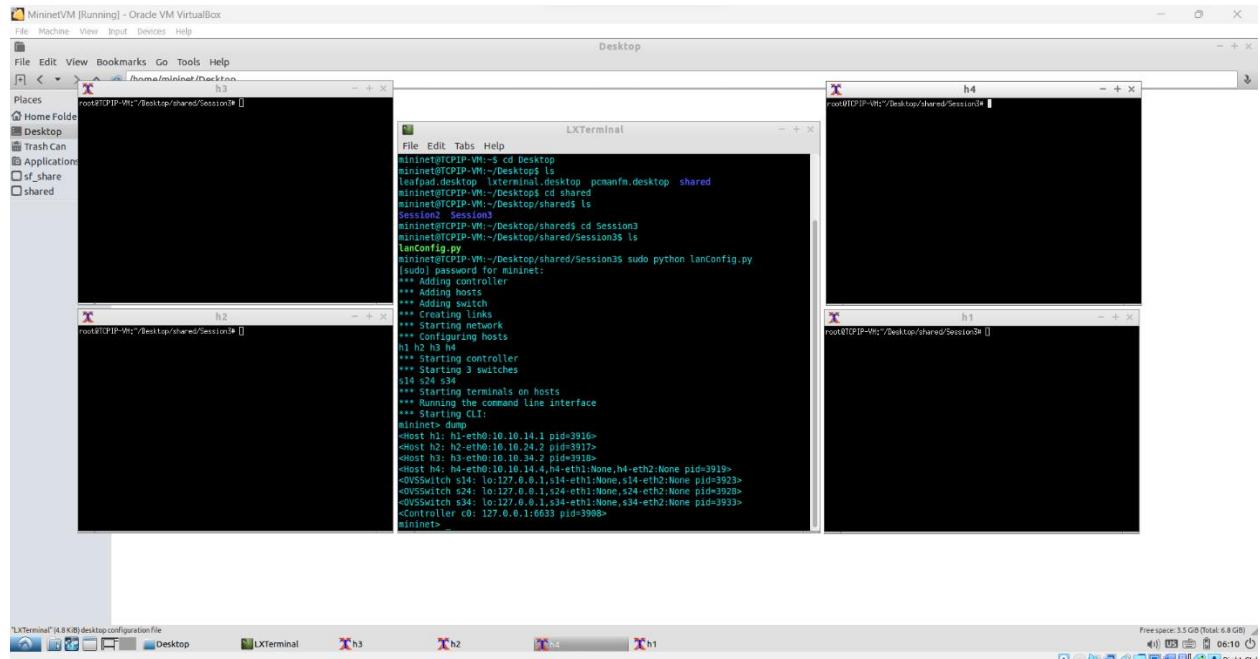
    info( '*** Starting terminals on hosts\n' )
    h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
    h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')
    h3.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h3 &')
    h4.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h4 &')

    info( '*** Running the command line interface\n' )
    CLI( net )

    info( '*** Closing the terminals on the hosts\n' )
    h1.cmd("killall xterm")
    h2.cmd("killall xterm")
    h3.cmd("killall xterm")
    h4.cmd("killall xterm")

    info( '*** Stopping network' )
    net.stop()

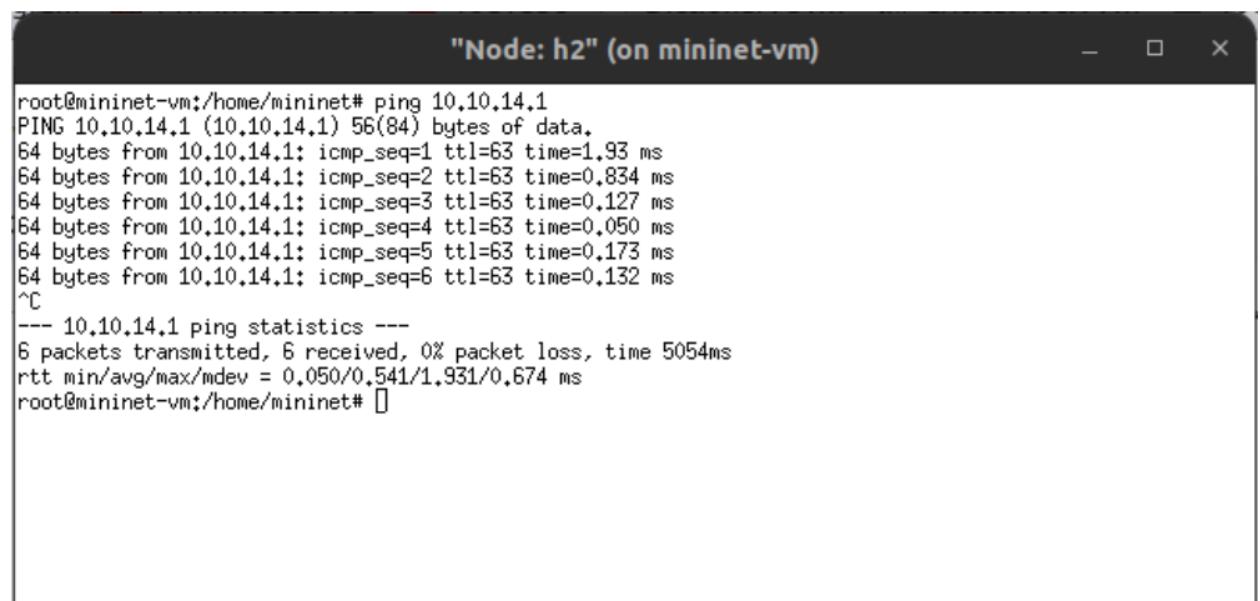
"main Function: This is called when the Python file is run"
if __name__ == '__main__':
    setLogLevel('info')
    firstNetwork()
```

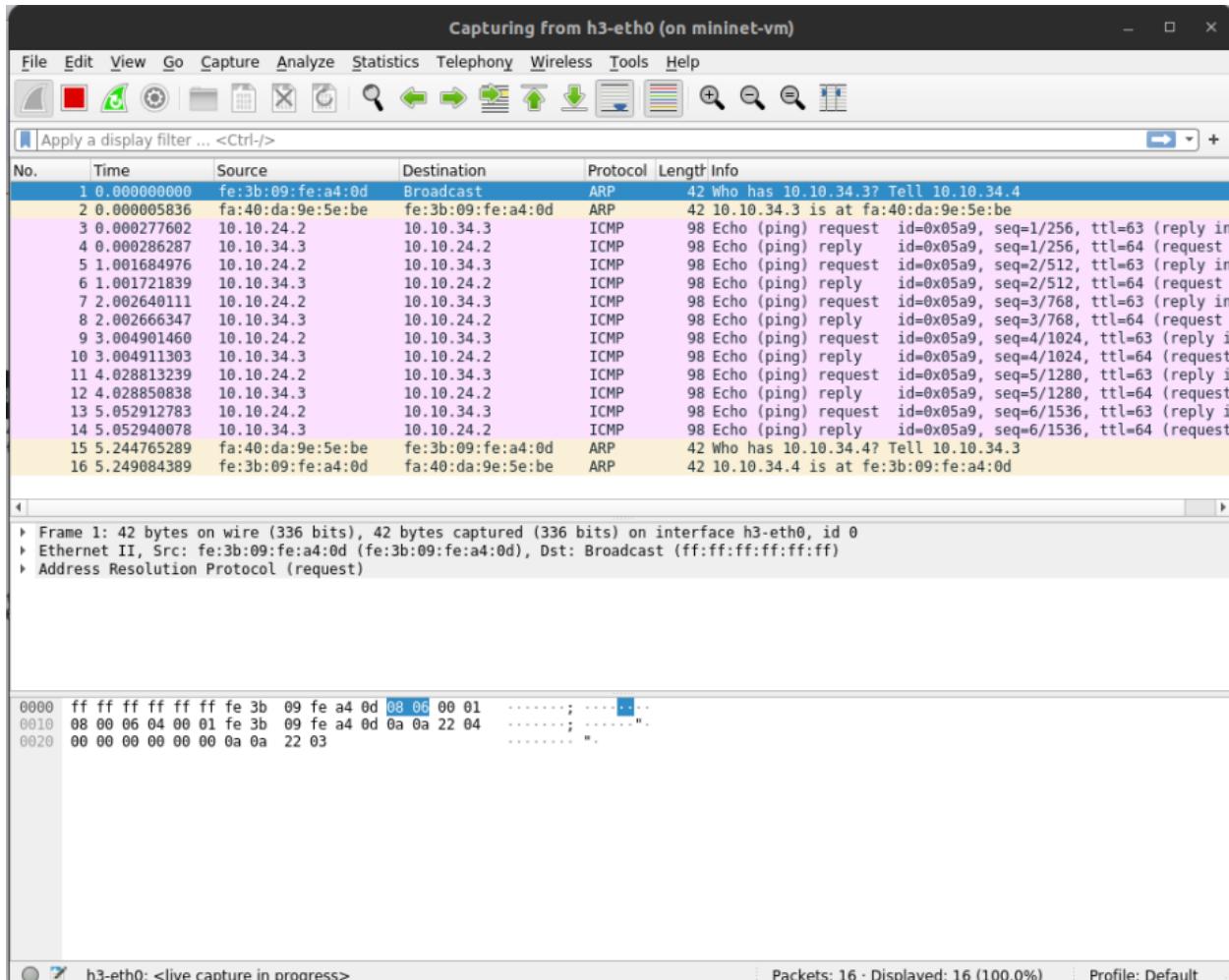


ابتدا home router h4 را تغییر می دهیم تا بسته های h2 را به h1 تغییر داده و مقصد آنها را به h3 تغییر دهیم. فقط باید بسته هایی که پروتکل icmp دارند را تغییر دهیم.



نتیجه در wireshark





همانطور که مشخص است h2 در حال پینگ کردن h1 می باشد اما h3 بسته هایی با مبدأ h2 و مقصد خودش دریافت کرده است. حال هاست h3 باید بسته ها را تغییر دهد و مقصد آنها را به h1 دوباره تغییر دهد که اینجا مشکلی بوجود خواهد آمد که بسته های جدید مبدأ h2 و مقصد خواهند داشت و توسط روتر h4 دوباره به دست h3 خواهند رسید و درون یک حلقه میافتیم. برای جلوگیری از این کار در h3 مبدأ بسته را نیز از h2 به h3 تغییر میدهیم. چون از جنس SNAT است باید از POSTROUTING استفاده کنیم.

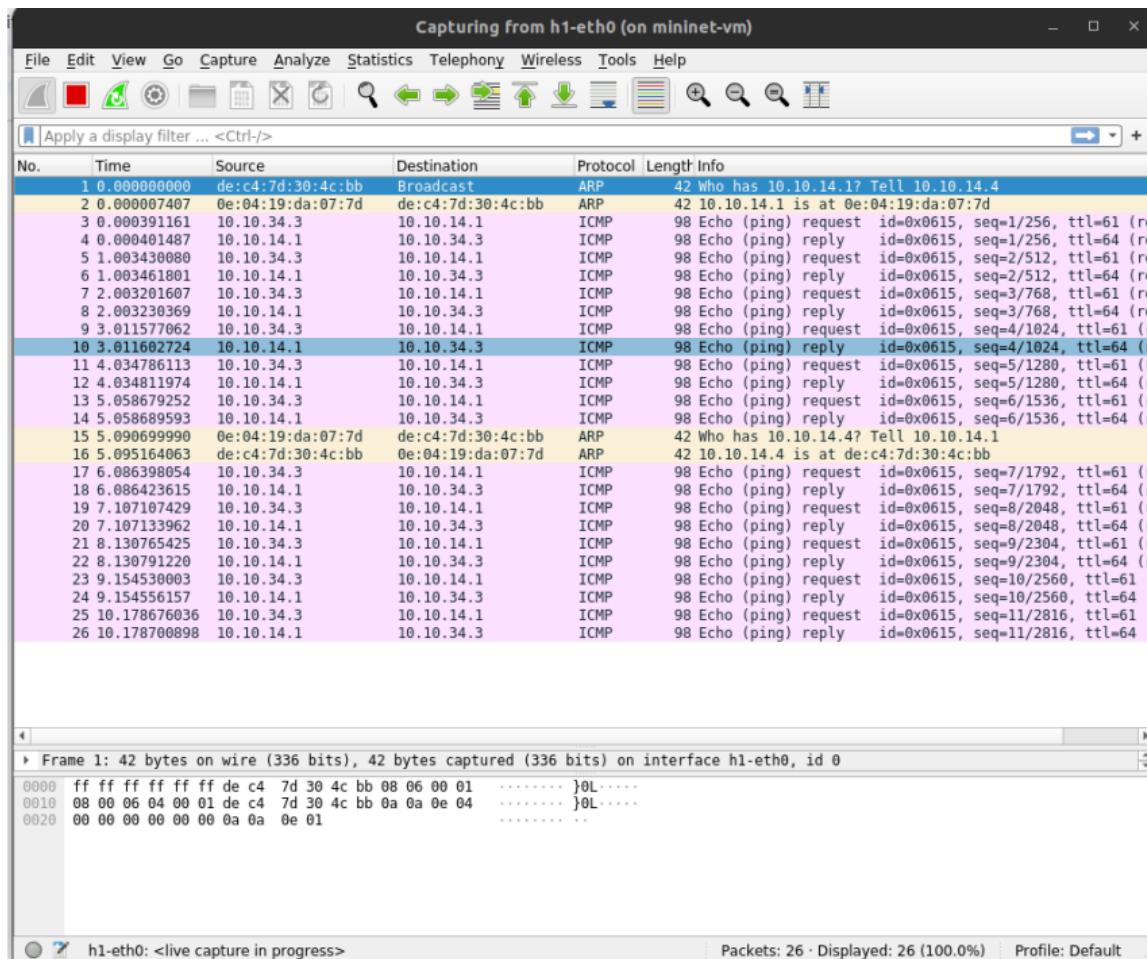
```
root@mininet-vm:/home/mininet# iptables -t nat -A POSTROUTING -p icmp -s 10.10.24.2 -d 10.10.14.1 -j SNAT --to 10.10.34.3
root@mininet-vm:/home/mininet# iptables -t nat -A PREROUTING -p icmp -s 10.10.24.2 -d 10.10.34.3 -j DNAT --to 10.10.14.1
root@mininet-vm:/home/mininet#
```

"Node: h2" (on mininet-vm)

```

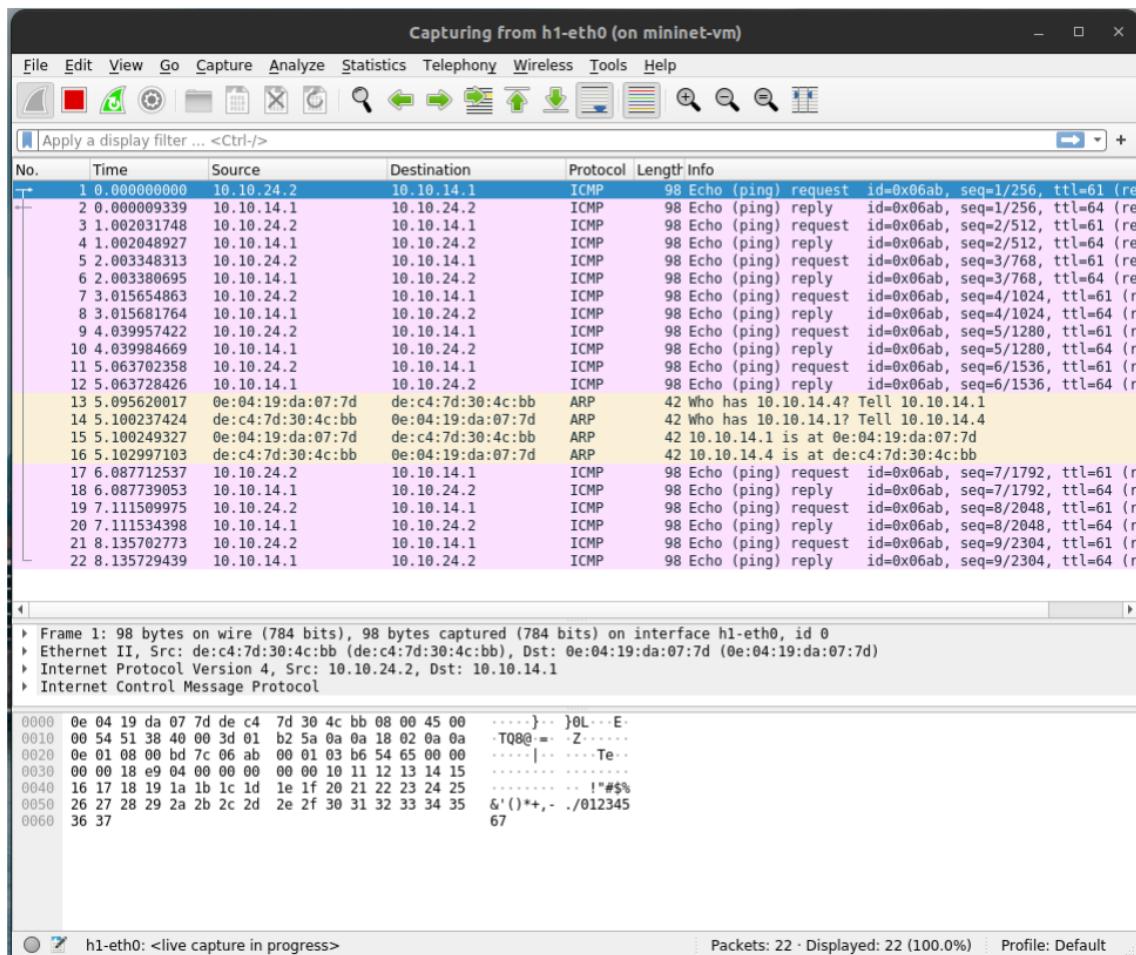
root@mininet-vm:/home/mininet# ping 10.10.14.1
PING 10.10.14.1 (10.10.14.1) 56(84) bytes of data.
64 bytes from 10.10.14.1: icmp_seq=1 ttl=61 time=1.94 ms
64 bytes from 10.10.14.1: icmp_seq=2 ttl=61 time=3.27 ms
64 bytes from 10.10.14.1: icmp_seq=3 ttl=61 time=0.588 ms
64 bytes from 10.10.14.1: icmp_seq=4 ttl=61 time=0.151 ms
64 bytes from 10.10.14.1: icmp_seq=5 ttl=61 time=0.163 ms
64 bytes from 10.10.14.1: icmp_seq=6 ttl=61 time=0.063 ms
64 bytes from 10.10.14.1: icmp_seq=7 ttl=61 time=0.166 ms
64 bytes from 10.10.14.1: icmp_seq=8 ttl=61 time=0.154 ms
64 bytes from 10.10.14.1: icmp_seq=9 ttl=61 time=0.163 ms
64 bytes from 10.10.14.1: icmp_seq=10 ttl=61 time=0.164 ms
64 bytes from 10.10.14.1: icmp_seq=11 ttl=61 time=0.152 ms
^C
--- 10.10.14.1 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10180ms
rtt min/avg/max/mdev = 0.063/0.633/3.273/0.980 ms
root@mininet-vm:/home/mininet#

```



در اینجا h1 به خاطر RPF میتواند بسته هایی که مبدأ آن drop کند پس باید مبدأ بسته ها را نیز به h2 تغییر داد. برای این کار یک rule در h4 اضافه میکنیم.

```
"Node: h4" (on mininet-vm)
root@mininet-vm:/home/mininet# iptables -t nat -A PREROUTING -p icmp -s 10.10.24.2 -d 10.10.14.1 -j DNAT --to 10.10.34.3
root@mininet-vm:/home/mininet# iptables -t nat -A POSTROUTING -p icmp -s 10.10.34.3 -d 10.10.14.1 -j SNAT --to 10.10.24.2
root@mininet-vm:/home/mininet#
```



مشاهده میکنیم که مبدأ بسته ها را از  $h3$  به  $h2$  تغییر داده ایم و سپس  $h2$  پینگ میکند  $h1$  و مشاهده میکنیم که در wireshark مربوط به  $h1$  بسته هایی با مبدأ  $h2$  و مقصد  $h1$  دیده میشود.

-3

خیر. چون پیام باید ابتدا به  $h3$  برسد و اگر table ip هاست  $h3$  را تغییر ندهیم نمی توانیم پیام را به  $h1$  فوروارد کنیم.

-4

بله؛ می تواند متوجه شود. با استفاده از ttl و rtt میتوانیم بفهمیم بسته در مسیر دیگری قرار گرفته است یا خیر. به طور معمول ttl باید 63 باشد ولی بعد از گذشتن از  $h3$  می توان مشاهده کرد که ttl آن به 61 کاهش پیدا کرده است.

## گزارش ۳

بکتاش انصاری

پوریا رحیمی

سوال ۱:

کد تغییر داده شده به شکل زیر است:

```
1 #!/usr/bin/python
2
3 """
4 This example shows how to create a Mininet object and add nodes to it manually.
5 """
6 "Importing Libraries"
7 from mininet.net import Mininet
8 from mininet.node import Controller
9 from mininet.cli import CLI
10 from mininet.log import setLogLevel, info
11
12 #Function definition: This is called from the main function
13 def firstNetwork():
14
15     "Create an empty network and add nodes to it."
16     net = Mininet()
17     info( '*** Adding controller\n' )
18     net.addController( 'c0' )
19
20     info( '*** Adding hosts\n' )
21     h1 = net.addHost( 'h1', ip = '10.10.14.1/24' )
22     h2 = net.addHost( 'h2', ip = '10.10.24.2/24' )
23     h3 = net.addHost( 'h3', ip = '10.10.34.3/24' )
24     h4 = net.addHost( 'h4', ip = '10.10.14.4/24' )
25
```

```

25     info( '*** Adding switch\n' )
26     s14 = net.addSwitch( 's14' )
27     s24 = net.addSwitch( 's24' )
28     s34 = net.addSwitch( 's34' )
29
30
31
32     info( '*** Creating links\n' )
33     net.addLink( h1, s14 )
34     net.addLink( h4, s14 )
35     net.addLink( h2, s24 )
36     net.addLink( h4, s24 )
37     net.addLink( h3, s34 )
38     net.addLink( h4, s34 )
39
40     h4.cmd('ip addr add 10.10.24.4/24 dev h4-eth1')
41     h4.cmd('ip addr add 10.10.34.4/24 dev h4-eth2')
42     h4.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')
43     h3.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')
44
45     info( '*** Starting network\n' )
46     net.start()
47     h1.cmd('ip route add default via 10.10.14.4')
48     h2.cmd('ip route add default via 10.10.24.4')
49     h3.cmd('ip route add default via 10.10.34.4')

```

```

47     h1.cmd('ip route add default via 10.10.14.4')
48     h2.cmd('ip route add default via 10.10.24.4')
49     h3.cmd('ip route add default via 10.10.34.4')
50
51
52     #This is used to run commands on the hosts
53     info( '*** Starting terminals on hosts\n' )
54     h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
55     h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')
56     h3.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h3 &')
57     h4.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h4 &')
58
59     info( '*** Running the command line interface\n' )
60     CLI( net )
61
62     info( '*** Closing the terminals on the hosts\n' )
63     h1.cmd("killall xterm")
64     h2.cmd("killall xterm")
65     h3.cmd("killall xterm")
66     h4.cmd("killall xterm")
67
68     info( '*** Stopping network' )
69     net.stop()
70
71  #main Function: This is called when the Python file is run

```

```

51
52     #This is used to run commands on the hosts
53     info( '*** Starting terminals on hosts\n' )
54     h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
55     h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')
56     h3.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h3 &')
57     h4.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h4 &')
58
59     info( '*** Running the command line interface\n' )
60     CLI( net )
61
62     info( '*** Closing the terminals on the hosts\n' )
63     h1.cmd("killall xterm")
64     h2.cmd("killall xterm")
65     h3.cmd("killall xterm")
66     h4.cmd("killall xterm")
67
68     info( '*** Stopping network' )
69     net.stop()
70
71 #main Function: This is called when the Python file is run
72 if __name__ == '__main__':
73     setLogLevel( 'info' )
74     firstNetwork()

```

```

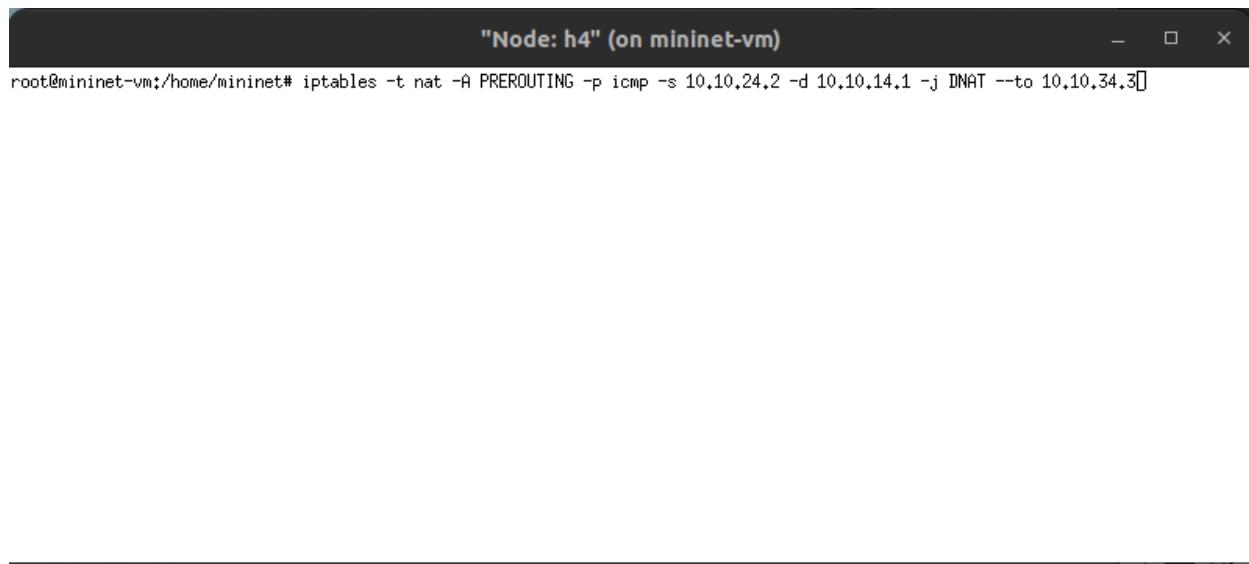
UNKNOWN COMMAND: <ctrl> dump
mininet> dump
<Host h1: h1-eth0:10.10.14.1 pid=921>
<Host h2: h2-eth0:10.10.24.2 pid=925>
<Host h3: h3-eth0:10.10.34.3 pid=927>
<Host h4: h4-eth0:10.10.14.4,h4-eth1:None,h4-eth2:None pid=929>
<OVSSwitch s14: lo:127.0.0.1,s14-eth1:None,s14-eth2:None pid=934>
<OVSSwitch s24: lo:127.0.0.1,s24-eth1:None,s24-eth2:None pid=937>
<OVSSwitch s34: lo:127.0.0.1,s34-eth1:None,s34-eth2:None pid=940>
<Controller c0: 127.0.0.1:6653 pid=914>
mininet> <ctrl>

```

## سوال ۲ :

برای این کار ابتدا باید h4 home router را تغییر دهیم به این شکل که پکت های h2 به h1 را تغییر داده و مقصد آنها را به h3 تغییر بدهیم. نکته قابل توجه این است که باید فقط پکت هایی که پروتکل icmp دارند را تغییر دهیم.

از دستور زیر استفاده میکنیم:

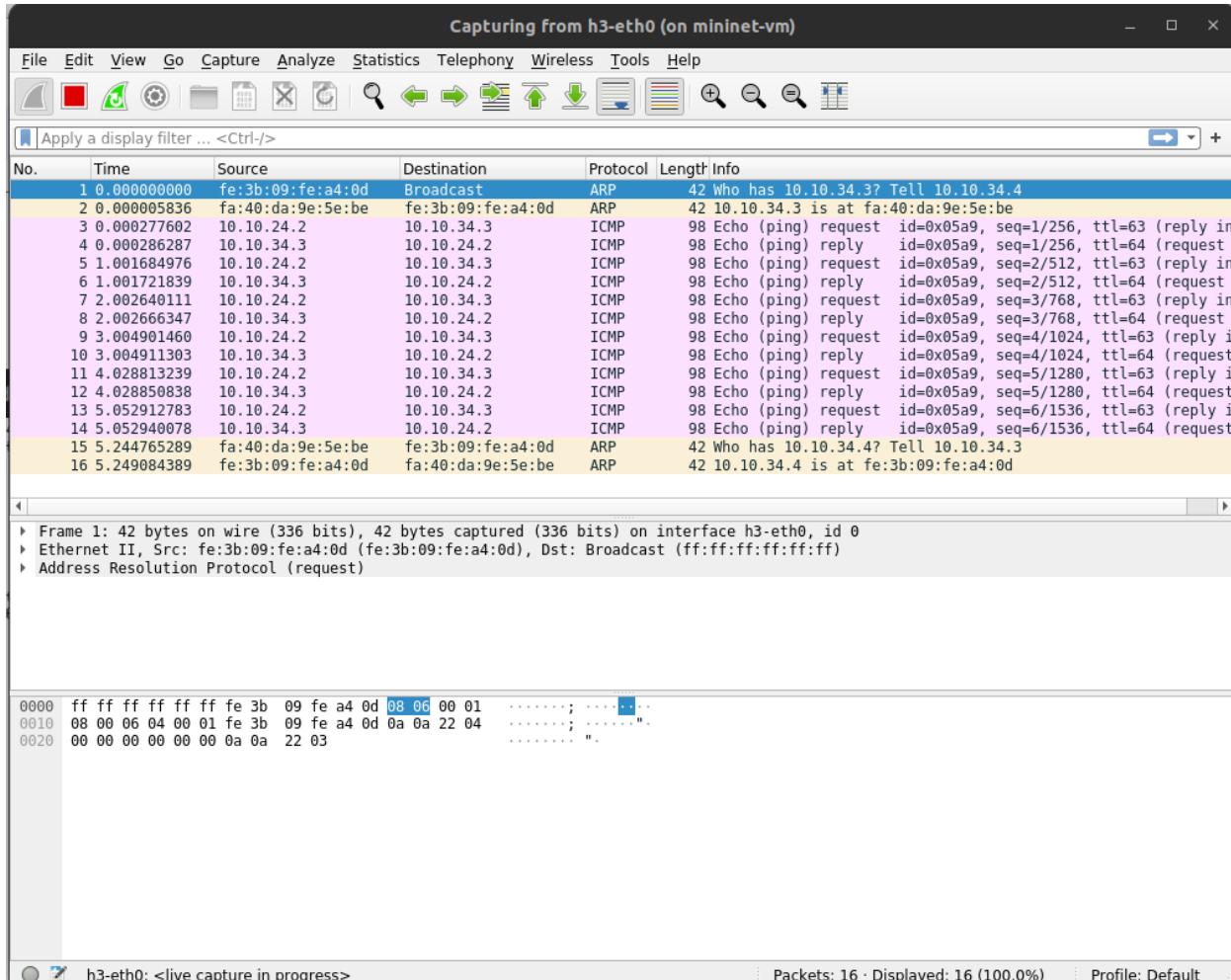


```
"Node: h4" (on mininet-vm)
root@mininet-vm:/home/mininet# iptables -t nat -A PREROUTING -p icmp -s 10.10.24.2 -d 10.10.14.1 -j DNAT --to 10.10.34.3
```

نتیجه را میتوانیم در Wireshark بینیم:

```
"Node: h2" (on mininet-vm) - X

root@mininet-vm:/home/mininet# ping 10.10.14.1
PING 10.10.14.1 (10.10.14.1) 56(84) bytes of data.
64 bytes from 10.10.14.1: icmp_seq=1 ttl=63 time=1.93 ms
64 bytes from 10.10.14.1: icmp_seq=2 ttl=63 time=0.834 ms
64 bytes from 10.10.14.1: icmp_seq=3 ttl=63 time=0.127 ms
64 bytes from 10.10.14.1: icmp_seq=4 ttl=63 time=0.050 ms
64 bytes from 10.10.14.1: icmp_seq=5 ttl=63 time=0.173 ms
64 bytes from 10.10.14.1: icmp_seq=6 ttl=63 time=0.132 ms
^C
--- 10.10.14.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5054ms
rtt min/avg/max/mdev = 0.050/0.541/1.931/0.674 ms
root@mininet-vm:/home/mininet#
```



همانطور که مشخص است h2 در حال پینگ کردن h1 میباشد اما h3 بسته هایی با مبدأ h2 و مقصد خودش دریافت کرده است.

حال هاست h3 باید بسته ها را تغییر دهد و مقصد آنها را به h1 دوباره تغییر دهد که اینجا مشکلی بوجود خواهد آمد که بسته های جدید مبدأ h2 و مقصد h1 خواهند داشت و توسط روتر h4 دوباره به دست h3 خواهند رسید و درون یک لوب میافتیم.

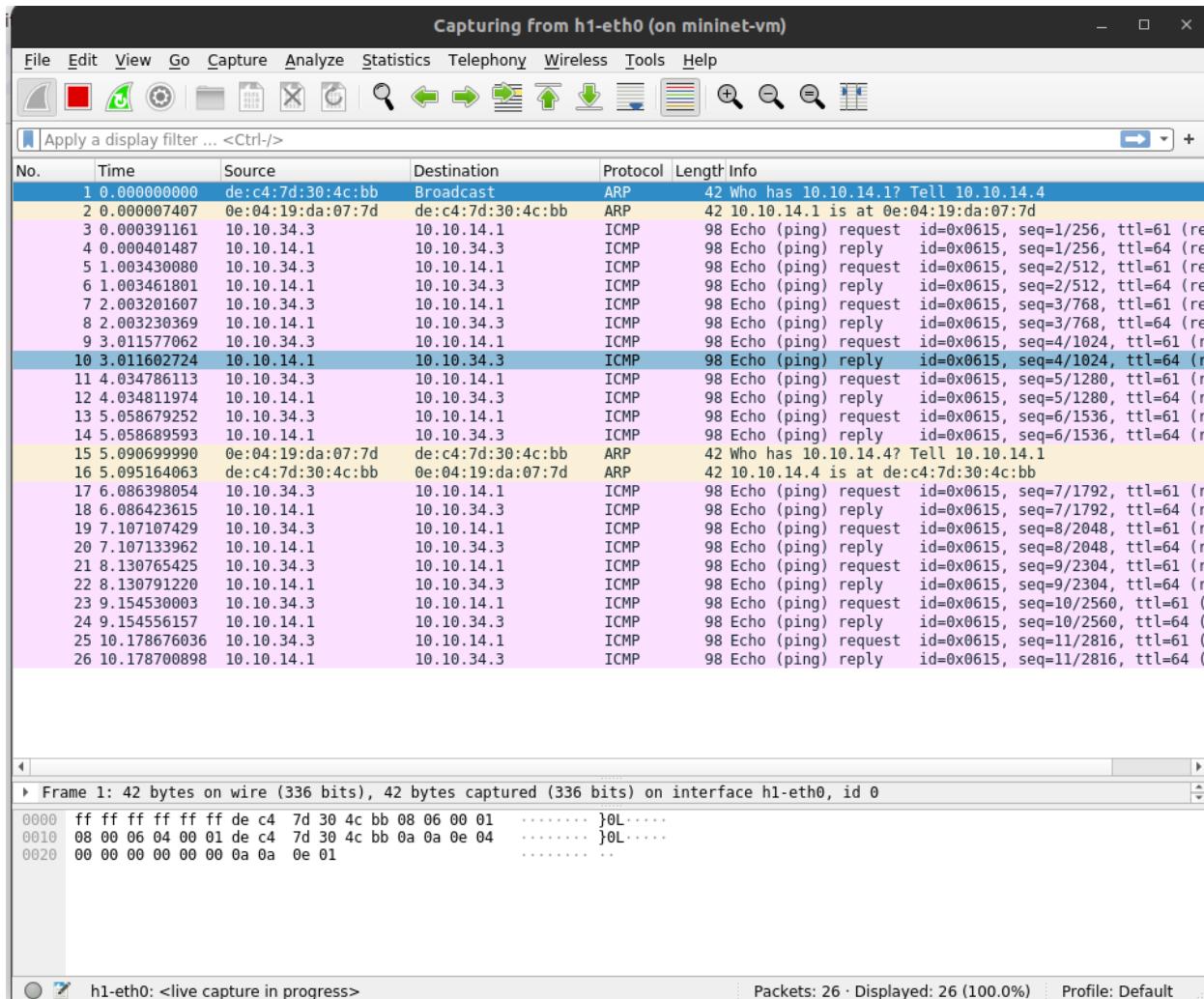
برای جلوگیری از این کار در h3 مبدأ بسته را نیز از h2 به h3 تغییر میدهیم. چون از جنس SNAT است باید از POSTROUTING استفاده کنیم.

در نهایت میتوانیم بینیم که بسته ها از مبدأ h3 به h1 خواهد رفت.

```
root@mininet-vm:/home/mininet# iptables -t nat -A POSTROUTING -p icmp -s 10.10.24.2 -d 10.10.14.1 -j SNAT --to 10.10.34.3
root@mininet-vm:/home/mininet# iptables -t nat -A PREROUTING -p icmp -s 10.10.24.2 -d 10.10.34.3 -j DNAT --to 10.10.14.1
root@mininet-vm:/home/mininet#
```

"Node: h2" (on mininet-vm)

```
root@mininet-vm:/home/mininet# ping 10.10.14.1
PING 10.10.14.1 (10.10.14.1) 56(84) bytes of data.
64 bytes from 10.10.14.1: icmp_seq=1 ttl=61 time=1.94 ms
64 bytes from 10.10.14.1: icmp_seq=2 ttl=61 time=3.27 ms
64 bytes from 10.10.14.1: icmp_seq=3 ttl=61 time=0.588 ms
64 bytes from 10.10.14.1: icmp_seq=4 ttl=61 time=0.151 ms
64 bytes from 10.10.14.1: icmp_seq=5 ttl=61 time=0.163 ms
64 bytes from 10.10.14.1: icmp_seq=6 ttl=61 time=0.063 ms
64 bytes from 10.10.14.1: icmp_seq=7 ttl=61 time=0.166 ms
64 bytes from 10.10.14.1: icmp_seq=8 ttl=61 time=0.154 ms
64 bytes from 10.10.14.1: icmp_seq=9 ttl=61 time=0.163 ms
64 bytes from 10.10.14.1: icmp_seq=10 ttl=61 time=0.164 ms
64 bytes from 10.10.14.1: icmp_seq=11 ttl=61 time=0.152 ms
^C
--- 10.10.14.1 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10180ms
rtt min/avg/max/mdev = 0.063/0.633/3.273/0.980 ms
root@mininet-vm:/home/mininet#
```



در اینجا h1 بدلیل RPF میتواند بسته هایی که مبدا آن h2 نیستند را drop کند پس باید مبدا بسته ها را نیز به h2 تغییر داد. برای این کار یک rule در h4 اضافه میکنیم.

```
"Node: h4" (on mininet-vm)
root@mininet-vm:/home/mininet# iptables -t nat -A PREROUTING -p icmp -s 10.10.24.2 -d 10.10.14.1 -j DNAT --to 10.10.34.3
root@mininet-vm:/home/mininet# iptables -t nat -A POSTROUTING -p icmp -s 10.10.34.3 -d 10.10.14.1 -j SNAT --to 10.10.24.2
root@mininet-vm:/home/mininet#
```

Capturing from h1-eth0 (on mininet-vm)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=1/256, ttl=61 (re)
2	0.000009339	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=1/256, ttl=64 (re)
3	1.002031748	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=2/512, ttl=61 (re)
4	1.002048927	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=2/512, ttl=64 (re)
5	2.003348313	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=3/768, ttl=61 (re)
6	2.003380695	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=3/768, ttl=64 (re)
7	3.015654863	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=4/1024, ttl=61 (re)
8	3.015681764	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=4/1024, ttl=64 (re)
9	4.039957422	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=5/1280, ttl=61 (re)
10	4.039984669	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=5/1280, ttl=64 (re)
11	5.063702358	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=6/1536, ttl=61 (re)
12	5.063728426	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=6/1536, ttl=64 (re)
13	5.095620017	0e:04:19:da:07:7d	de:c4:7d:30:4c:bb	ARP	42	Who has 10.10.14.4? Tell 10.10.14.1
14	5.100237424	de:c4:7d:30:4c:bb	0e:04:19:da:07:7d	ARP	42	Who has 10.10.14.1? Tell 10.10.14.4
15	5.100249327	0e:04:19:da:07:7d	de:c4:7d:30:4c:bb	ARP	42	10.10.14.1 is at 0e:04:19:da:07:7d
16	5.102997103	de:c4:7d:30:4c:bb	0e:04:19:da:07:7d	ARP	42	10.10.14.4 is at de:c4:7d:30:4c:bb
17	6.087712537	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=7/1792, ttl=61 (re)
18	6.087739053	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=7/1792, ttl=64 (re)
19	7.111509975	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=8/2048, ttl=61 (re)
20	7.111534398	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=8/2048, ttl=64 (re)
21	8.135702773	10.10.24.2	10.10.14.1	ICMP	98	Echo (ping) request id=0x06ab, seq=9/2304, ttl=61 (re)
22	8.135729439	10.10.14.1	10.10.24.2	ICMP	98	Echo (ping) reply id=0x06ab, seq=9/2304, ttl=64 (re)

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface h1-eth0, id 0

Ethernet II, Src: de:c4:7d:30:4c:bb (de:c4:7d:30:4c:bb), Dst: 0e:04:19:da:07:7d (0e:04:19:da:07:7d)

Internet Protocol Version 4, Src: 10.10.24.2, Dst: 10.10.14.1

Internet Control Message Protocol

0000 0e 04 19 da 07 7d de c4 7d 30 4c bb 08 00 45 00 . . . } . }0L . . E.

0010 00 54 51 38 40 00 3d 01 b2 5a 0a 18 02 0a 0a TQ8@= . Z . .

0020 0e 01 08 00 bd 7c 06 0b 00 01 03 06 54 65 00 00 . . . | . . Te . .

0030 00 00 18 e9 04 00 00 00 00 00 10 11 12 13 14 15 . . . . . . . . . .

0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 . . . . . . !#\$%

0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &(')\*+, - ./012345

0060 36 37 67

h1-eth0: <live capture in progress>

Packets: 22 · Displayed: 22 (100.0%) · Profile: Default

مشاهده میکنیم که مبدأ بسته ها را از h2 به h3 تغییر داده ایم و سپس h2 پینگ میکند و مشاهده میکنیم که در wireshark مربوط به h1 بسته هایی با مبدأ h2 و مقصد h1 دیده میشود.

در بالا سوال های 2 و 3 را با هم جواب دادم.

سوال 4:

خیر. چون پیام باید ابتدا به h3 برسد و اگر ip table هاست h3 را تغییر ندهیم نمیتوانیم پیام را به h1 فوروارد کنیم.

سوال 5:

بله میتواند متوجه شود.

با استفاده از ttl و rtt میتوانیم بفهمیم پکت در مسیر دیگری قرار گرفته است یا خیر. به طور معمول ttl باید 63 باشد ولی بعد از گذشتن از h3 میتوان مشاهده کرد که ttl آن به 61 کاهش پیدا کرد.

# Network devices

# OSI model

## 7. Application layer

Consists of application programs that use the network.

## 6. Presentation layer

Standardizes data presentation to the applications that use the network.

## 5. Session layer

Manages sessions between applications.

## 4. Transport layer

Provides end-to-end error detection and correction.

## 3. Network layer

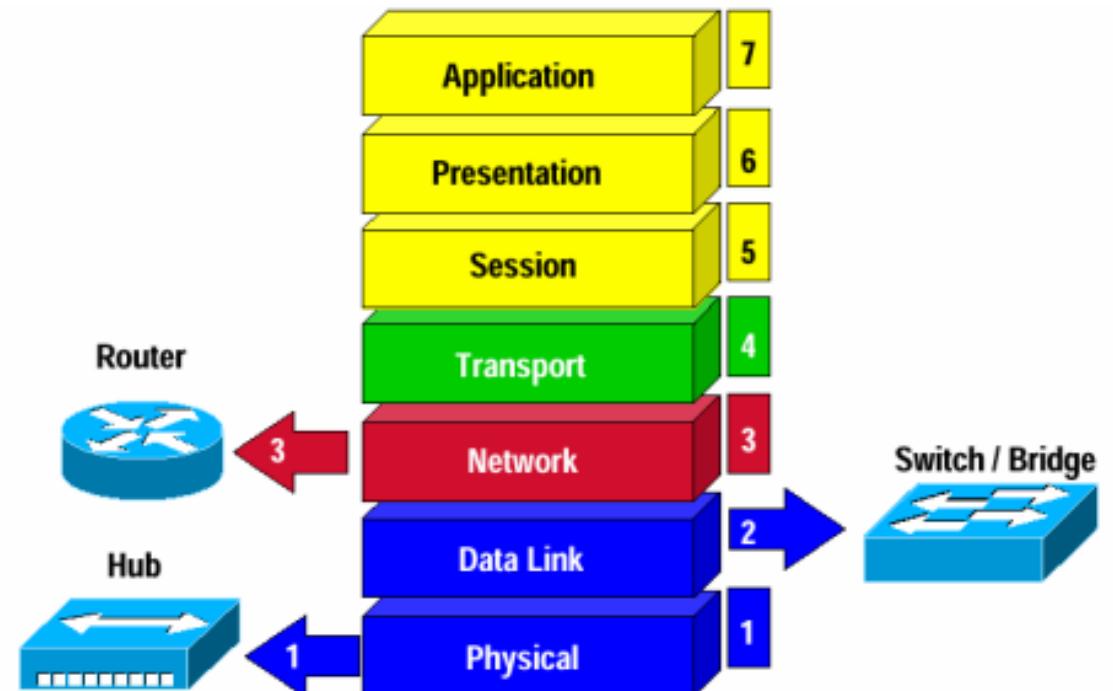
Decides which physical path the data will take.

## 2. Data link layer

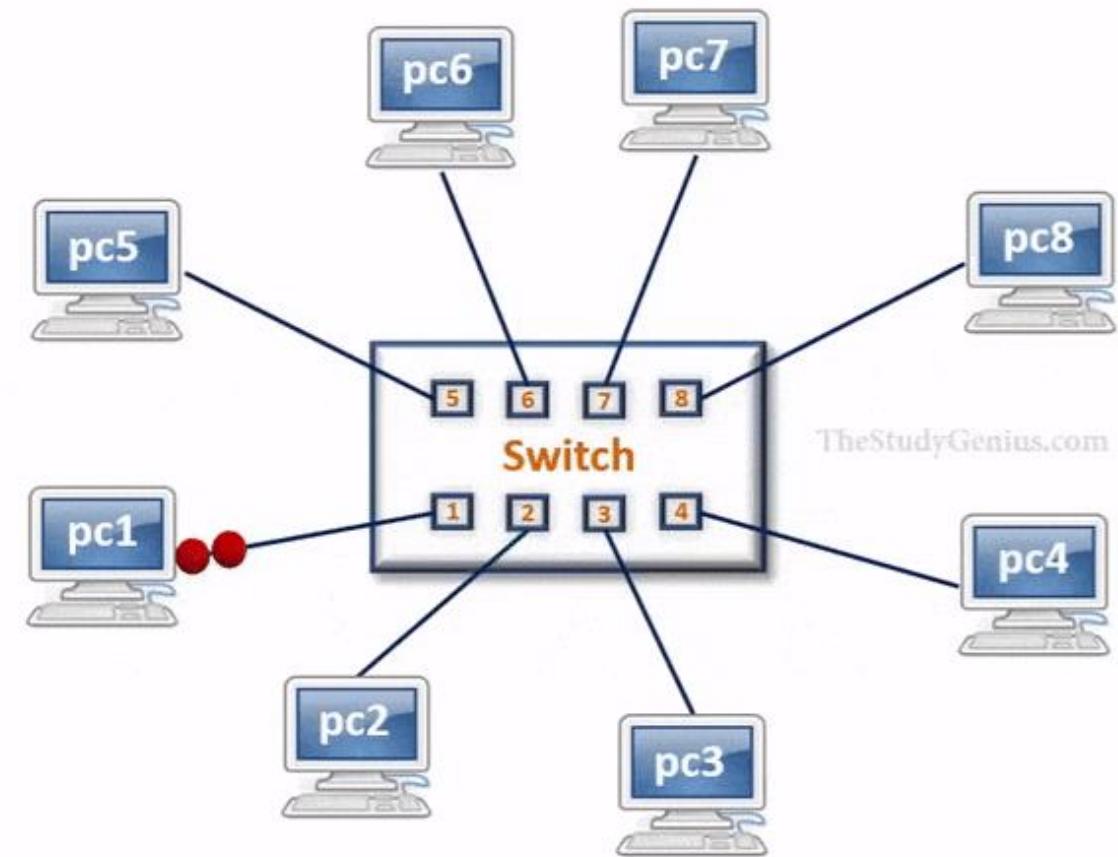
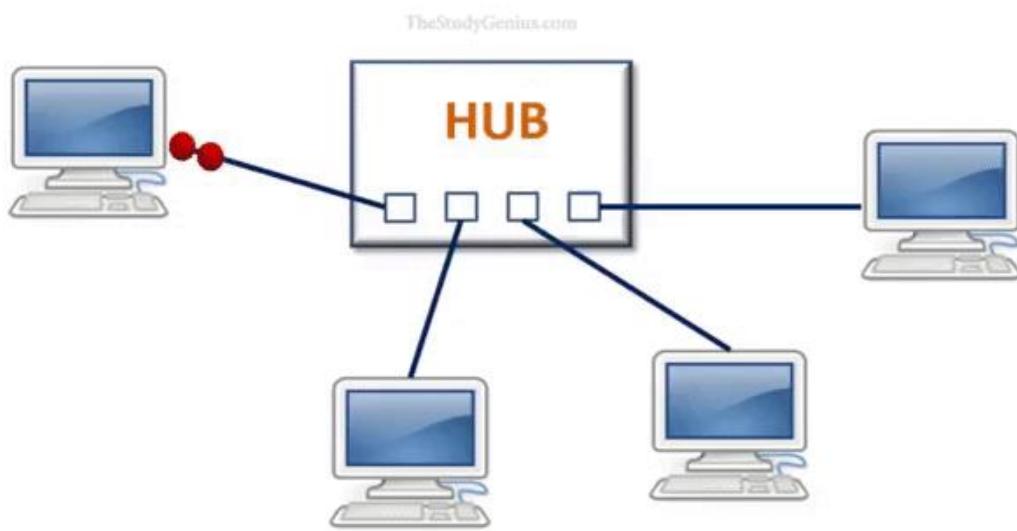
Provides reliable data delivery across the physical link.

## 1. Physical layer

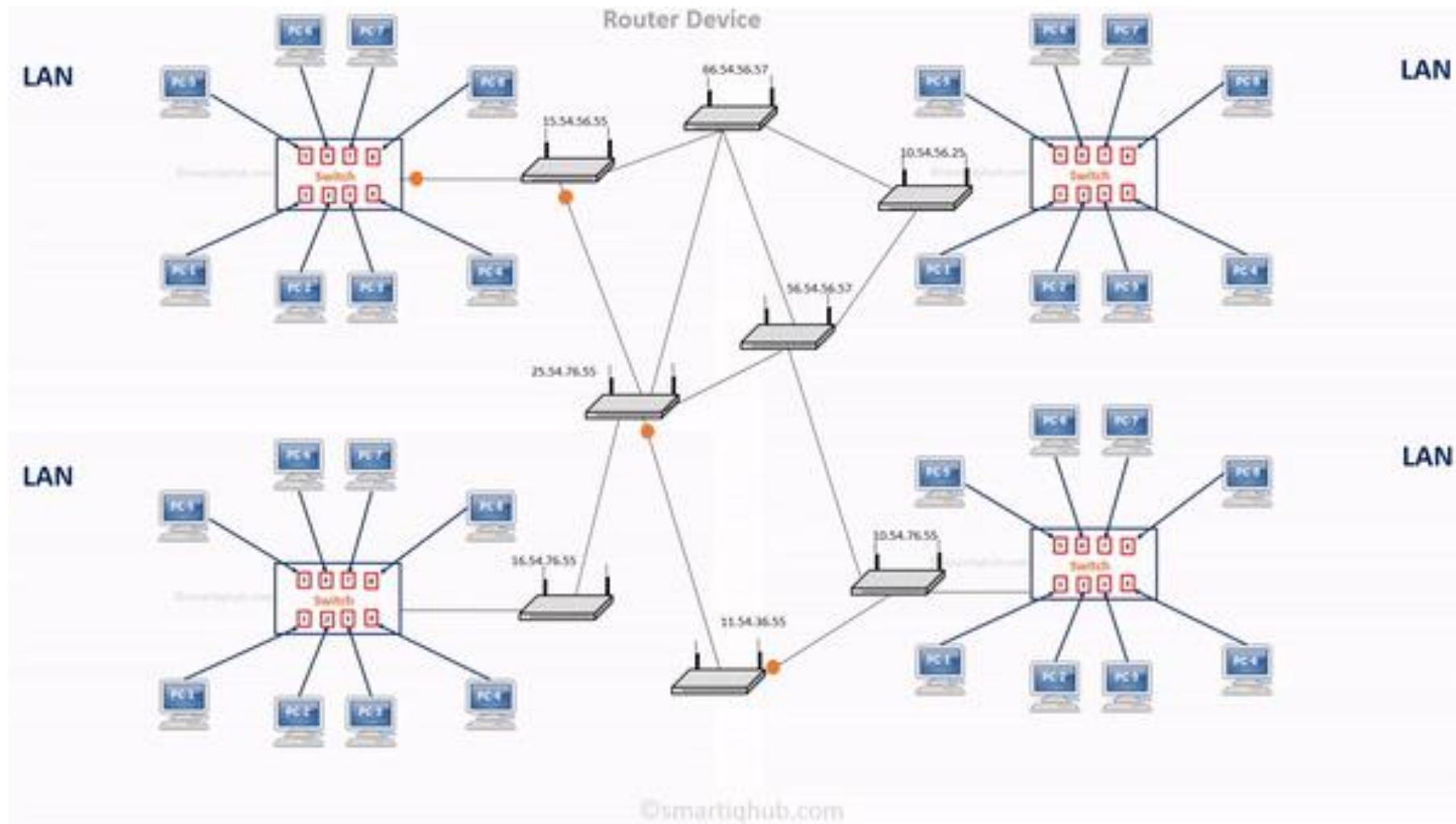
Defines the physical characteristics of the network media.



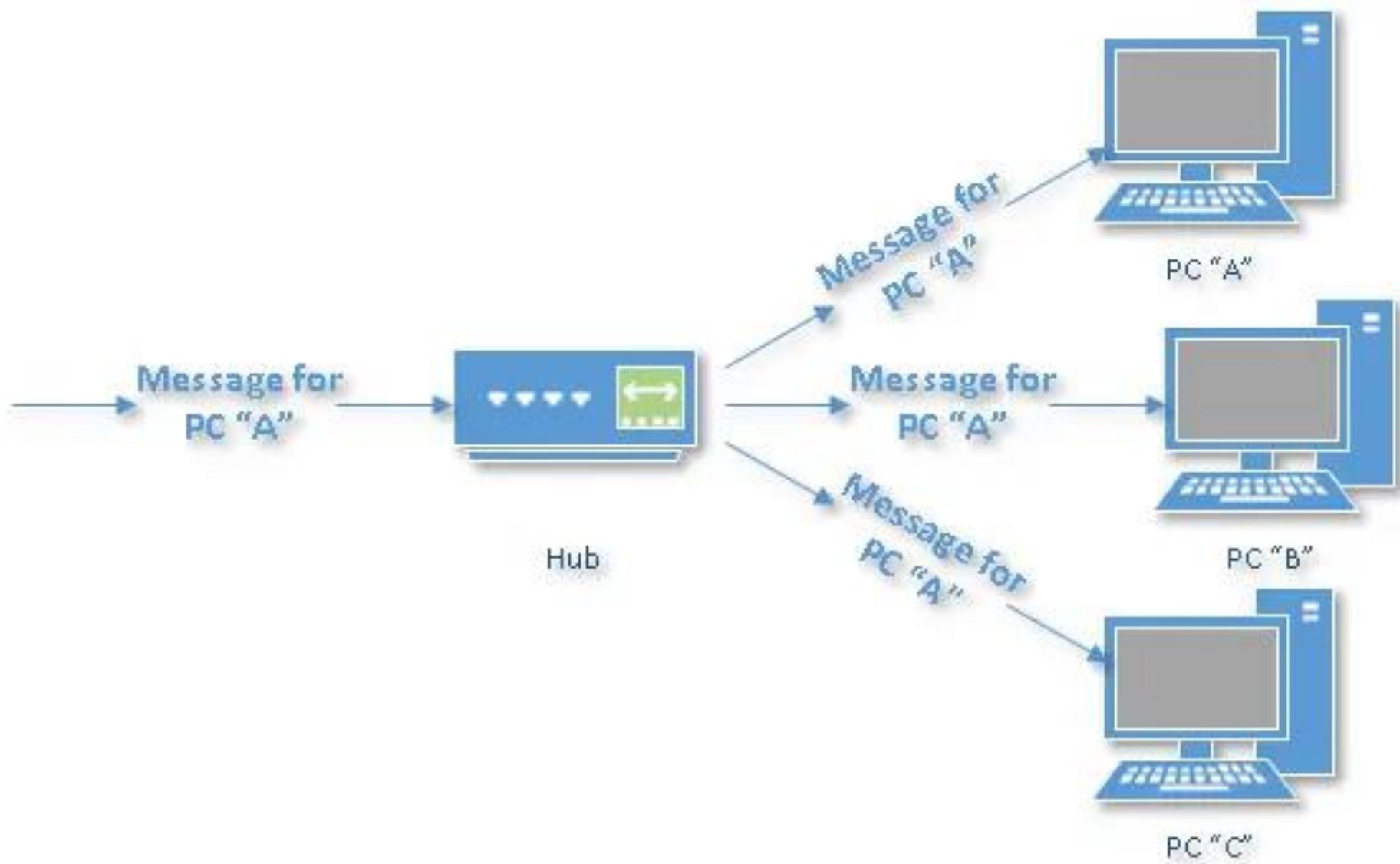
# Hub vs. Switch



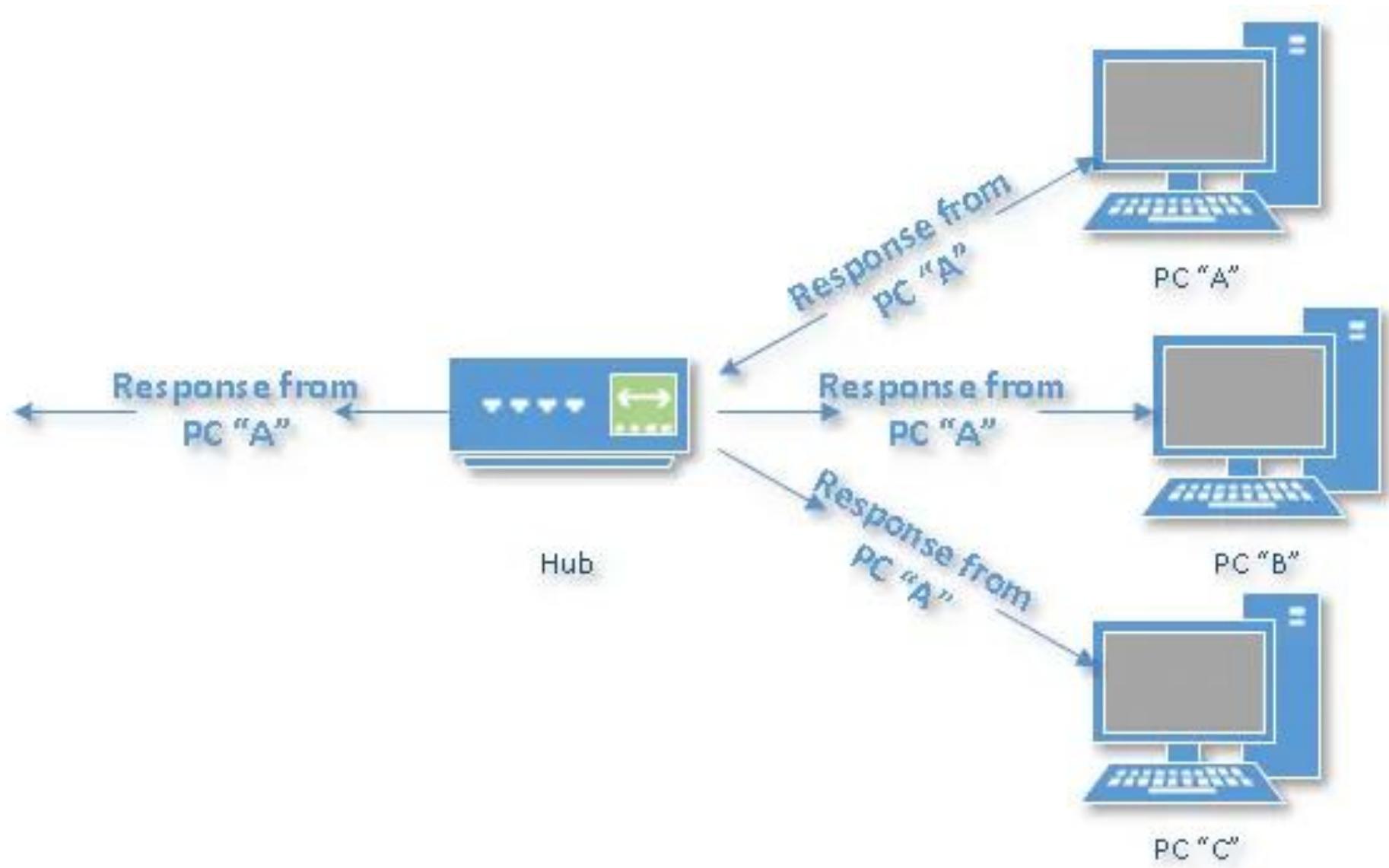
# Routers



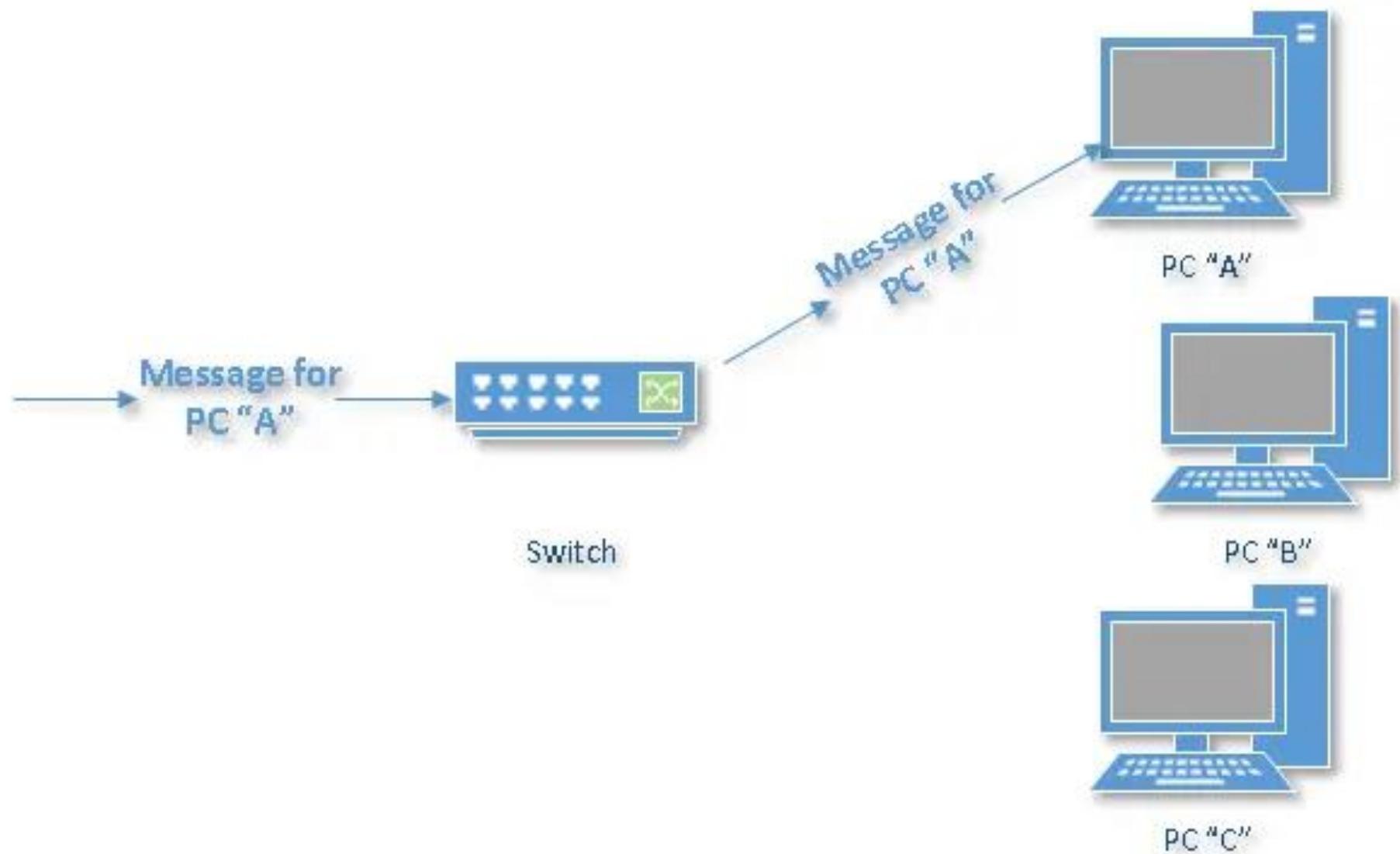
# Hub



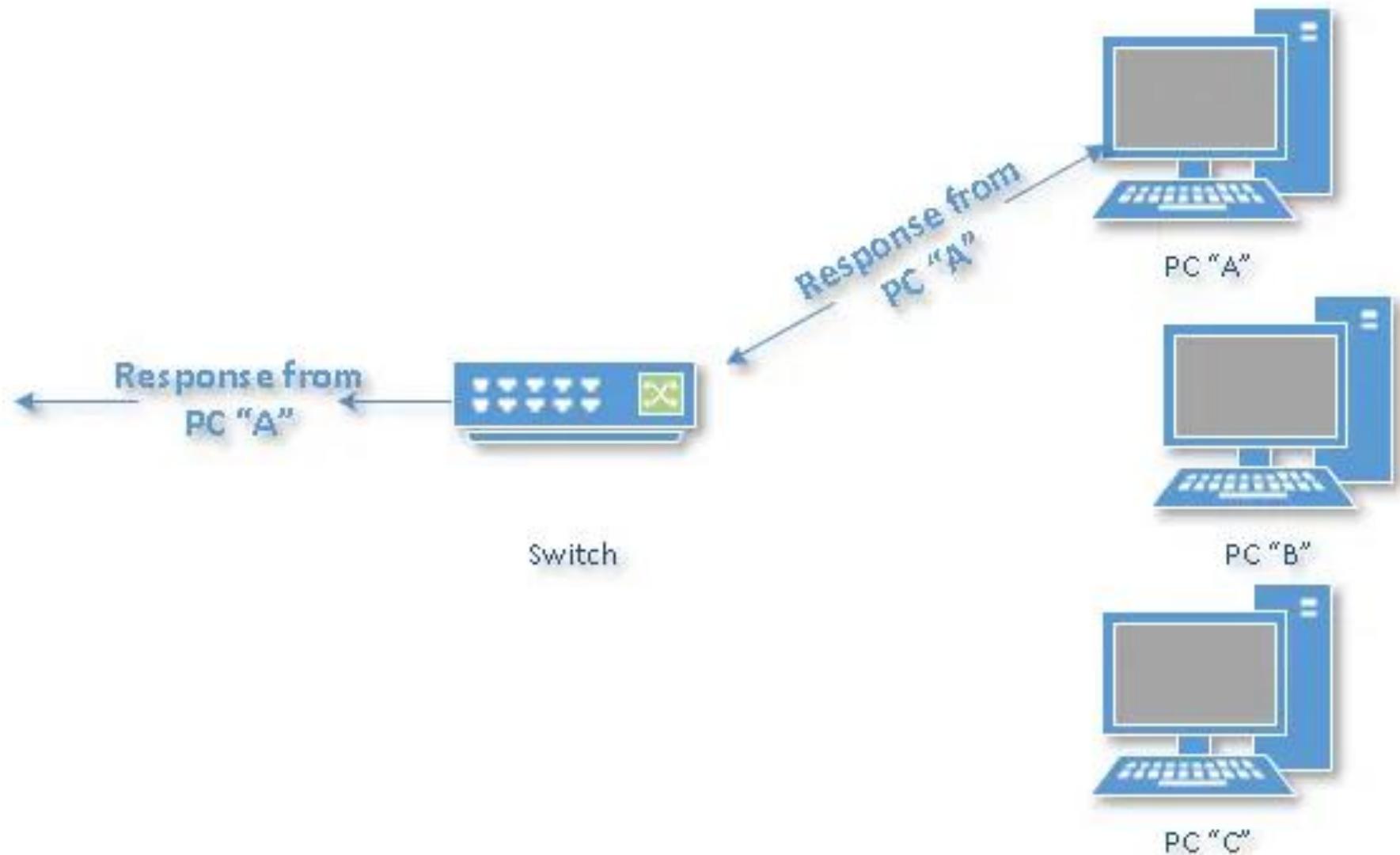
# Hub



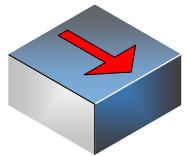
# Switch



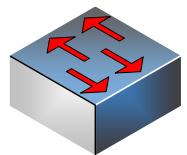
# Switch



# Network devices



- Hub
  - Broadcast packets



- Switch
  - Forward packet to the destination interface



- Router
  - Change source MAC address and then forward packet to the destination interface

# Comparison

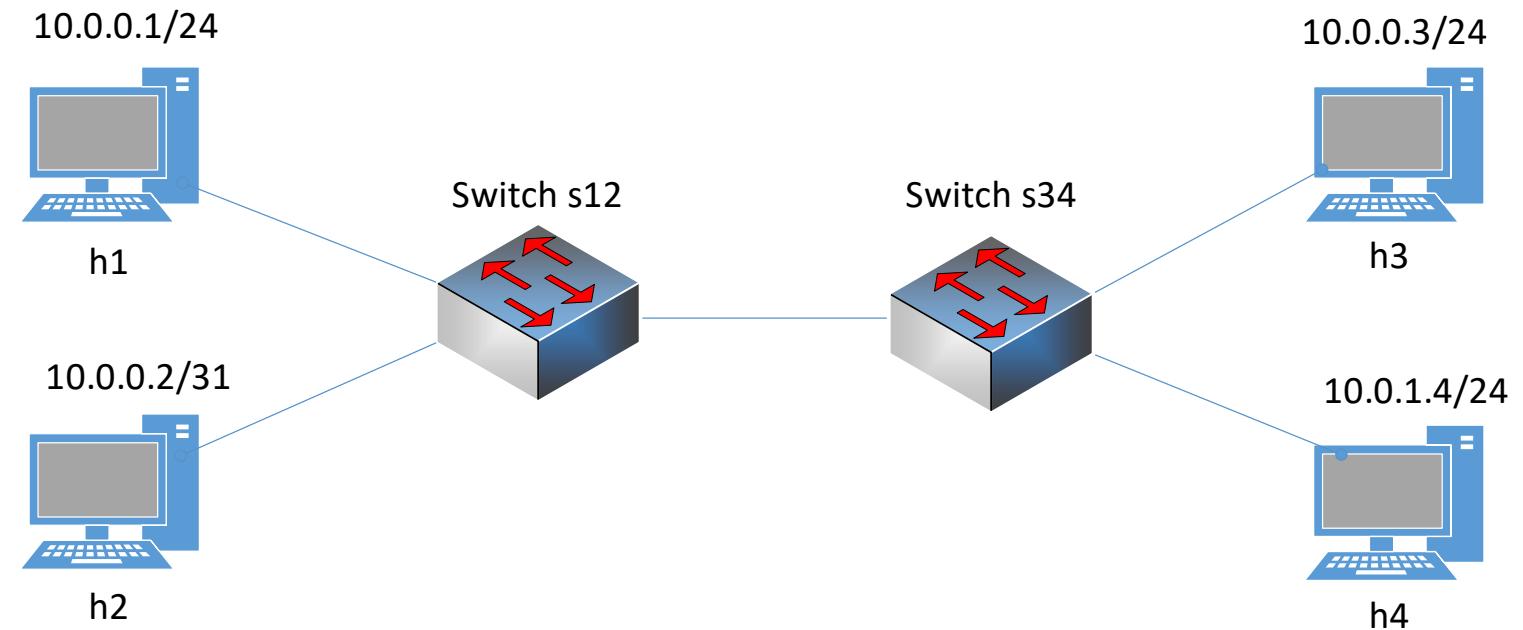
Hub	Switch	Router
HUB work on Physical Layer of OSI Model	Switch work on Data Link Layer of OSI Model	Router work on Network Layer of OSI Model
HUB is Broadcast Device	Switch is Unicast/Multicast Device	Router is a routing device use to create route for transmitting data packets
Hubs is use to connect device in the same network	Switch is use to connect devices in the same network	Router is use to connect two or more different network.
Hub sends data in the form of binary bits	Switch sends data in the form of frames	Router sends data in the form packets
Hub only works in half duplex	Switch works in full duplex	Router works in full duplex
Only one device can send data at a time	Multiple devices can send data at the same time	Multiple devices can send data at the same time
Hub does not store any mac address or IP address	Switch store MAC Address	Router stores IP address

# Hub function

- \$ cd Desktop/shared
- \$ sudo python lab4-1.py

- # ifconfig
- Lan1: h1, h2, h3
- Lan2: h4

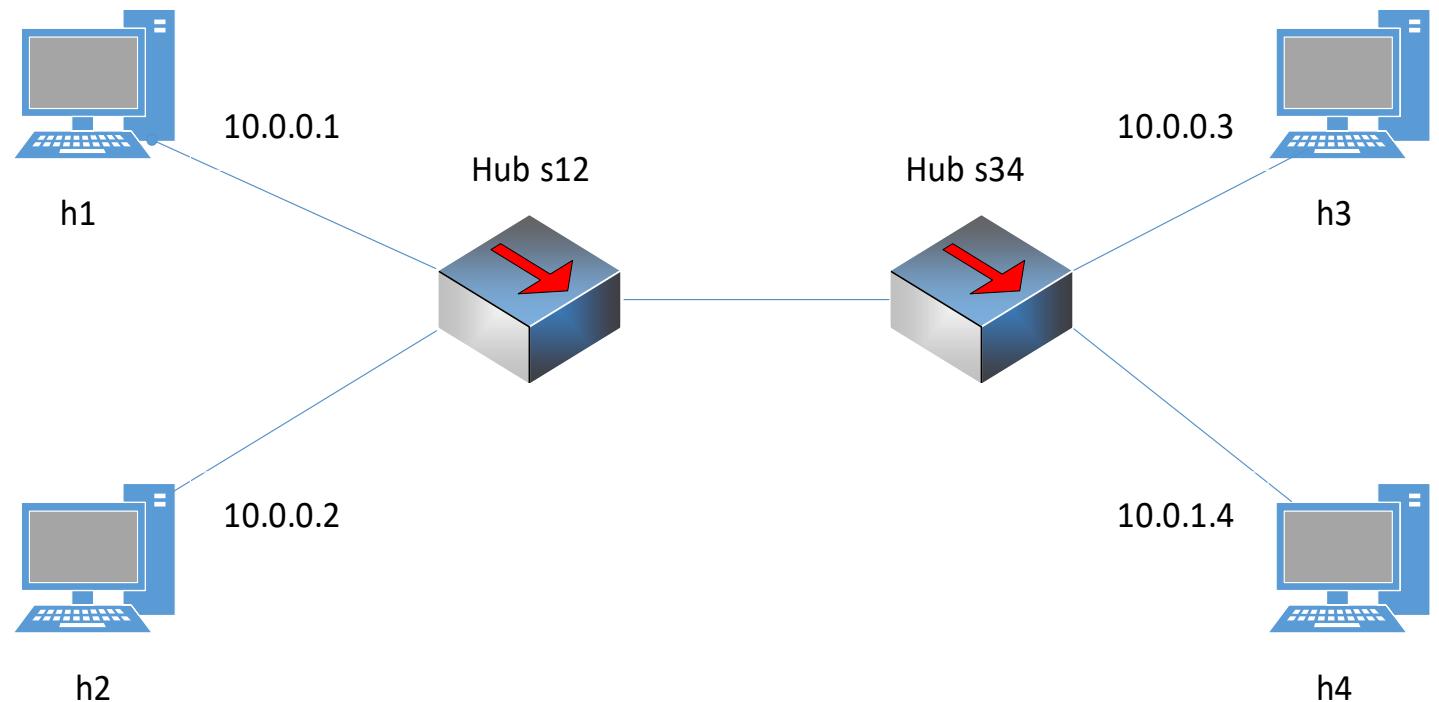
- Correct subnet mask of h2-eth0:
  - # ifconfig h2-eth0 10.0.0.2 netmask 255.255.255.0



# Hub function

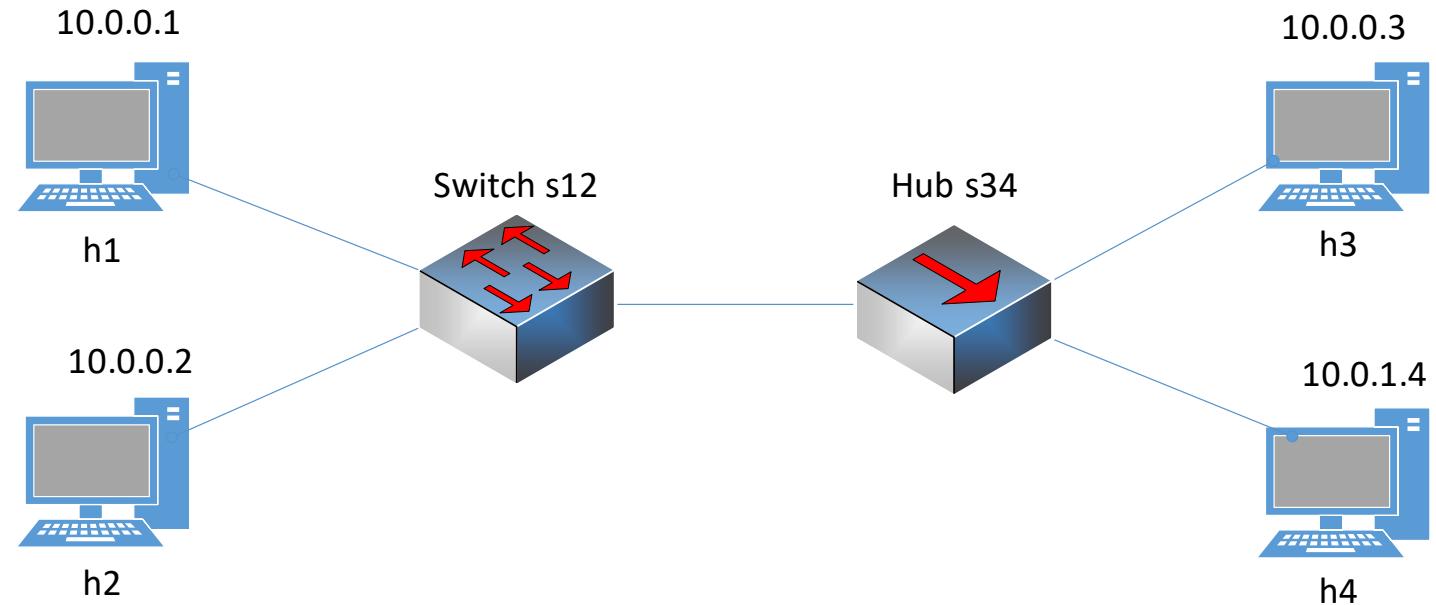
- Configure s12 and s34 as hubs:
  - mininet> sh ovs-ofctl add-flow s12 action=flood
  - mininet> sh ovs-ofctl add-flow s34 action=flood

- h1 ping h2



# Switch function

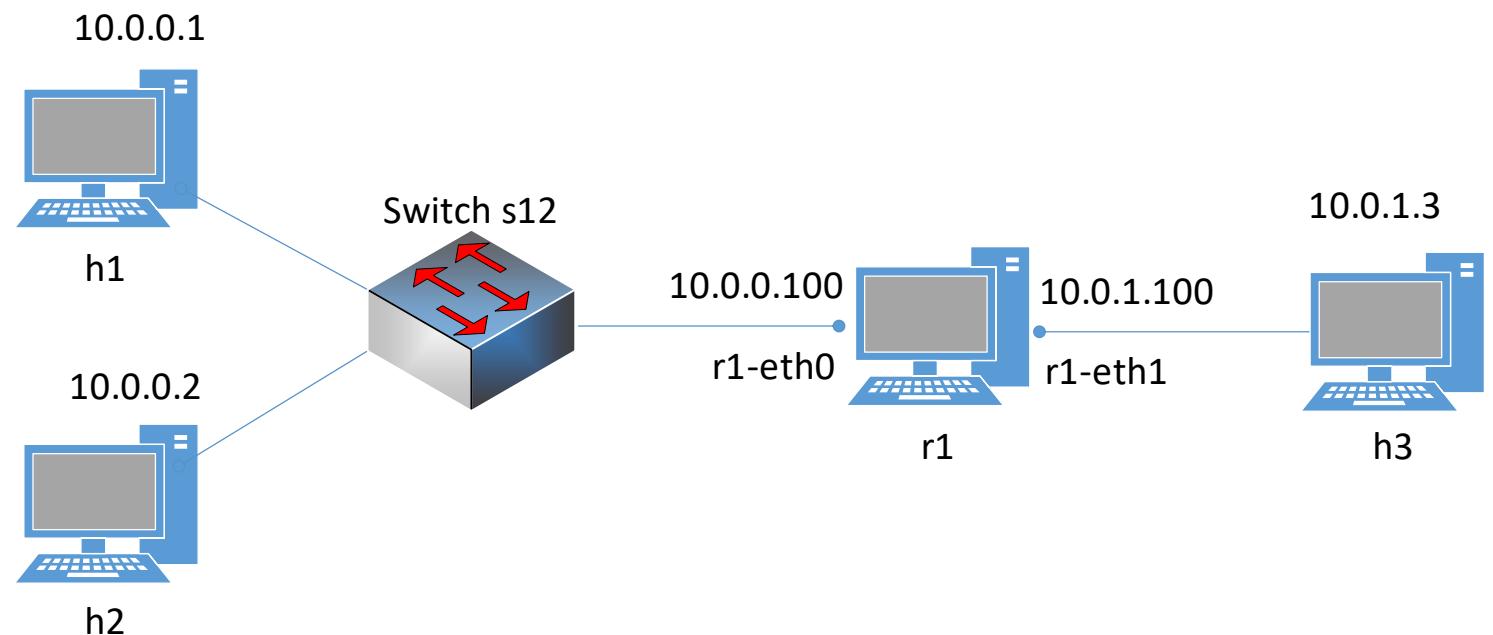
- Return s12 to a switch:
  - mininet> sh ovs-ofctl add-flow s12 action=normal
- Delete a specified entry of ARP cache:
  - # arp -d 10.0.0.2
- h1 ping h2
- h1 ping h4
- h1 ping h3



# Router function

- mininet> exit
- \$ sudo mn -c
- \$ sudo python lab4-2.py

- mininet> pingall

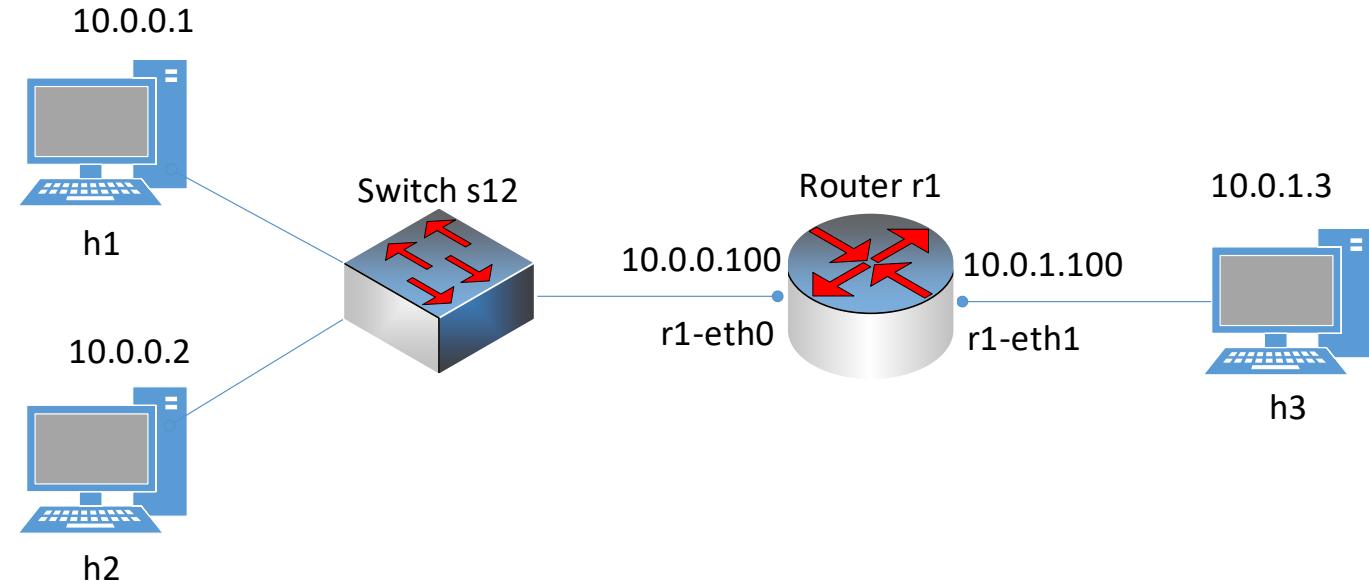


# Router function

- Enable IP forwarding on r1:
  - `# echo 1 > /proc/sys/net/ipv4/ip_forward`
- `mininet> pingall`

- Show gateway of each host:
  - `# ip route`

- Correct gateway:
  - `# ip route del default via 10.0.0.101`
  - `# ip route add default via 10.0.0.100`



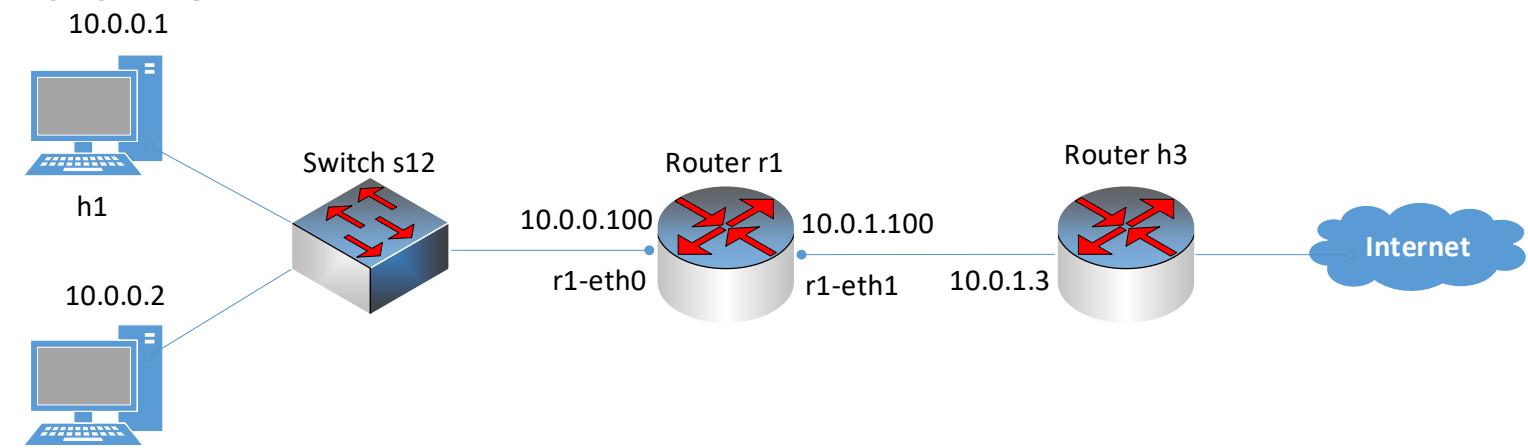
# Routing with multiple hops

- For h3:
  - `# echo 1 > /proc/sys/net/ipv4/ip_forward`
  - `# ip route del default via 10.0.1.100`

- For r1:
  - `# ip route add default via 10.0.1.3`

- h1 ping h3
- h2 ping h3

- `# ip route add 10.0.0.0/24 via 10.0.1.100`



## آزمایشگاه شبکه

### آزمایش ۴: بررسی نحوه کار تجهیزات شبکه در لایه‌های ۱، ۲ و ۳

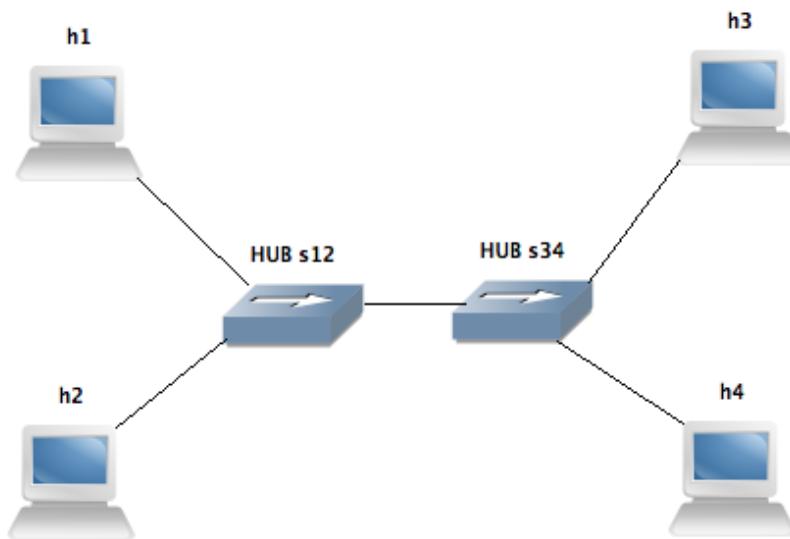
هدف این آزمایش، درک تفاوت در نحوه کارکرد تجهیزات شبکه‌ای لایه‌های ۱، ۲ و ۳ است.

#### الف) آشنایی با نحوه کارکرد هاب (hub)

هاب یک سیستم میانجی تک لایه‌ای است که بیت (سمبل)‌هایی را که از یکی از پورت‌هایی دریافت می‌کند، روی سایر پورت‌هاییش تکرار می‌کند. در این بخش، دقیق‌تر به بررسی عملکرد این تجهیز شبکه‌ای خواهیم پرداخت.

- یک ترمینال در VM خود باز کنید و اسکریپت lab4-1.py را اجرا نمایید.

- با اجرای اسکریپت یادشده، شبکه نمایش داده شده در شکل ۱ ایجاد خواهد شد.



شکل ۱- پیکربندی شبکه با دو هاب

- در خط دستور mininet، از دستور زیر برای تبدیل عملکرد دو سوئیچ s12 و s34 به هاب استفاده کنید. در هر مورد، عبارت switch name را با نام سوئیچ‌های مورد نظر جایگزین نمایید.

```
mininet> sh ovs-ofctl add-flow switch name action=flood
```

- در تопولوژی مورد نظر ماشین‌های h1، h2 و h3 باشد روی subnet با آدرس 10.0.0.0/24 واقع شوند و چهارمین بایت آدرس IP آنها هم به ترتیب ۱، ۲ و ۳ باشد. ماشین h4 هم باید دارای آدرس IP به فرم 10.0.1.4/24 باشد. پیکربندی آدرس‌های IP برای هریک از ماشین‌ها را در پنجره ترمینال مربوط به خودشان بررسی نموده و در صورت وجود اشکال در پیکربندی، آن را با استفاده از دستور مناسب، مرتفع نمایید.

سؤال ۱- آیا ماشینی هست که دارای پیکربندی اشتباه باشد؟ از چه دستوری برای رفع مشکل استفاده می‌کنید؟

برنامه WireShark را روی هر چهار ماشین راهاندازی نموده و شروع به capture کردن روی اینترفیس eth0 همه ماشین‌ها بنمایید.

از ماشین h1 ( تنها یک مرتبه ) ماشین h2 را پینگ کنید.

سؤال ۲- آیا تفاوتی میان ترافیک capture شده توسط چهار ماشین در WireShark ملاحظه می‌کنید؟ پاسخ خود را توضیح دهید.

ب) آشنایی با نحوه کارکرد سوئیچ (switch)

سوئیچ یک عنصر ارتباطی در سطح لایه ۲ است که قادر می‌باشد از طریق اخذ تصمیمات فورواردینگ (مبتنی بر آدرس‌های MAC) محدوده یک شبکه محلی (LAN) را گسترش دهد. در این بخش، می‌خواهیم با نحوه کار سوئیچ آشنا شویم.

ابتدا با دستور مناسب، کارکرد s12 را طوری تغییر می‌دهیم که به صورت یک سوئیچ عمل نماید. برای این منظور، از دستوری شبیه به زیر استفاده نمایید:

```
mininet> sh ovs-ofctl add-flow switch name action=normal
```

توجه کنید که s12 و s34 را ابتدا به صورت سوئیچ ایجاد کرده بودیم، اما در بخش قبلی با استفاده از دستورات OpenFlow، کارکرد آنها را تغییر دادیم تا به صورت هاب عمل نمایند. بعداً در آزمایش‌های آتی با این قبیل دستورات آشنا خواهید شد.

برای آزمایش پیکربندی سوئیچ بار دیگر برنامه WireShark را جهت شنود روی هر چهار ماشین راهاندازی نموده، محتوای کش ARP در h1 را پاک کنید و مجدداً h2 را ( تنها یک مرتبه ) از h1 پینگ نمایید.

سؤال ۳- انواع مختلف بسته‌هایی را که در ماشین‌های مختلف ملاحظه می‌کنید، تشریح نمایید.

سؤال ۴- توضیح دهید که چه تفاوتی با حالتی که دو هاب داشتیم، به وجود آمده است.

سؤال ۵- از h1 ماشین h4 را ( تنها یک مرتبه ) پینگ کنید و ترافیک مورد مشاهده و همینطور استنباط خود را تشریح نمایید.

حال روی اینترفیس‌های h1, h2, h3 و h4 تمرکز نموده و از h1 به h3 ( تنها یک مرتبه ) پینگ کنید.

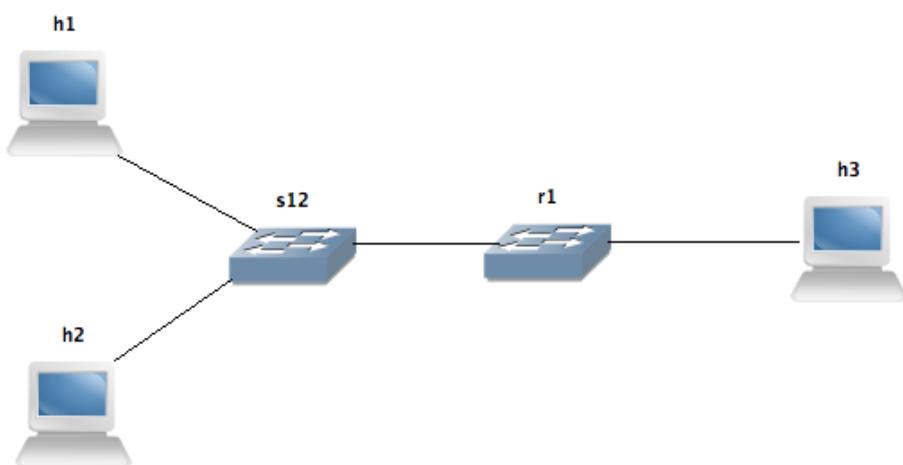
سؤال ۶- بسته‌های دریافتی روی h2 را ملاحظه کرده و مشاهدات خود را تشریح نمایید.

سؤال ۷- بسته‌های ارسالی توسط **h1** با آنها بی که توسط **h3** دریافت می‌شوند، مقایسه نمایید (به ویژه به لحاظ آدرس‌های **MAC** مبدأ و مقصد آنها). وجود تشابه و تمایز آنها (در صورت وجود) چیست؟ توضیح دهید.

### ج) آشنایی با نحوه کارکرد روتر (router)

پیش‌تر در قالب آزمایش‌های ۱ و ۲، با روتر آشنا شده‌ایم. در این بخش بنا داریم قدری جزئی‌تر در مورد فرآیند مسیریابی یک بسته تحقیق نماییم.

- از برنامه Mininet خارج شده و توپولوژی قبلی را `clean up` کنید (sudo mn -c).
- حال، اسکریپت با عنوان `lab4-2.py` را اجرا نمایید.
- مطابق شکل ۲، توپولوژی جدید از سه ماشین، یک روتر و یک سوئیچ تشکیل شده است.



شکل ۲- پیکربندی شبکه با یک سوئیچ و یک روتر

یک تست همبندی انجام دهید تا تعیین کدام ماشین‌ها می‌توانند با هم‌دیگر ارتباط برقرار کنند. برای این منظور با اجرای دستور `pingall`، از هر ماشین به هر ماشین دیگر پینگ نمایید. با اجرای این دستور، همبندی با روتر هم تست می‌شود.

### سؤال ۸- چه درصدی از بسته‌ها **drop** می‌شوند؟

### سؤال ۹- کدام **host**ها قادر به برقراری ارتباط با هم نیستند؟

- برای رفع مشکل همبندی، اسکریپت `lab4-2.py` را باز کرده و آن را بررسی کنید.

### سؤال ۱۰- از چه **subnet mask**‌ای در سرتاسر فایل استفاده شده است؟

سؤال ۱۱- اینترفیس‌های روتر **r1** و آدرس‌های IP نظیر آنها چیست؟

سؤال ۱۲- آیا می‌توانید اشتباهات پیکربندی را بیابید؟ دستورات لازم برای اعمال تصحیحات را بنویسید.

- برای ادامه کار، حتماً اشتباهات یافته شده را تصحیح کرده و مجدداً تست همبندی را با `pingall` انجام دهید.

سؤال ۱۳- در صد بسته‌های **drop** شده چقدر است؟

- با راهاندازی **WireShark**، ترافیک هر دو اینترفیس روتر **r1** را `capture` نمایید. از **h1** یک مرتبه **h3** را پینگ کنید.

سؤال ۱۴- هنگام مسیریابی بسته‌ها از اینترفیس **r1-eth0** و اینترفیس **r1-eth1** چه تغییری در بسته‌های IP رخ می‌دهد؟

سؤال ۱۵- هدف از اعمال این تغییرات چیست؟

---

د) مسیریابی چند-گامه

در این بخش، می‌خواهیم مشاهده کنیم که با اضافه کردن یک روتر دوم به شبکه چه اتفاقی می‌افتد.

- آدرس **IP** مربوط به روتر **r1** را آدرس **default gateway** مربوط به ماشین **h3** قرار دهید.
- در **h3** **default gateway** را حذف کرده و قابلیت **forwarding** را هم در آن فعال کنید.

سؤال ۱۶- دستورات لازم برای منظور فوق را بنویسید.

- با استفاده از **WireShark**، اینترفیس **eth0** از ماشین‌های **h3** و **r1** را مانیتور کنید. سعی کنید از **h1** به **h3** و همینطور از **h2** به **h3** پینگ نمایید (تنها یک مرتبه).

سؤال ۱۷- بر اساس شنود **WireShark**، توضیح دهید که پینگ‌های فوق چرا کار نمی‌کنند؟!

سؤال ۱۸- با فرض اینکه بعداً از **h3** خواهیم خواست که به عنوان **gateway** دسترسی به اینترنت عمل کند، یک تک‌دستور بنویسید که مشکل پینگ از **h1** و **h2** به سوی **h3** را حل کند. این تک‌دستور را روی کدام ماشین اجرا می‌نمایید؟

- مجدداً از **h1** و **h2** ماشین **h3** را (یک مرتبه) پینگ کنید و مطمئن شوید که توانسته‌اید مشکل قبل را برطرف نمایید.

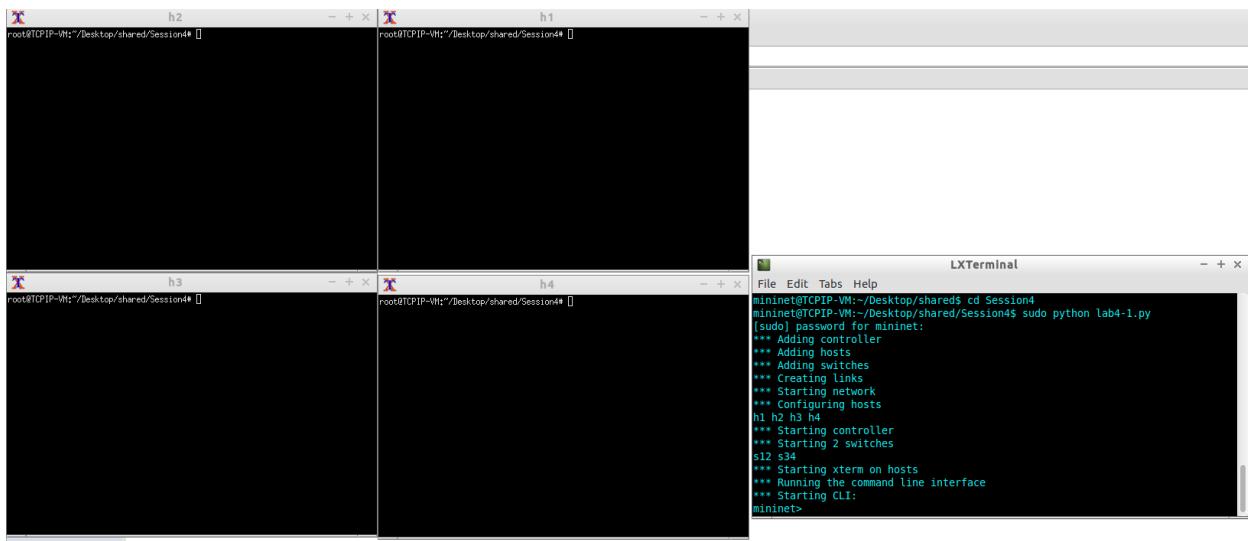
## گزارشکار آزمایش چهارم آزمایشگاه شبکه های کامپیوتري: بررسی نحوه کار تجهیزات شبکه در لایه های ۱، ۲ و ۳

سیده شکیبا انارکی فیروز - ۹۹۴۴۲۰۴۷

بهاره کاووسی نژاد - ۹۹۴۳۱۲۱۷

(الف) آشنایی با نحوه کارکرد هاب (hub)

یک ترمینال در VM خود باز کنید و اسکریپت lab4-1.py را اجرا نمایید.



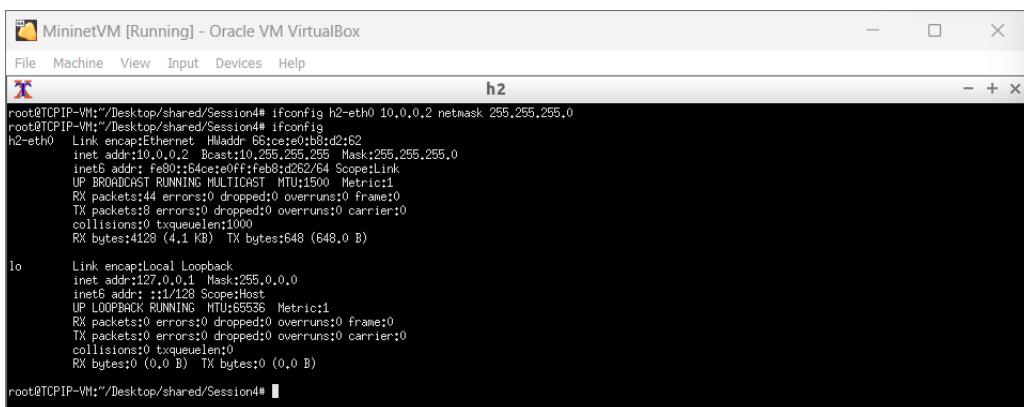
تبدیل سوئیچ های s12 و s34 به هاب:

```
mininet> sh ovs-ofctl add-flow s12 action=flood
mininet> sh ovs-ofctl add-flow s34 action=flood
mininet>
```

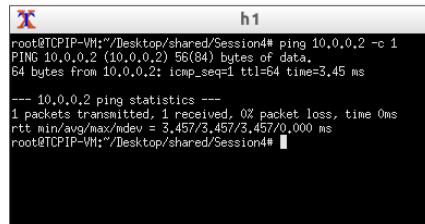
سؤال ۱- آیا ماشینی هست که دارای پیکربندی اشتباه باشد؟ از چه دستوری برای رفع مشکل استفاده می کنید؟

بله، مقدار netmask برای h2 درست نیست. به منظور تصحیح آن از دستور زیر استفاده می کنیم:

ابتدا با دستور زیر مقدار subnet mask را برای h2 درست می کنیم:



حال از ماشین h1 تنها یک بار ماشین h2 را به صورت زیر ping می کنیم:

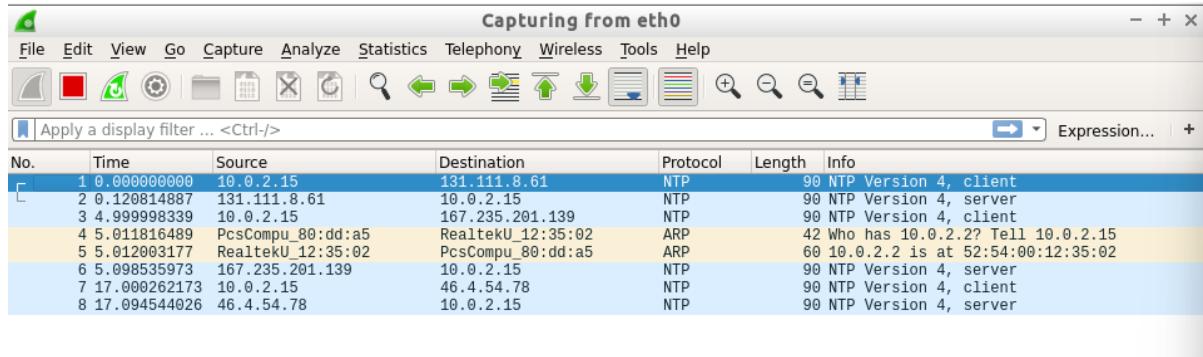


```

root@TCPiP-VM:~/Desktop/shared/Session4# ping 10.0.0.2 -c 1
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.45 ms
--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.457/3.457/3.457/0.000 ms
root@TCPiP-VM:~/Desktop/shared/Session4#

```

H1:

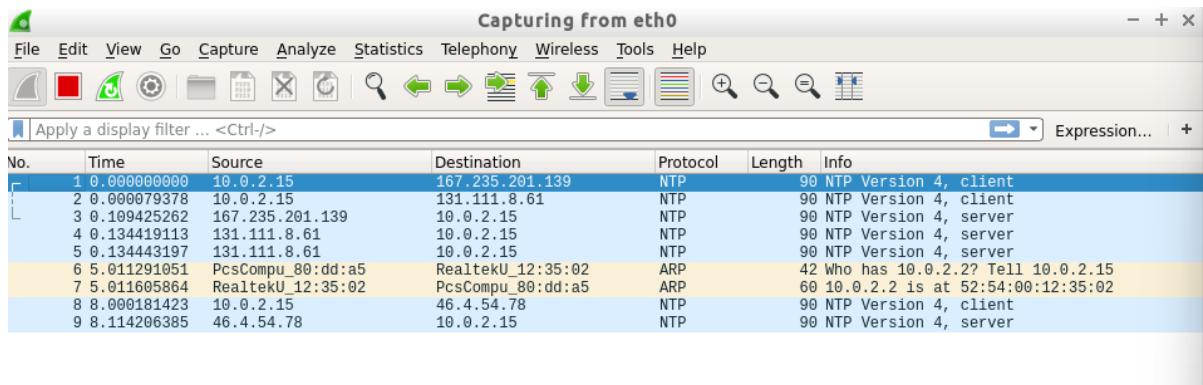


Capturing from eth0

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	131.111.8.61	NTP	90	NTP Version 4, client
2	0.120814887	131.111.8.61	10.0.2.15	NTP	90	NTP Version 4, server
3	4.99999839	10.0.2.15	167.235.201.139	NTP	90	NTP Version 4, client
4	5.011816489	PcsCompu_80:dd:a5	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
5	5.012003177	PcsCompu_80:dd:a5	RealtekU_12:35:02	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
6	5.098535973	167.235.201.139	10.0.2.15	NTP	90	NTP Version 4, server
7	17.000262173	10.0.2.15	46.4.54.78	NTP	90	NTP Version 4, client
8	17.094544626	46.4.54.78	10.0.2.15	NTP	90	NTP Version 4, server

H2:

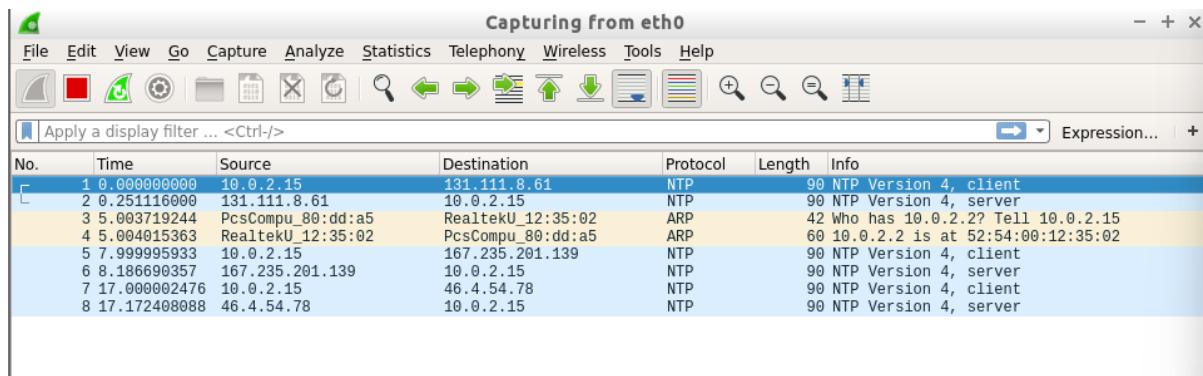


Capturing from eth0

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	167.235.201.139	NTP	90	NTP Version 4, client
2	0.000079378	10.0.2.15	131.111.8.61	NTP	90	NTP Version 4, server
3	0.109425262	167.235.201.139	10.0.2.15	NTP	90	NTP Version 4, server
4	0.134419113	131.111.8.61	10.0.2.15	NTP	90	NTP Version 4, server
5	0.134443197	131.111.8.61	10.0.2.15	NTP	90	NTP Version 4, server
6	5.011291051	PcsCompu_80:dd:a5	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
7	5.011605864	RealtekU_12:35:02	PcsCompu_80:dd:a5	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
8	8.000181423	10.0.2.15	46.4.54.78	NTP	90	NTP Version 4, client
9	8.114206385	46.4.54.78	10.0.2.15	NTP	90	NTP Version 4, server

H3:

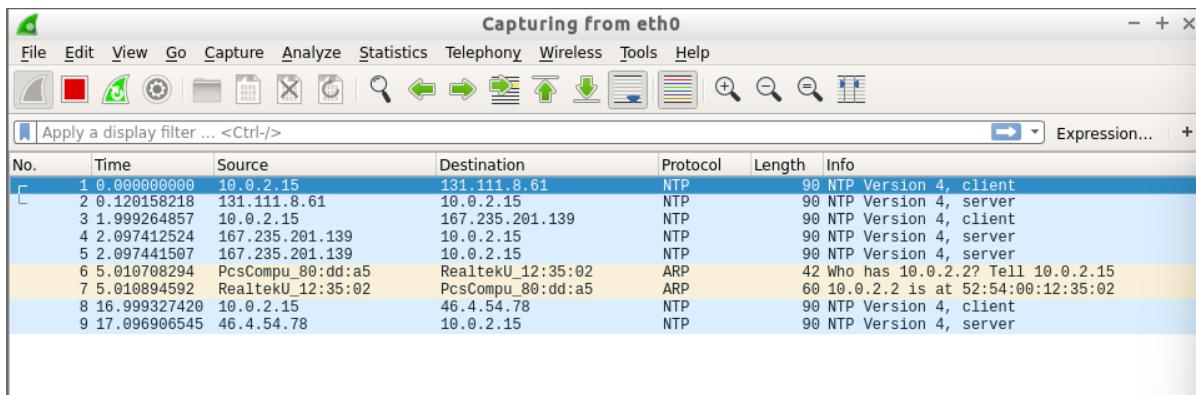


Capturing from eth0

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	131.111.8.61	NTP	90	NTP Version 4, client
2	0.251116000	131.111.8.61	10.0.2.15	NTP	90	NTP Version 4, server
3	5.003719244	PcsCompu_80:dd:a5	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
4	5.004015363	RealtekU_12:35:02	PcsCompu_80:dd:a5	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
5	5.999995933	10.0.2.15	167.235.201.139	NTP	90	NTP Version 4, client
6	6.8.186690357	167.235.201.139	10.0.2.15	NTP	90	NTP Version 4, server
7	7.17.000002476	10.0.2.15	46.4.54.78	NTP	90	NTP Version 4, client
8	7.17.172408088	46.4.54.78	10.0.2.15	NTP	90	NTP Version 4, server

H4:



همانطور که مشاهده می شود بسته های ارسال شده که شامل ARP، ICMP Request، ARP Reply و ICMP Request هستند در واقع ارتباط همه ماشین ها با هم برقرار است.

سوال ۲- آیا تفاوتی میان ترافیک capture شده توسط چهار ماشین در Wireshark ملاحظه می کنید؟ پاسخ خود را توضیح دهید.

خیر؛ تفاوتی میان ترافیک های capture شده نیست چرا که در این شبکه برای اتصال از hub استفاده شده است. hub هر بسته ای را که به دستش برسد به سایر پورتهایش ارسال می کند، بنابراین با ارسال ping از h1 به h2، پیام capture شده توسط تمامی ماشین ها یکسان است.

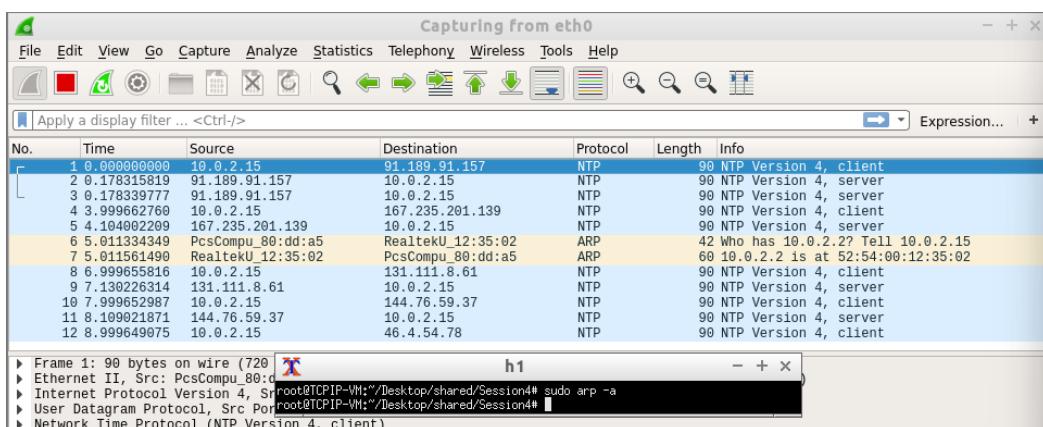
(b) آشنایی با نحوه کارکرد سوئیچ (switch)

ابتدا با دستور مناسب، کارکرد 512 را طوری تغییر می دهیم که به صورت یک سوئیچ عمل نماید. برای این منظور، از دستوری شبیه به زیر استفاده نمایید:

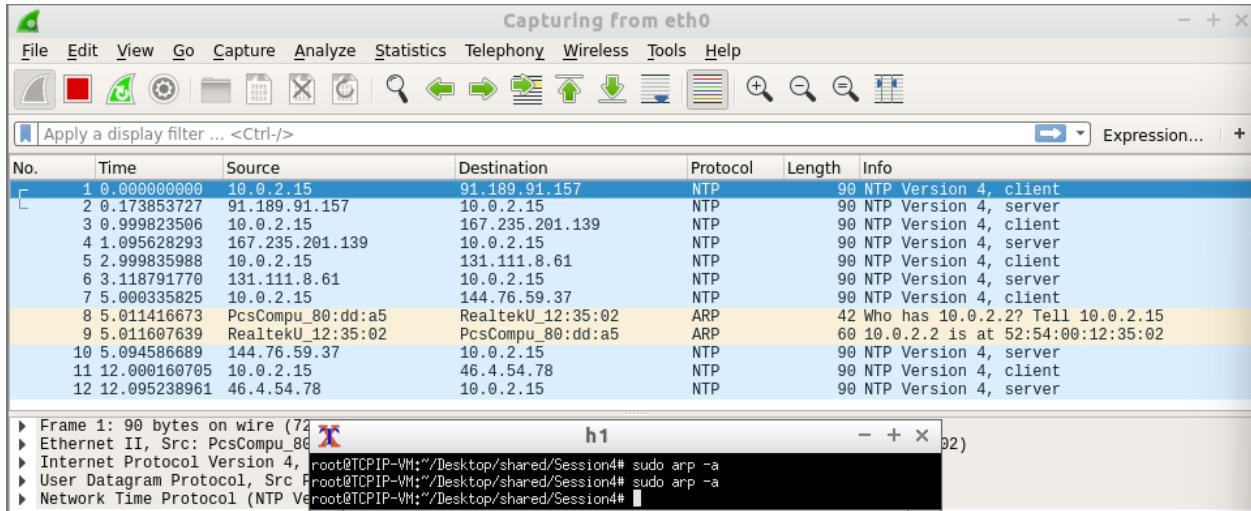
mininet> sh ovs-ofctl add-flow switch name?action=normal

```
mininet@TCPIP-VM:~/Desktop/shared/Session4$ sudo /usr/bin/ovs-ofctl add-flow s12 action=normal
mininet@TCPIP-VM:~/Desktop/shared/Session4$ _
```

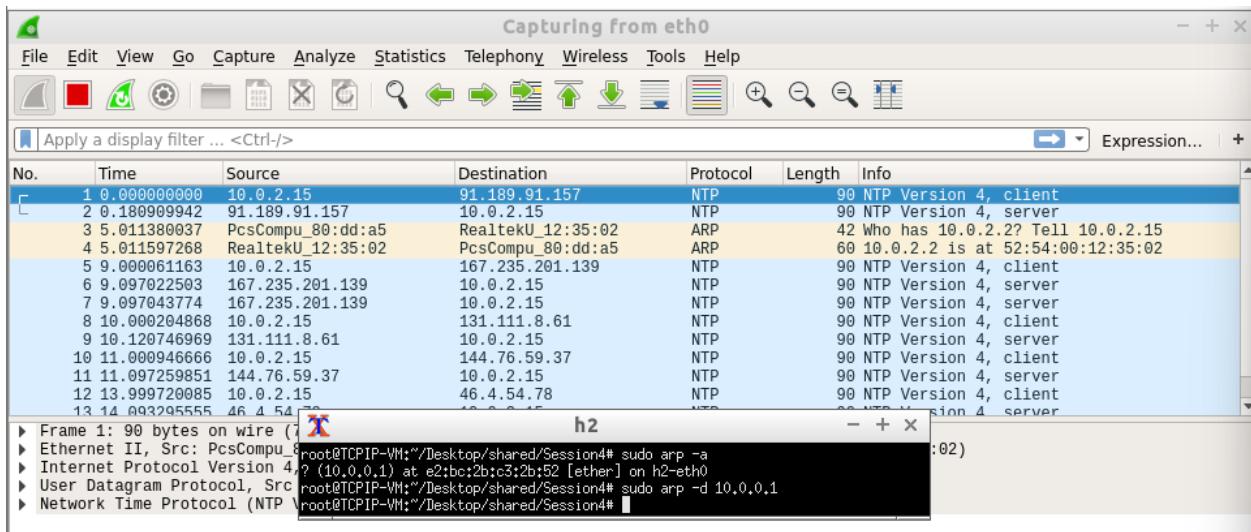
برای آزمایش پیکربندی سوئیچ بار دیگر برنامه Wireshark را جهت شنود روی هر چهار ماشین راه اندازی نموده، محتوای کش ARP در h1 را پاک کنید و مجدداً h2 را ( تنها یک مرتبه ) از h1 پینگ نمایید.



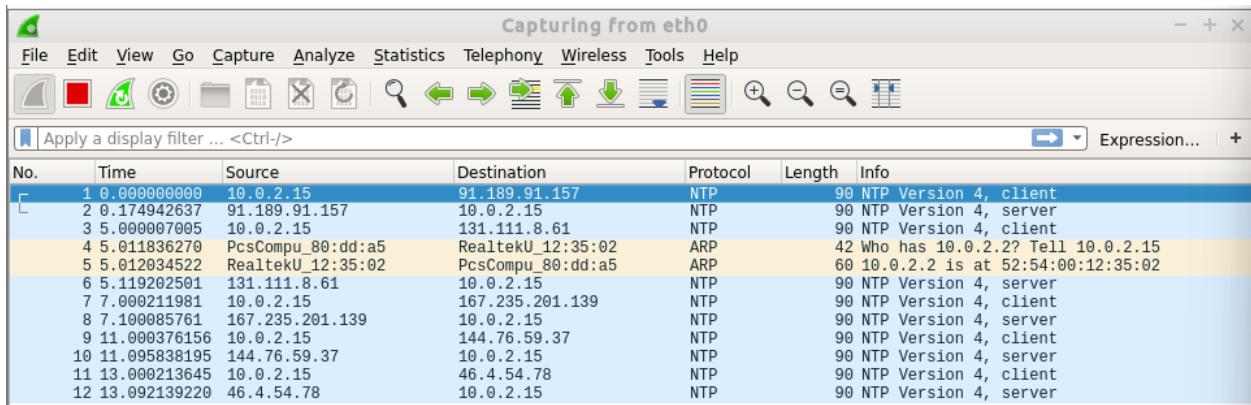
H1:

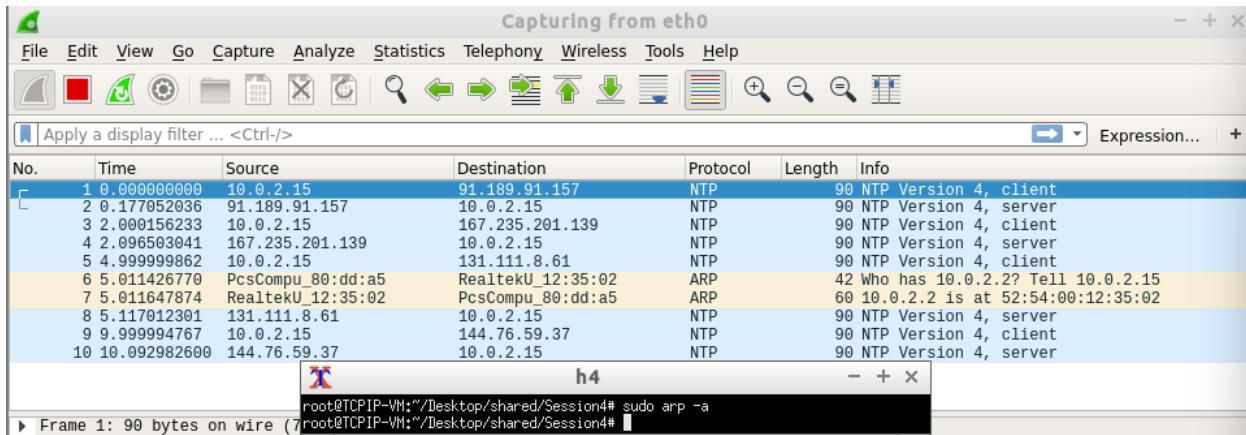


H2:



H3:





سؤال ۳- انواع مختلف بسته هایی را که در ماشین های مختلف ملاحظه می کنید، تشریح نمایید.

ابتدا ۱۲ کرا به سوییچ تبدیل می کنیم. در این سوال با ping کردن  $h_1$  و  $h_2$  توسط  $h_1$  و برای شناسایی هدف، در ابتدا پیام "۱۰۰۰۰۲" از نوع ARP، برای همه ارسال می شود. بعد از شناسایی هدف، چون  $h_2$  یک سوییچ بوده، پاسخ  $h_2$  را دیگر به سمت  $h_1$  نفرستاده و فقط برای  $h_1$  می فرستد. در ادامه  $h_2$  نیز برای آپدیت کردن ARP Table خود، یک پیام فرستاده و آدرس فیزیکی  $h_1$  را جویا می شود که در نهایت  $h_1$  به آن پاسخ داده و جدول  $h_2$  آپدیت می شود. در نتیجه از ۴ نوع بسته استفاده شده است؛ این بسته ها عبارتند از:

ARP Request .

ARP Reply .

ICMP Request .

ICMP Reply .

سؤال ۴- توضیح دهید که چه تفاوتی با حالتی که دو هاب داشتیم، به وجود آمده است.

در این حالت با توجه به استفاده از سوییچ، پیام های از نوع ICMP، پیام های از نوع ARP که برای پیدا کردن آدرس فیزیکی hostها است، به hostهای ۳ و ۴ ارسال می شود. باید توجه شود که سوییچ پاسخ host هدف را تنها به hostی که ARP Request را ارسال کرده باز می گرداند. در نتیجه در این حالت، بسته های زیر فقط در  $h_1$  و  $h_2$  دیده می شوند:

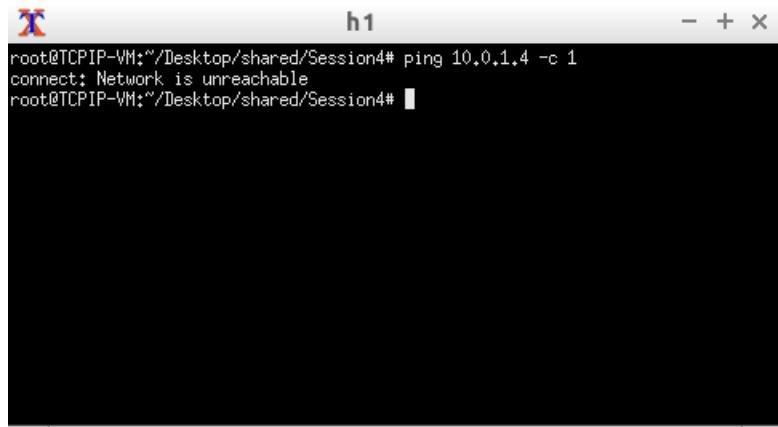
ICMP REQUEST .

ICMP REPLY .

ARP REPLY .

در حالیکه بسته ARP REQUEST در همه ماشین ها دیده می شود؛ زیرا این بسته توسط ماشین مبدأ broadcast می شود و همه آن را دریافت می کنند.

سؤال ۵- از h۱ ماشین را (تنها یک مرتبه) پینگ کنید و ترافیک مورد مشاهده و همینطور استنباط خود را تشریح نمایید.



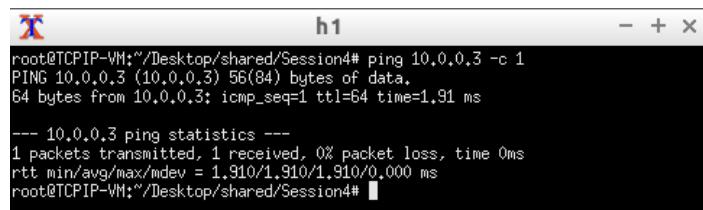
```
root@TCPPIP-VM:~/Desktop/shared/Session4# ping 10.0.1.4 -c 1
connect: Network is unreachable
root@TCPPIP-VM:~/Desktop/shared/Session4#
```

این دو host، در یک شبکه نیستند؛ در نتیجه این ping کردن ممکن نیست و ارور "Network is unreachable" نمایش داده می شود. هیچ بسته ای در هیچکدام از ماشین ها در wireshark مشاهده نمی شود.

سؤال ۶- بسته های دریافتی روی h۲ را ملاحظه کرده و مشاهدات خود را تشریح نمایید.

به جز H2 بقیه ماشین ها همه بسته های ارسالی توسط H1 را دریافت کرده اند؛ زیرا بین H1 و H2 سوییچ قرار دارد و بسته به پورت متصل به H3 از سوییچ می رود. در نتیجه به H2 بسته ای نمی فرستد. از طرف دیگر، بسته ای که از آن پورت سوییچ می رود، به یک هاب می رسد و هاب این بسته را به H۴ نیز می فرستد. البته بسته arp request توسط همه دریافت می شود.

حال از ماشین ۱ ماشین ۳ را تنها یک بار ping می کنیم:



```
root@TCPPIP-VM:~/Desktop/shared/Session4# ping 10.0.0.3 -c 1
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=1.91 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.910/1.910/1.910/0.000 ms
root@TCPPIP-VM:~/Desktop/shared/Session4#
```

H1:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	c6:4f:d0:75:6b:58	Broadcast	ARP	42	who has 10.0.0.3? Tell 1
2	0.000027416	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	10.0.0.3 is at f2:ba:dd:16:3a:bb
3	0.0000799608	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=0x00000000
4	0.0000800053	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=0x00000000
5	5.218530920	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	Who has 10.0.0.1? Tell 1
6	5.219882418	c6:4f:d0:75:6b:58	f2:ba:dd:16:3a:bb	ARP	42	10.0.0.1 is at c6:4f:d0:75:6b:58

H2:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	c6:4f:d0:75:6b:58	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1

H3:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	c6:4f:d0:75:6b:58	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1
2	0.000027416	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	10.0.0.3 is at f2:ba:dd:
3	0.000799608	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=
4	0.000808053	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.218530920	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	Who has 10.0.0.1? Tell 1
6	5.219882418	c6:4f:d0:75:6b:58	f2:ba:dd:16:3a:bb	ARP	42	10.0.0.1 is at c6:4f:d0:

H4:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	c6:4f:d0:75:6b:58	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1
2	0.000046795	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	10.0.0.3 is at f2:ba:dd:
3	0.000799191	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=
4	0.000826009	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.218601857	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	Who has 10.0.0.1? Tell 1
6	5.219882222	c6:4f:d0:75:6b:58	f2:ba:dd:16:3a:bb	ARP	42	10.0.0.1 is at c6:4f:d0:

سؤال ۷- بسته های ارسالی توسط  $h_1$  را با آنها یک دریافت می شوند، مقایسه نمایید (به ویژه به لحاظ آدرس های MAC مبدأ و مقصد آنها). وجود تشابه و تفاوت آنها (در صورت وجود) چیست؟ توضیح دهید.

H1:

○ آدرس MAC منبع: دستگاه  $h_1$

○ آدرس MAC مقصد: ابتدا آدرس MAC مقصد، آدرس MAC دستگاه  $h_3$  خواهد بود، زیرا  $h_1$  تلاش می کند به صورت مستقیم با  $h_3$  ارتباط برقرار کند. با این حال،  $S_{12}$  که یک سوییچ است، جدول آدرس MAC خود را مورد بررسی قرار داده و بسته را به پورت متصل به  $S_{34}$  (جایی که  $h_3$  قرار دارد)، ارسال خواهد کرد. به عبارت دیگر، آدرس MAC مقصد همچنان همان آدرس  $h_3$  دستگاه  $h_3$  خواهد بود.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	62:4c:49:ea:12:80	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1
2	0.003084050	ba:67:0b:4c:26:4e	62:4c:49:ea:12:80	ARP	42	10.0.0.3 is at ba:67:0b:
3	0.003092473	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=
4	0.006161503	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.159139570	ba:67:0b:4c:26:4e	62:4c:49:ea:12:80	ARP	42	Who has 10.0.0.1? Tell 1
6	5.159154101	62:4c:49:ea:12:80	ba:67:0b:4c:26:4e	ARP	42	10.0.0.1 is at 62:4c:49:

Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface h1-eth0, id 0  
 Ethernet II, Src: ba:67:0b:4c:26:4e (ba:67:0b:4c:26:4e), Dst: 62:4c:49:ea:12:80 (62:4c:49:ea:12:80)  
 Destination: 62:4c:49:ea:12:80 (62:4c:49:ea:12:80)  
 Source: ba:67:0b:4c:26:4e (ba:67:0b:4c:26:4e)  
 Type: IPv4 (0x0800)  
 Internet Protocol Version 4, Src: 10.0.0.3, Dst: 10.0.0.1  
 Internet Control Message Protocol

H3:

○ آدرس MAC منبع: دستگاه  $h_3$

○ آدرس MAC مقصد: اگر  $h_3$  یک بسته به سمت  $h_1$  ارسال کند، آدرس MAC مقصد، آدرس MAC دستگاه  $h_1$  خواهد بود.  $S_{34}$  بسته را به تمامی پورت های خود، از جمله پورت متصل به  $S_{12}$ ، ارسال می کند؛ زیرا به عنوان یک هاب عمل می کند، بنابراین  $S_{12}$  به عنوان یک سوییچ، جدول آدرس MAC خود را بررسی

کرده و بسته را به پورت متصل به  $h_1$  ارسال می کند. بنابراین، آدرس MAC مقصد همچنان همان آدرس MAC دستگاه  $h_1$  خواهد بود. در کل، حتی با اینکه ۵۳۴ به عنوان یک هاب عمل می کند، آدرسهای MAC منبع و مقصد در بسته های ارسالی توسط  $h_1$  و  $h_3$  همچنان همانند انتظار برای ارتباط عادی بین دستگاه ها باقی می ماند. تغییر در عملکرد ناشی از وجود هاب در چگونگی ارسال بسته ها در داخل شبکه است، نه در آدرسهای MAC درون بسته ها.

در نتیجه می بینیم که این اطلاعات در  $h_1$  و  $h_3$  تفاوتی ندارند، در واقع سوییج و هاب، اطلاعات مربوط به آدرس مک مبدا و مقصد را تغییر نمی دهند.

ج) آشنایی با نحوه کارکرد روتر (router)

تست همبندی:

```

LXTerminal
File Edit Tabs Help
mininet@TCP/IP-VM:~/Desktop/shared/Session4$ sudo python lab4-2.py
*** Adding controller
*** Adding hosts
*** Adding router
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2 h3 r1
*** Starting controller
*** Starting 1 switches
s12
*** Configuring hosts
*** Starting xterm on hosts
*** Running the command line interface
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X r1
h2 -> h1 X r1
h3 -> X X r1
r1 -> h1 h2 h3
*** Results: 33% dropped (8/12 received)
mininet>

```

سؤال ۸- چه درصدی از بسته ها drop می شوند؟

۳۳ درصد از بسته ها drop شده و ۸/۱۲ آنها به مقصد می رسند.

سؤال ۹- کدام host ها قادر به برقراری ارتباط با هم نیستند؟

هاست h۱ با h۳ و هاست h۲ با h۳ بالعکس نمی توانند ارتباط برقرار کنند؛ مگر اینکه router را به عنوان gateway برای دو شبکه معرفی کنیم و ip\_forward را روشن کنیم.

سؤال ۱۰- از چه mask subnet ای در سرتاسر فایل استفاده شده است؟

با توجه به یکسان بودن ۲۴ بیت اول آدرس های IP، در تمام اسکریپت از subnet ۰/۲۴ یا 255.255.255.0 استفاده شده.

سؤال ۱۱- اینترفیس های روتر r۱ و آدرس های IP نظیر آنها چیست؟

s IP address = 10.0.0.100/24 .1'r0-eth0

s IP address = 10.0.1.100/24 .2'r1-eth0

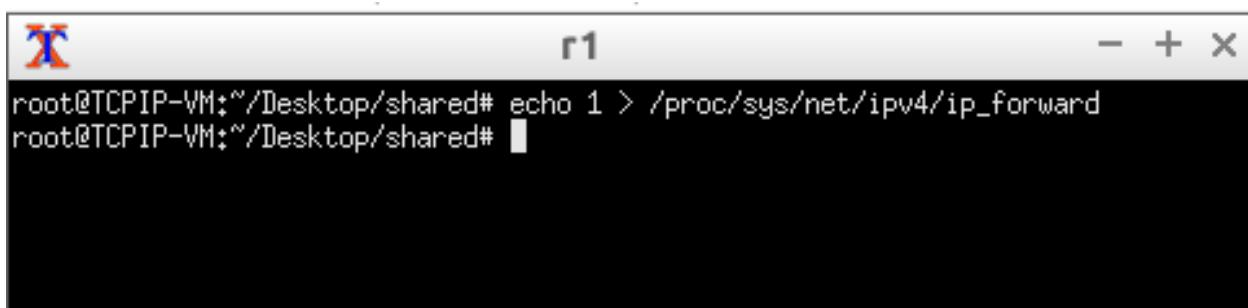
سؤال ۱۲- آیا می توانید اشتباهات پیکربندی را بیابید؟ دستورات لازم برای تصحیحات را بنویسید.

یکی از اشتباهات، تعریف اشتباه default gateway در h2 می باشد که برای درست کردن آن از دستور زیر استفاده می کنیم :



```
root@TCPIP-VM:~/Desktop/shared# ip route del default via 10.0.0.101
root@TCPIP-VM:~/Desktop/shared# ip route add default via 10.0.0.100
root@TCPIP-VM:~/Desktop/shared#
```

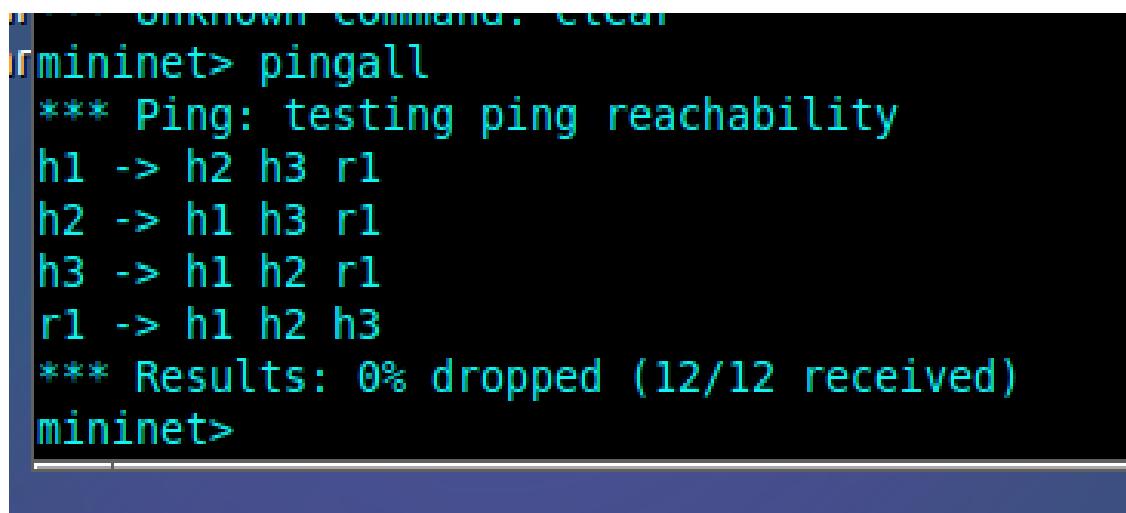
علاوه بر آن، با توجه به اینکه r۱ یک روتر است، باید ip\_forward در آن فعال باشد که بنابراین از دستور زیر در اینترفیس r۱ استفاده می کنیم:



```
root@TCPIP-VM:~/Desktop/shared# echo 1 > /proc/sys/net/ipv4/ip_forward
root@TCPIP-VM:~/Desktop/shared#
```

سؤال ۱۳- در صد بسته های drop شده چقدر است؟

پس از رفع مشکل شبکه، اگر یک بار دیگر دستور pingall را اجرا کنیم باید ببینیم که همه بسته ها با موفقیت ارسال شده اند و هیچ بسته ای drop نشده است در نتیجه با اجرای دستور pingall داریم به صفر خواهیم رسید.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 r1
h2 -> h1 h3 r1
h3 -> h1 h2 r1
r1 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

سؤال ۱۴- هنگام مسیریابی بسته ها از ایترفیس `eth0-r1` و `eth1-r1` چه تغییری در بسته های IP رخ می دهد؟ روتر آدرس MAC بسته های دریافتی اش را عوض کرده و سپس به مقصدشان ارسال می کند، به این صورت که بسته خروجی از روتر، دارای آدرس فیزیکی ایترفیسی از روتر که برروی آن قرار گرفته، و آدرس فیزیکی مقصد این بسته، ایترفیسی از مقصد است که مستقیماً به روتر متصل است. این عمل در هردو ایترفیس `21` اتفاق می افتد.

سؤال ۱۵- هدف از اعمال این تغییرات چیست؟

روترها زمانی آدرس MAC بسته ها را تغییر می دهند که آنها را بین `subnet` های مختلف ارسال می کنند تا جداسازی `broadcast` domain ها حفظ شود و تضمین شود که بسته به مقصد صحیح در `subnet` مقصد تحویل داده شود. این بخشی از عملکرد روترها در یک محیط شبکه `multi subnet` است.

سؤال ۱۶- دستورات لازم برای منظور فوق را بنویسید.

در `21`:

ip route change default via 10.0.1.3

در `h3`:

ip route del default via 10.0.1.100

```

root@TCPIP-VM:~/Desktop/shared# ip route change default via 10.0.1.3
root@TCPIP-VM:~/Desktop/shared# 

root@TCPIP-VM:~/Desktop/shared# ip route del default via 10.0.1.100
RTNETLINK answers: No such process
root@TCPIP-VM:~/Desktop/shared# echo 1 > /proc/sys/net/ipv4/ip_forward
root@TCPIP-VM:~/Desktop/shared# ip route del default via 10.0.1.100

```

Echo 1 > /proc/sys/net/ipv4/ip\_forward

سؤال 17- بر اساس شنود WireShark، توضیح دهید که پینگهای فوق چرا کار نمی کنند؟!  
زیرا هنگامی که پینگ میخواهد به  $h1$  یا  $h2$  باز گردد در  $r1$  آبی  $h3$  بر روی default GW تنظیم شده و پاسخ به  $h1$  باز نمیگردد.

سؤال 18- با فرض اینکه بعداً از  $h3$  خواهیم خواست که به عنوان gateway دسترسی به اینترنت عمل کند، یک تک دستور بنویسید که مشکل پینگ از  $h1$  و  $h2$  به سوی  $h3$  را حل کند. این تک دستور را روی کدام ماشین اجرا می نمایید؟  
بر روی  $r1$  این دستور را وارد میکنیم تا پیام های reply از gw صحیح به  $h1$  یا  $h2$  برسند

ip route add 10.0.0.0/24 via 10.0.1.100

```

root@TCPIP-VM:~/Desktop/shared# ping h3
ping: unknown host h3
root@TCPIP-VM:~/Desktop/shared# ping 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
^C
--- 10.0.1.3 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 4999ms

root@TCPIP-VM:~/Desktop/shared# 

root@TCPIP-VM:~/Desktop/shared# ip route del default via 10.0.0.101
root@TCPIP-VM:~/Desktop/shared# ip route add default via 10.0.0.100
root@TCPIP-VM:~/Desktop/shared# ping 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
^C
--- 10.0.1.3 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 3998ms

root@TCPIP-VM:~/Desktop/shared# 

```



## دانشکده مهندسی کامپیوتر

### آزمایشگاه شبکه های کامپیوتری

استاد : شکوفه نوروزی

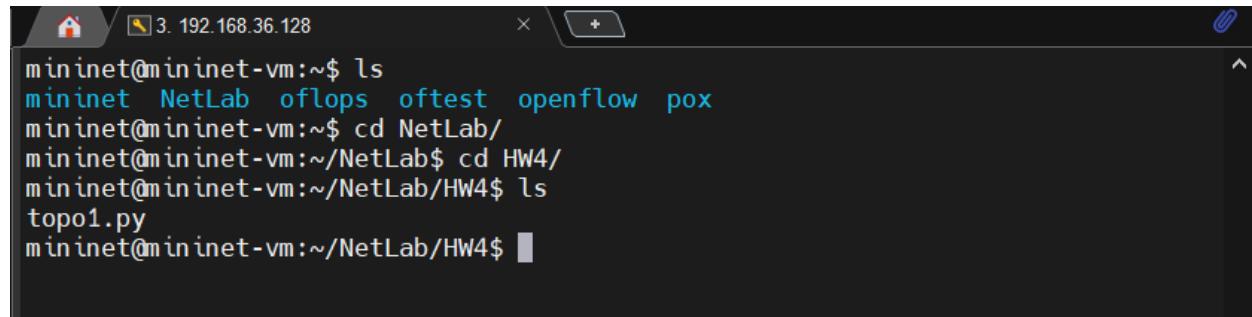
پوریا رحیمی (99521289)

بکتاش انصاری (99521082)

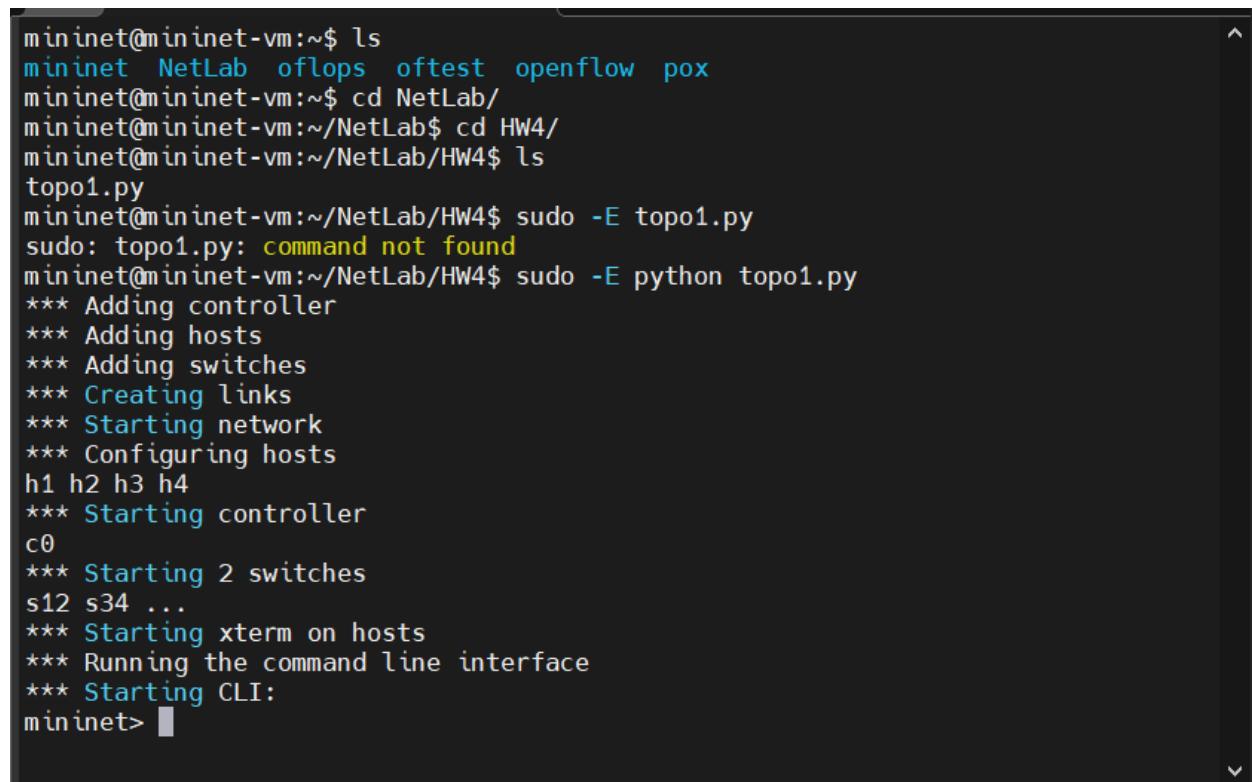
پاییز 1402

## الف) آشنایی با نحوه کارکرد هاب (hub)

یک ترمینال در VM خود باز کنید و اسکریپت topo1.py را اجرا نمایید.



```
mininet@mininet-vm:~$ ls
mininet NetLab oflops oftest openflow pox
mininet@mininet-vm:~$ cd NetLab/
mininet@mininet-vm:~/NetLab$ cd HW4/
mininet@mininet-vm:~/NetLab/HW4$ ls
topo1.py
mininet@mininet-vm:~/NetLab/HW4$
```



```
mininet@mininet-vm:~$ ls
mininet NetLab oflops oftest openflow pox
mininet@mininet-vm:~$ cd NetLab/
mininet@mininet-vm:~/NetLab$ cd HW4/
mininet@mininet-vm:~/NetLab/HW4$ ls
topo1.py
mininet@mininet-vm:~/NetLab/HW4$ sudo -E topo1.py
sudo: topo1.py: command not found
mininet@mininet-vm:~/NetLab/HW4$ sudo -E python topo1.py
*** Adding controller
*** Adding hosts
*** Adding switches
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s12 s34 ...
*** Starting xterm on hosts
*** Running the command line interface
*** Starting CLI:
mininet>
```

تبديل سوییچ های s12 و s34 به hub

```
*** Starting CLI: mininet> sh ovs-ofctl add-flow s12 action=flood mininet> sh ovs-ofctl add-flow s34 action=flood mininet> █
```

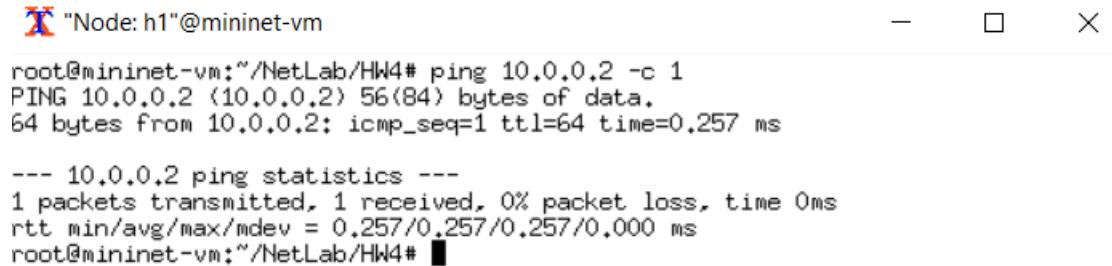
سوال 1 - بله، مقدار netmask برای h2 درست نمی باشد و باید درست شود. برای همین از دستور زیر برای درست کردن این مشکل استفاده می کنیم :

در دستور زیر مقدار subnet mask را برای h2 درست می کنیم :

```
X "Node: h2"@mininet-vm
root@mininet-vm:~/NetLab/HW4# ifconfig h2-eth0 10.0.0.2 netmask 255.255.255.0
root@mininet-vm:~/NetLab/HW4# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.2 netmask 255.255.255.0 broadcast 10.255.255.255
        ether f2:77:3c:03:20:c4 txqueuelen 1000 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 928 bytes 121444 (121.4 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 928 bytes 121444 (121.4 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

حال از ماشین h1 تنها یک بار ماشین h2 را ping می کنیم و به صورت زیر می باشد :



```

root@mininet-vm:~/NetLab/HW4# ping 10.0.0.2 -c 1
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.257 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.257/0.257/0.257/0.000 ms
root@mininet-vm:~/NetLab/HW4# █

```

ماشین h2 را توسط ماشین h1 تنها یک بار ping کردیم و نتیجه به صورت زیر می باشد :

: H1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1
2	0.000190126	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	10.0.0.2 is at 1e:23:ff:c0:c9:8c
3	0.000192281	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=1
4	0.000245214	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=1
5	5.006411880	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	Who has 10.0.0.1? Tell 1
6	5.006438708	7a:c1:db:69:44:1c	1e:23:ff:c0:c9:8c	ARP	42	10.0.0.1 is at 7a:c1:db:69:44:1c

: H2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1
2	0.000190126	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	10.0.0.2 is at 1e:23:ff:c0:c9:8c
3	0.000192281	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=1
4	0.000245214	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=1
5	5.006411880	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	Who has 10.0.0.1? Tell 1
6	5.006438708	7a:c1:db:69:44:1c	1e:23:ff:c0:c9:8c	ARP	42	10.0.0.1 is at 7a:c1:db:69:44:1c

: H3

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1
2	0.000012637	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	10.0.0.2 is at 1e:23:ff:
3	0.000061245	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=
4	0.000069225	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.006260127	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	Who has 10.0.0.1? Tell 1
6	5.006268713	7a:c1:db:69:44:1c	1e:23:ff:c0:c9:8c	ARP	42	10.0.0.1 is at 7a:c1:db:

: H4

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1
2	0.000011792	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	10.0.0.2 is at 1e:23:ff:
3	0.000060530	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=
4	0.000068399	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.006260238	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	Who has 10.0.0.1? Tell 1
6	5.006267904	7a:c1:db:69:44:1c	1e:23:ff:c0:c9:8c	ARP	42	10.0.0.1 is at 7a:c1:db:

همانطور که مشاهده می شود بسته های ارسال شده که شامل ARP Request , ICMP Request , ARP Reply , ICMP Reply توسط H1 و جواب های داده شده توسط H2 به همه ماشین ها ارسال می شود و در واقع ارتباط همه ماشین ها با هم برقرار می باشد همانطور که در بالا نیز مشاهده می شود.

سوال 2 - در پاسخ سوال باید گفت که خیر تفاوتی میان ترافیک های capture شده نیست چرا که در این شبکه و برای اتصالات از hub استفاده شده است و hub هر بسته ای را که به دستش برسد به سایر پورتهایش ارسال می کند، بنابراین با ارسال ping از h1 به h2، پیام شده توسط تمامی ماشین ها یکسان می باشد.

ب) بررسی نحوه کار Switch

: switch به hub تغییر

```
mininet> sh ovs-ofctl add-flow s12 action=normal
mininet> █
```

برای آزمایش پیکربندی سوئیچ بار دیگر برنامه WireShark را جهت شنود روی هدر چهار ماشین راه اندازی نموده، محتوای کش ARP در h1 را پاک کنید و مجدداً h2 را تنها یک مرتبه از h1 پینگ نمایید.



```
X "Node: h1"@mininet-vm
-
root@mininet-vm:~/NetLab/HW4# arp -a
? (10.0.0.2) at 1e:23:ff:c0:c9:8c [ether] on h1-eth0
root@mininet-vm:~/NetLab/HW4# arp -d 10.0.0.2
root@mininet-vm:~/NetLab/HW4# arp -a
root@mininet-vm:~/NetLab/HW4#
root@mininet-vm:~/NetLab/HW4# █
```

: H1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1
2	0.000160481	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	10.0.0.2 is at 1e:23:ff:c0:c9:8c
3	0.000162131	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=10.0.0.1
4	0.000208235	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=10.0.0.2
5	5.039891993	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	Who has 10.0.0.1? Tell 1
6	5.039899007	7a:c1:db:69:44:1c	1e:23:ff:c0:c9:8c	ARP	42	10.0.0.1 is at 7a:c1:db:69:44:1c

: H2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1
2	0.000007047	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	10.0.0.2 is at 1e:23:ff:
3	0.000075547	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=
4	0.000083686	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.036091280	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	Who has 10.0.0.1? Tell 1
6	5.039838158	7a:c1:db:69:44:1c	1e:23:ff:c0:c9:8c	ARP	42	10.0.0.1 is at 7a:c1:db:

: H3

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1

: H4

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1

- 3 سوال

ابتدا  $s12$  را به سوییچ تبدیل می کنیم. در این سوال با  $h2$  ping کردن  $h1$ ، در ابتدا پیام 10.0.0.2 کیست؟ برای شناسایی هدف و از نوع ARP، برای همه ارسال می شود. بعد از شناسایی هدف، چون  $s12$

یک سویچ بوده، پاسخ h2 را دیگر به سمت hub نفرستاده و فقط برای h1 می فرستد که در ادامه h2 نیز برای آپدیت کردن ARP Table خود، یک پیام فرستاده و آدرس فیزیکی h1 را جویا می شود که در نهایت h1 به آن پاسخ داده و جدول h2 اپدیت می شود. در نتیجه با توجه به توضیحاتی که داده شد در این بخش از 4 نوع بسته استفاده شده است. بسته ها عبارتند از :

ARP Request , ARP Reply , ICMP Request , ICMP Reply

#### سوال 4 -

در این حالت با توجه به اینکه از سوییچ استفاده کرده ایم بنابراین پیام های از نوع ICMP ، به همه هاست ها ارسال نشده و تنها پیام های درخواست ARP که برای پیدا کردن آدرس فیزیکی هاست ها است، به هاست های ۳ و ۴ ارسال می شود. باید توجه شود که سوییچ پاسخ هاست هدف را تنها به هاستی که Request ARP را ارسال کرده باز میگردد و نه سایر هاست ها!

در نتیجه در این حالت، بسته های REQUEST ICMP ,REPLY ICMP فقط در h1 و h2 دیده می شوند.

اما بسته REQUEST ARP در همه ماشین ها دیده می شود و این به این خاطر است که این بسته توسط ماشین مبدأ broadcast می شود و همه آن را دریافت میکنند.

: H1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1
2	0.000160481	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	10.0.0.2 is at 1e:23:ff:
3	0.000162131	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=
4	0.000208235	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.039891993	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	Who has 10.0.0.17? Tell 1
6	5.039899007	7a:c1:db:69:44:1c	1e:23:ff:c0:c9:8c	ARP	42	10.0.0.1 is at 7a:c1:db:

: H2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1
2	0.000007047	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	10.0.0.2 is at 1e:23:ff:
3	0.000075547	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=
4	0.000083686	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.036091280	1e:23:ff:c0:c9:8c	7a:c1:db:69:44:1c	ARP	42	Who has 10.0.0.17? Tell 1
6	5.039838158	7a:c1:db:69:44:1c	1e:23:ff:c0:c9:8c	ARP	42	10.0.0.1 is at 7a:c1:db:

: H3

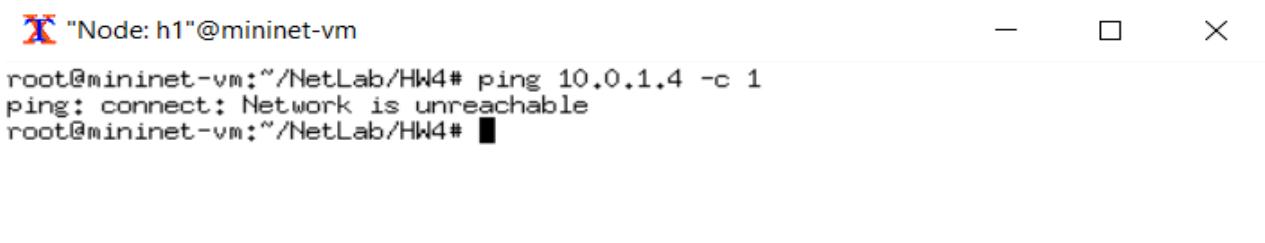
Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1

: H4

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	7a:c1:db:69:44:1c	Broadcast	ARP	42	Who has 10.0.0.2? Tell 1

## سوال 5 -

این عمل امکان پذیر نبوده چرا که این دو هاست در یک شبکه نیستند در نتیجه به ارور **network is unreachable** بر خواهیم خورد و هیچ بسته ای در هیچ کدام از ماشین ها درون **wireshark** مشاهده نخواهد شد.



```
"Node: h1"@\ninet-vm\nroot@\ninet-vm:~/NetLab/HW4# ping 10.0.1.4 -c 1\nping: connect: Network is unreachable\nroot@\ninet-vm:~/NetLab/HW4#
```

## سوال 6 -

به جز  $H_2$  بقیه ماشین ها همه بسته های ارسالی توسط  $H_1$  را دریافت کرده اند و این به این خاطر است که بین  $H_1$  و  $H_2$  سوییچ قرار دارد و سوییچ بسته را به پورتی از خودش می برد که  $H_3$  آن جا قرار دارد و در نتیجه به  $H_2$  بسته ای نمی فرستد. از طرف دیگر، بسته ای که از آن پورت سوییچ می رود، به یک هاب می رسد و هاب این بسته را به  $H_4$  نیز می فرستد. البته بسته **request arp** را همه دریافت می کنند.

حال از ماشین 1 ماشین 3 را تنها یک بار **ping** می کنیم :

Node: h1" @mininet-vm

```
root@mininet-vm:~/NetLab/HW4# ping 10.0.0.3 -c 1
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=1.51 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.513/1.513/1.513/0.000 ms
root@mininet-vm:~/NetLab/HW4# █
```

در نتیجه طبق گفته ها و مشاهدات داریم :

: H1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	c6:4f:d0:75:6b:58	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1
2	0.000952609	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	10.0.0.3 is at f2:ba:dd:
3	0.000955228	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=
4	0.001501034	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.219950930	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	Who has 10.0.0.17? Tell 1
6	5.219956822	c6:4f:d0:75:6b:58	f2:ba:dd:16:3a:bb	ARP	42	10.0.0.1 is at c6:4f:d0:

: H2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	c6:4f:d0:75:6b:58	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1

: H3

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	c6:4f:d0:75:6b:58	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1
2	0.000027416	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	10.0.0.3 is at f2:ba:dd:
3	0.000799608	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=
4	0.000808053	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.218530920	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	Who has 10.0.0.1? Tell 1
6	5.219882418	c6:4f:d0:75:6b:58	f2:ba:dd:16:3a:bb	ARP	42	10.0.0.1 is at c6:4f:d0:

: H4

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	c6:4f:d0:75:6b:58	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1
2	0.000046795	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	10.0.0.3 is at f2:ba:dd:
3	0.000799191	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=
4	0.000826009	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.218601857	f2:ba:dd:16:3a:bb	c6:4f:d0:75:6b:58	ARP	42	Who has 10.0.0.1? Tell 1
6	5.219882222	c6:4f:d0:75:6b:58	f2:ba:dd:16:3a:bb	ARP	42	10.0.0.1 is at c6:4f:d0:

- 7 سوال

: h1 توسط های ارسالی بسته

آدرس MAC منبع: آدرس MAC منبع، آدرس MAC دستگاه h1 خواهد بود.

آدرس MAC مقصد: ابتدا آدرس MAC مقصد، آدرس MAC دستگاه h3 خواهد بود، زیرا h1 تلاش می کند به صورت مستقیم با h3 ارتباط برقرار کند. با این حال، s12 یک سوییچ است، بنابراین جدول آدرس MAC خود را مورد بررسی قرار داده و بسته را به پورت متصل به s34 جایی که h3 قرار دارد، ارسال خواهد کرد. به عبارت دیگر، آدرس MAC مقصد همچنان همان آدرس MAC h3 خواهد بود.

: h3 توسط های ارسالی بسته

آدرس MAC منبع: آدرس MAC منبع، آدرس MAC دستگاه h3 خواهد بود.

آدرس MAC مقصد: اگر h3 یک بسته به سمت h1 ارسال کند، آدرس MAC مقصد آن زمان آدرس MAC دستگاه h1 خواهد بود. S34 به عنوان یک هاب عمل می کند، بنابراین بسته را به تمامی پورت های خود ارسال می کند، از جمله پورت متصل به s12 . s12 ، به عنوان یک سوییچ، جدول آدرس MAC خود را بررسی کرده و بسته را به پورت متصل به h1 ارسال می کند. بنابراین، آدرس MAC مقصد همچنان همان آدرس MAC h1 خواهد بود. در کل، حتی با اینکه s34 به عنوان یک هاب عمل می کند، آدرس های MAC منبع و مقصد در بسته های ارسالی توسط h1 و h3 همچنان همانند انتظار برای ارتباط عادی بین دستگاه ها باقی می ماند. تغییر در عملکرد ناشی از وجود هاب در این است که چگونگی ارسال بسته ها در داخل شبکه است، نه در آدرس های MAC درون بسته ها.

: H1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	62:4c:49:ea:12:80	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1
2	0.003084050	ba:67:0b:4c:26:4e	62:4c:49:ea:12:80	ARP	42	10.0.0.3 is at ba:67:0b:
3	0.003092473	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=
4	0.006161503	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.159139570	ba:67:0b:4c:26:4e	62:4c:49:ea:12:80	ARP	42	Who has 10.0.0.1? Tell 1
6	5.159154101	62:4c:49:ea:12:80	ba:67:0b:4c:26:4e	ARP	42	10.0.0.1 is at 62:4c:49:

```

Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface h1-eth0, id 0
Ethernet II, Src: ba:67:0b:4c:26:4e (ba:67:0b:4c:26:4e), Dst: 62:4c:49:ea:12:80 (62:4c:49:ea:12:80)
  Destination: 62:4c:49:ea:12:80 (62:4c:49:ea:12:80)
  Source: ba:67:0b:4c:26:4e (ba:67:0b:4c:26:4e)
  Type: IPv4 (0x0800)
  Internet Protocol Version 4, Src: 10.0.0.3, Dst: 10.0.0.1
  Internet Control Message Protocol

```

: H3

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	62:4c:49:ea:12:80	Broadcast	ARP	42	Who has 10.0.0.3? Tell 1
2	0.000015680	ba:67:0b:4c:26:4e	62:4c:49:ea:12:80	ARP	42	10.0.0.3 is at ba:67:0b:
3	0.001145882	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=
4	0.001174978	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=
5	5.155273930	ba:67:0b:4c:26:4e	62:4c:49:ea:12:80	ARP	42	Who has 10.0.0.1? Tell 1
6	5.161198131	62:4c:49:ea:12:80	ba:67:0b:4c:26:4e	ARP	42	10.0.0.1 is at 62:4c:49:

```
▶ Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface h3-eth0, id 0
  ▶ Ethernet II, Src: 62:4c:49:ea:12:80 (62:4c:49:ea:12:80), Dst: ba:67:0b:4c:26:4e (ba:67:0b:4c:26:4e)
    ▶ Destination: ba:67:0b:4c:26:4e (ba:67:0b:4c:26:4e)
    ▶ Source: 62:4c:49:ea:12:80 (62:4c:49:ea:12:80)
    ▶ Type: IPv4 (0x0800)
  ▶ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.3
  ▶ Internet Control Message Protocol
```

در نتیجه می بینیم که این اطلاعات در h1 و h3 تفاوتی ندارند، در واقع سوییچ و هاب، اطلاعات مربوط به آدرس مک مبدا و مقصد را تغییر نمی دهند.

ج) آشنایی با نحوه کارکرد مسیریاب (روتر)

سوال 8 -

پس از دستور pingall می بینیم که r1-eth0 ,h1 ,h2 در یک شبکه و h3 , r1-eth1 در یک شبکه دیگر حضور دارند. روتر به ماشین های هر دو شبکه دسترسی دارد ولی خود ماشین ها فقط به هاست های درون شبکه خود دسترسی دارند.

```

mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X r1
h2 -> h1 X r1
h3 -> X X r1
r1 -> h1 h2 h3
*** Results: 33% dropped (8/12 received)
mininet>

```

mininet-vm 4% 0.33 GB / 6.76 GB 0.01 Mb/s 0.00 Mb/s 8 mil

با توجه به درصدی که بدست آوردیم 33 درصد از بسته های ما drop می شوند و فقط 8 بسته از 12 بسته به مقصد می رسند.

### سوال 9 -

هاست h1 با h3 و هاست h2 با h3 و بالعکس نمی توانند با هم ارتباط برقرار کنند مگر اینکه روتر را به عنوان gateway برای دو شبکه معرفی کنیم و ip\_forward را روشن کنیم.

### سوال 10 -

با توجه به یکسان بودن ۲۴ بیت اول آدرس های IP ، subnet mask برابر با 255.255.255.0 می باشد.

### سوال 11 -

r0-eth0's IP address = 10.0.0.100/24 .1

r1-eth0's IP address = 10.0.1.100/24 .2

```

X "Node: r1"@mininet-vm
root@mininet-vm:~/NetLab/HW4# ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 624 bytes 99584 (99.5 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 624 bytes 99584 (99.5 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

r1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.100 netmask 255.255.255.0 broadcast 10.0.0.255
        ether 22:de:f2:75:cc:ce txqueuelen 1000 (Ethernet)
        RX packets 13 bytes 826 (826.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 8 bytes 560 (560.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

r1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.1.100 netmask 255.255.255.0 broadcast 10.0.1.255
        ether 5e:ac:3a:ec:93:66 txqueuelen 1000 (Ethernet)
        RX packets 6 bytes 476 (476.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 4 bytes 280 (280.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

## سوال - 12

یکی از اشتباهات، تعریف اشتباه default gateway در h2 می باشد که برای درست کردن آن از دستور زیر استفاده می کنیم :

```

X "Node: h2"@mininet-vm
root@mininet-vm:~/NetLab/HW4# ip route del default via 10.0.0.101
root@mininet-vm:~/NetLab/HW4# ip route add default via 10.0.0.100
root@mininet-vm:~/NetLab/HW4# █

```

علاوه بر آن، با توجه به اینکه r1 یک روتراست، باید ip\_forward در آن فعال باشد که بنابراین از دستور زیر در اینترفیس r1 استفاده می کنیم:

 "Node: r1" @mininet-vm  
root@mininet-vm:~/NetLab/HW4# echo 1 > /proc/sys/net/ipv4/ip\_forward  
root@mininet-vm:~/NetLab/HW4# █

### سوال 13 -

پس از رفع مشکل شبکه ، اگر یک بار دیگر دستور pingall را اجرا کنیم باید ببینیم که همه بسته ها با موفقیت ارسال شده اند و هیچ بسته ای نشده است در نتیجه با اجرای دستور pingall داریم :

```
*** Ping: testing ping reachability
h1 -> h2 h3 r1
h2 -> h1 h3 r1
h3 -> h1 h2 r1
r1 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> █
```

و همانطور که انتظار داشتیم 0 درصد drop داریم.

## سوال 14 -

روتر آدرس MAC بسته های دریافتی اش را عوض کرده و سپس به مقصدشان ارسال می کند، به این صورت که بسته خروجی از روتر، دارای آدرس فیزیکی اینترفیسی از روتر که برروی آن قرار گرفته، و آدرس فیزیکی مقصد این بسته، اینترفیسی از مقصد است که مستقیماً به روتر متصل است. این عمل در هردو اینترفیس ۲۱ اتفاق می افتد.

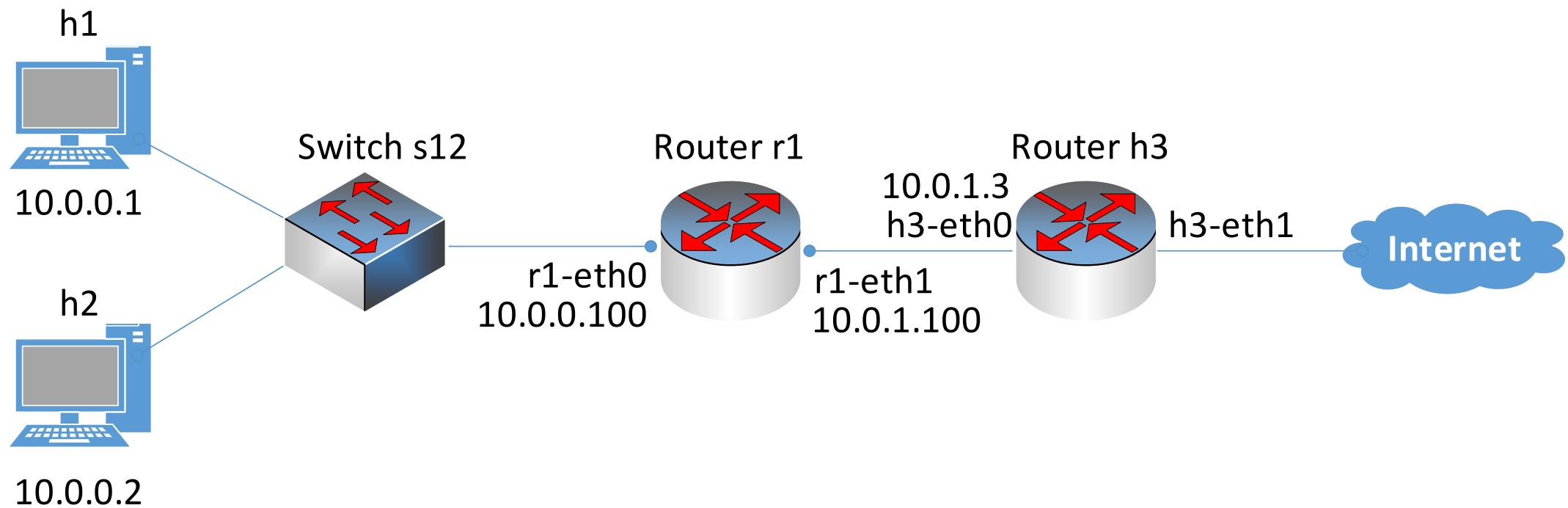
## سوال 15 -

روترها زمانی آدرس MAC بسته ها را تغییر می دهند که آنها را بین های مختلف ارسال می کنند تا جداسازی broadcast subnet ها حفظ شود و تضمین شود که بسته به مقصد صحیح در subnet مقصد تحویل داده شود. این بخشی از عملکرد روترها در یک محیط شبکه multi subnet است.

# Connecting Virtual Environment to the Real World Using NAT

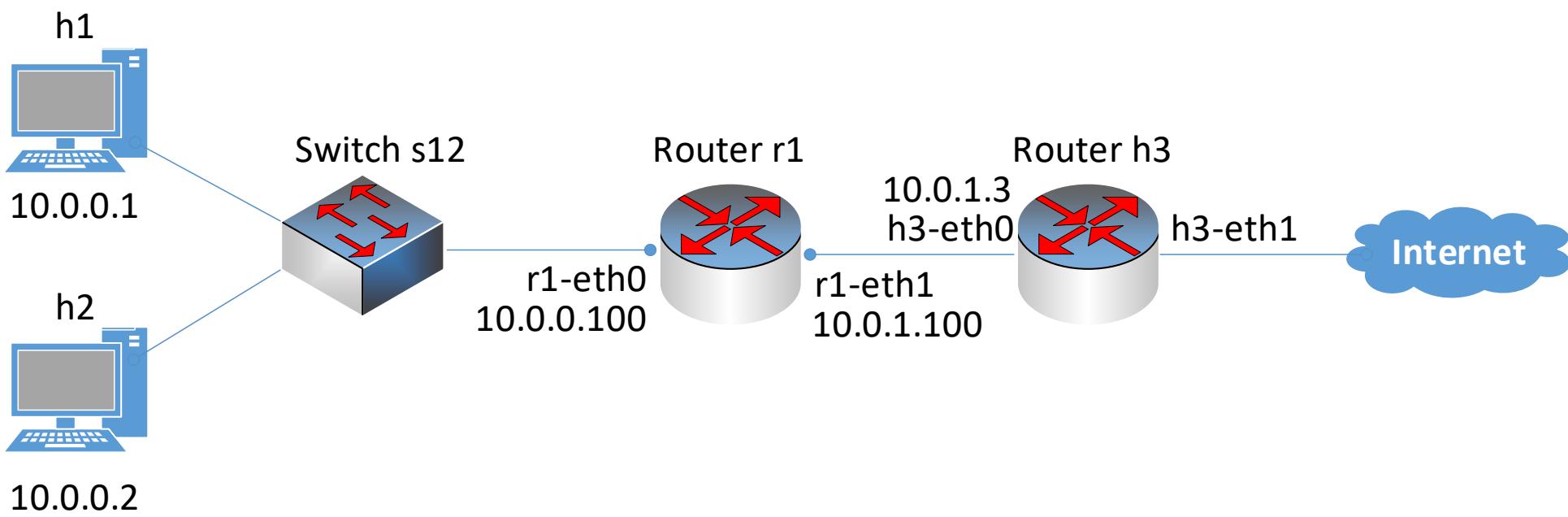
# Connect a virtual network to the Internet

- Network configuration with a connection to the real world

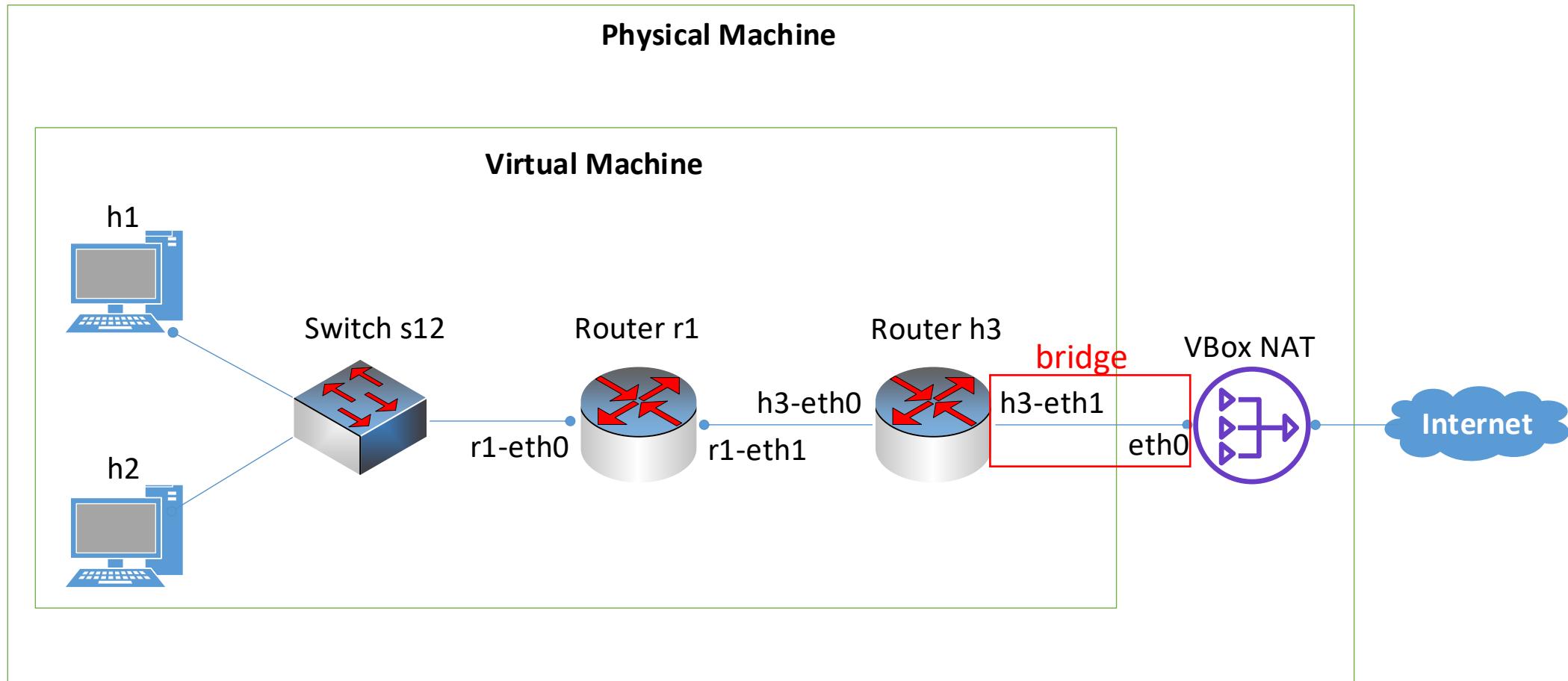


# Connect a virtual network to the Internet

- Main steps:
  1. We require a real IP address on h3-eth1 interface of h3.
  2. We need to masquerade the traffic coming from h1 and h2.

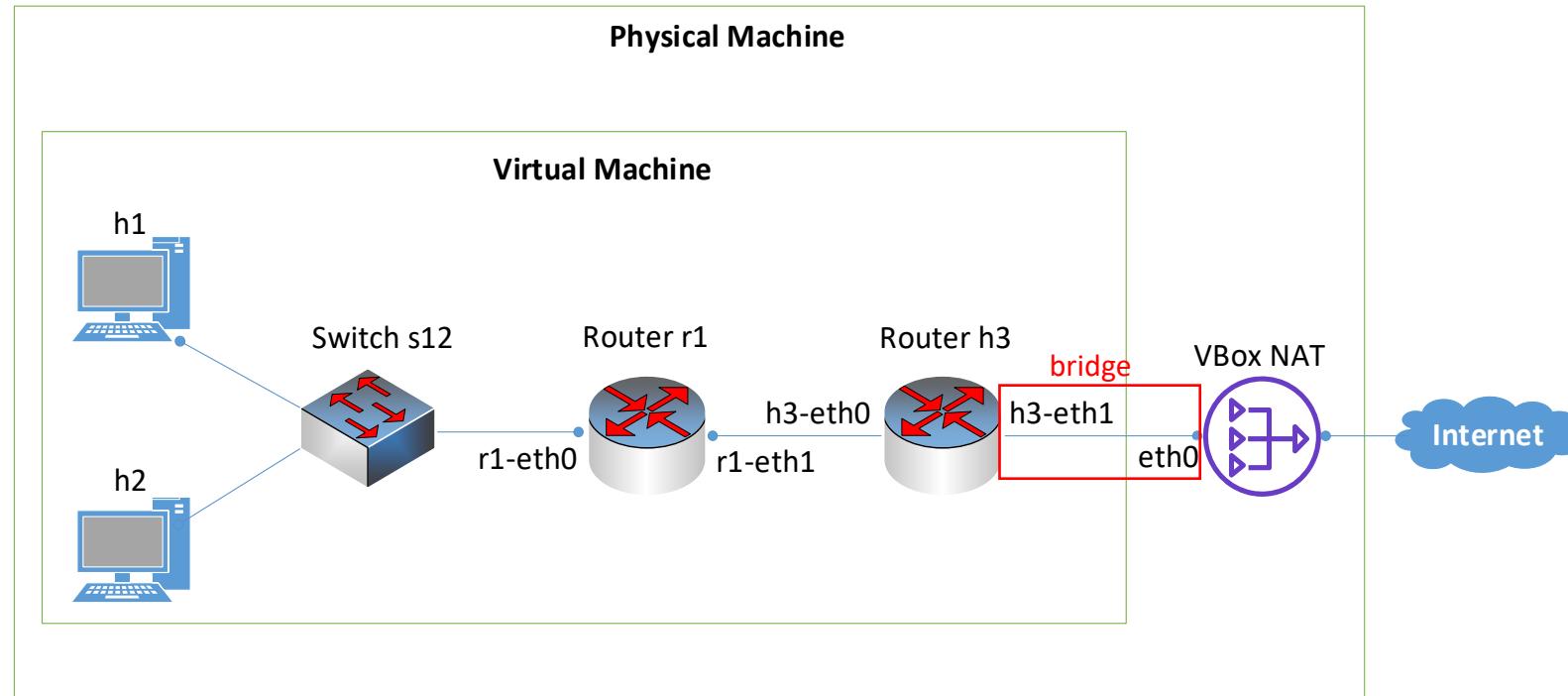


# Bridging the network adapter

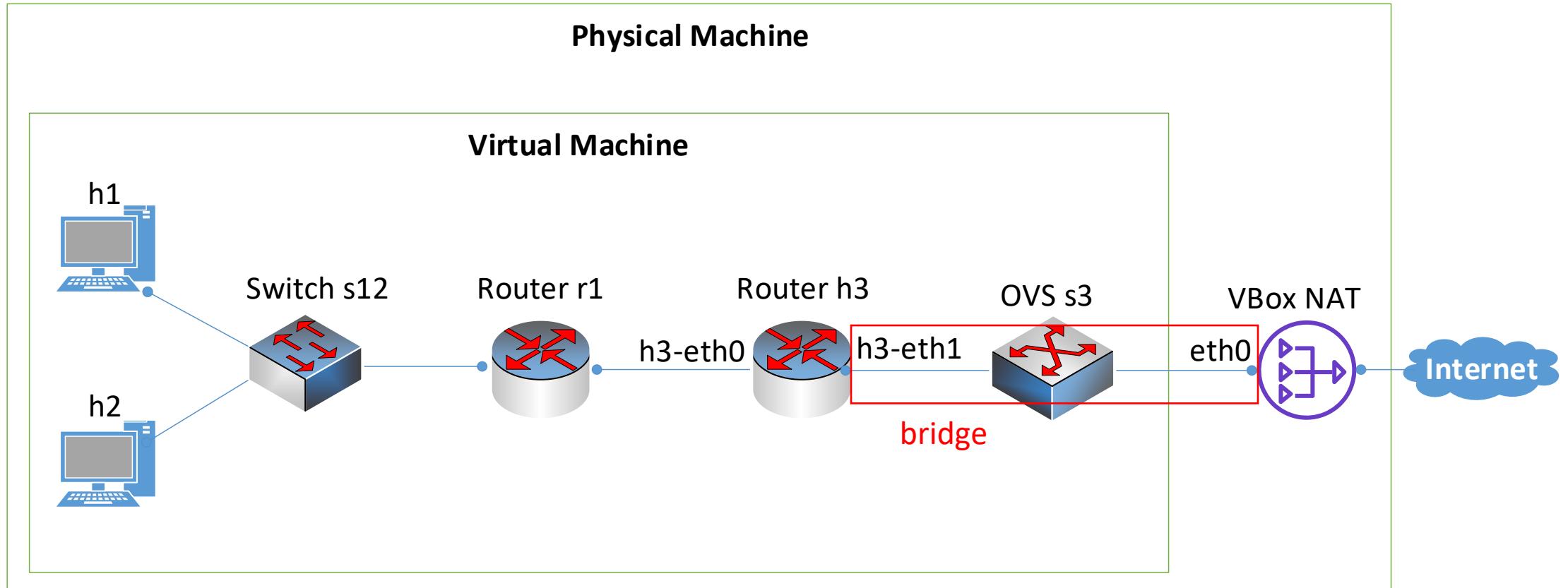


# Borrowing an IP Address

1. Create a bridge between a real interface in your host machine, and h3's eth1.
2. Finding and setting up a suitable IP address for h3-eth1.

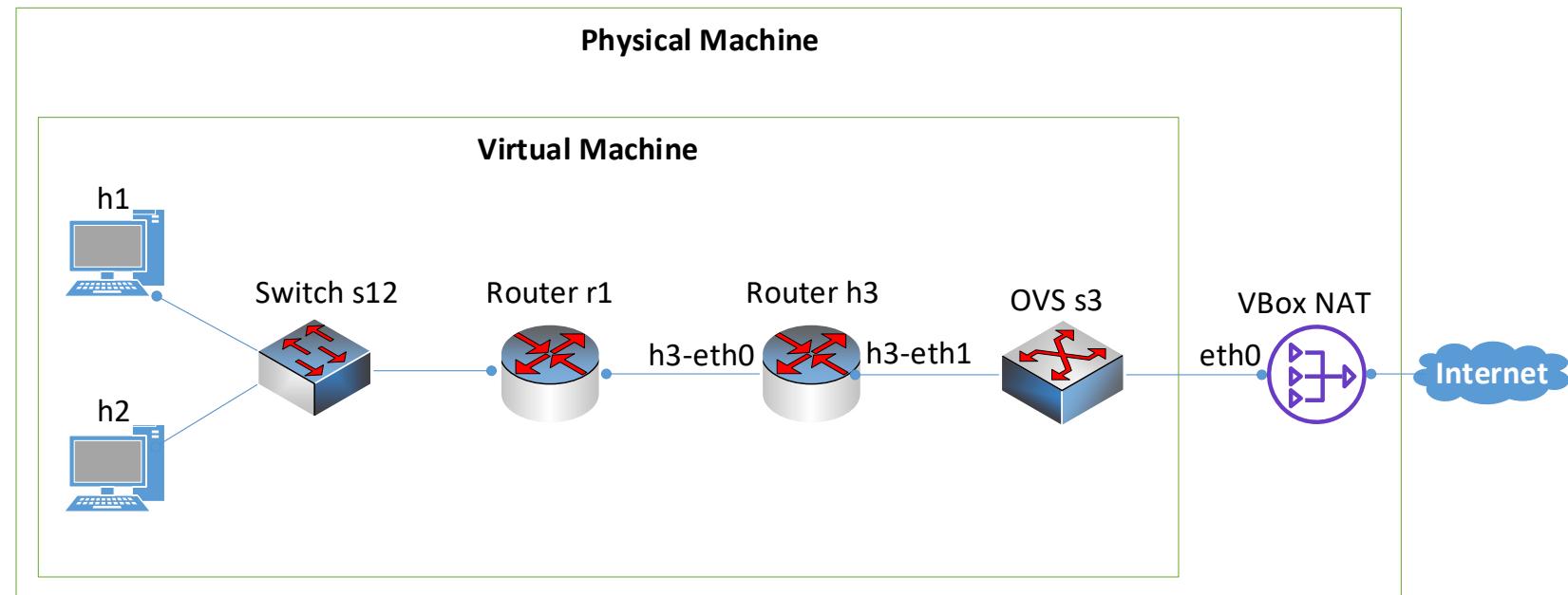


# Using Open vSwitch (OVS)



# Bridging the network adapter

- `$ sudo mn -c`
- `$ sudo ip addr flush dev eth0`
- `$ sudo python lab4.py`
- `mininet> sh ovs-vsctl add-port s3 eth0`



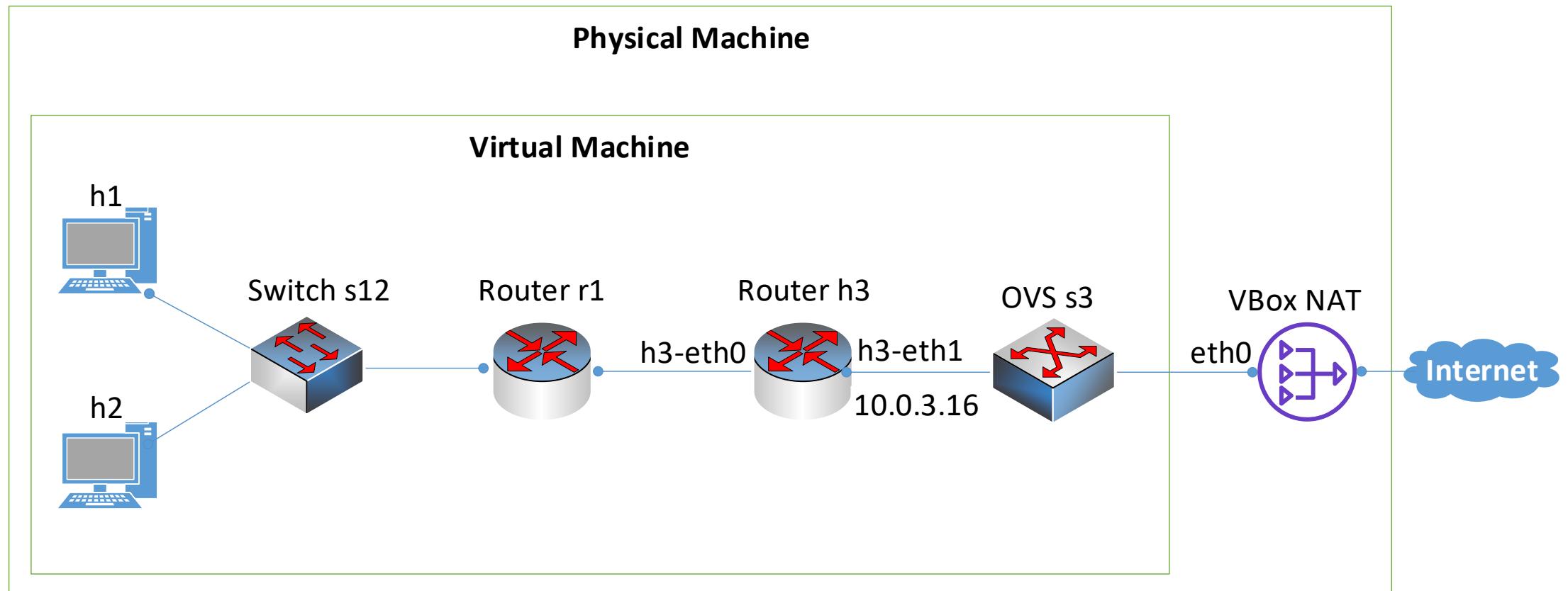
# Show the interface

- mininet> sh ovs-vsctl show

```
mininet> sh ovs-vsctl add-port s3 eth0
mininet> sh ovs-vsctl show
e7a21c84-4464-4b53-9d84-7ac031b48c46
  Bridge s3
    Controller "tcp:127.0.0.1:6653"
      is_connected: true
      fail_mode: secure
    Port s3-eth1
      Interface s3-eth1
    Port eth0
      Interface eth0
    Port s3
      Interface s3
      type: internal
  Bridge s12
    Controller "tcp:127.0.0.1:6653"
      is_connected: true
      fail_mode: secure
    Port s12-eth1
      Interface s12-eth1
    Port s12-eth2
      Interface s12-eth2
    Port s12
      Interface s12
      type: internal
    Port s12-eth3
      Interface s12-eth3
  ovs version: "2.13.1"
```

# Set an IP address to h3-eth1

- (h3)# dhclient h3-eth1



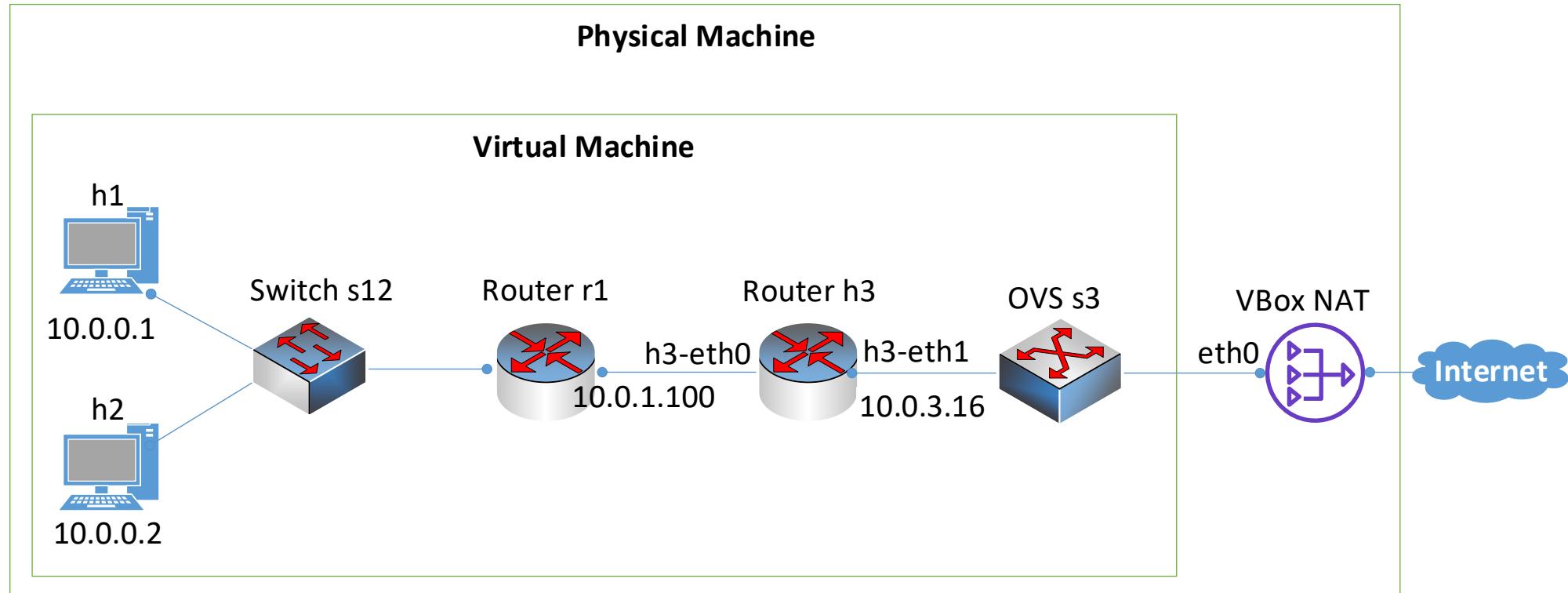
# Check connectivity

- # ping google.com

```
root@mininet-vm:/home/mininet/Downloads# ping google.com
ping: google.com: Temporary failure in name resolution
```

- This error occurs when the system cannot translate a website name into an IP address. The system cannot communicate with the DNS server and returns the error.
- # ping 8.8.8.8

# Masquerade



- (h3)# iptables -t nat -A POSTROUTING -o h3-eth1 -j MASQUERADE

# Specify the address of DNS server

- Ping google with its DNS IP address:
  - # ping 8.8.8.8
- Configure the DNS server:
  - # sudo su
  - # echo nameserver 8.8.8.8 > /etc/resolv.conf
- Ping google with its domain name:
  - # ping google.com

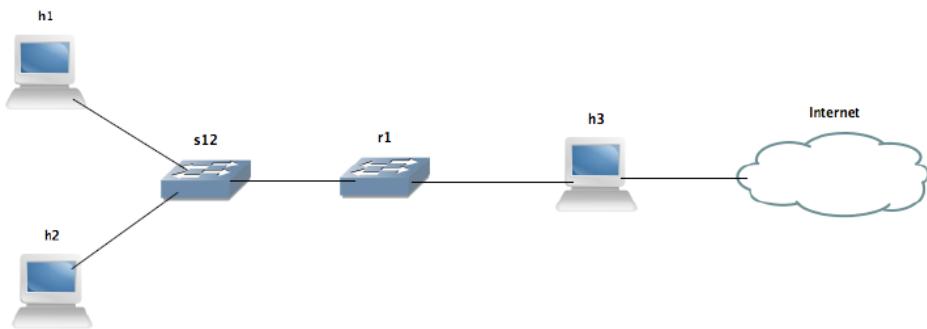
## آزمایشگاه شبکه

### آزمایش ۵: اتصال محیط مجازی به دنیای واقعی با استفاده از NAT

هدف این آزمایش، برقراری ارتباط شبکه مجازی ایزوله‌ای است که در آزمایش ۳ راهاندازی کردیم با شبکه اینترنت واقعی.

#### الف) مقدمه

با شبکه‌ای که در شکل ۱ نشان داده شده است، کار می‌کنیم. ماشین‌های h1 و h2 نقش دو workstation را بازی می‌کنند، r1 روتر ما خواهد بود و h3 نیز روتر مرزی است که قرار است اتصال ما را به دنیای خارج برقرار نماید.



شکل ۱ - پیکربندی شبکه دارای ارتباط با دنیای واقعی

سوال ۱- فرض کنید تنها یک ارتباط واقعی با دنیای بیرون داریم (یعنی: تنها یک آدرس IP معتبر در اختیار داریم). می‌خواهیم برای دو کلاینت h1 و h2 دسترسی اینترنت فراهم کنیم. از چه راهکاری برای حل این مسأله باید بهره‌برداری کرد؟ توضیح دهید که این راهکار چگونه مشکل را برطرف خواهد نمود.

دو گام اساسی برای اتصال محیط مجازی به محیط واقعی اینترنت وجود دارد:

- ۱) نیازمند یک آدرس IP واقعی روی اینترفیس h3-eth1 از ماشین h3 هستیم.
- ۲) باید ترافیک وارد h1 و h2 را جعل هویت یا اصطلاحاً masquerade نماییم.

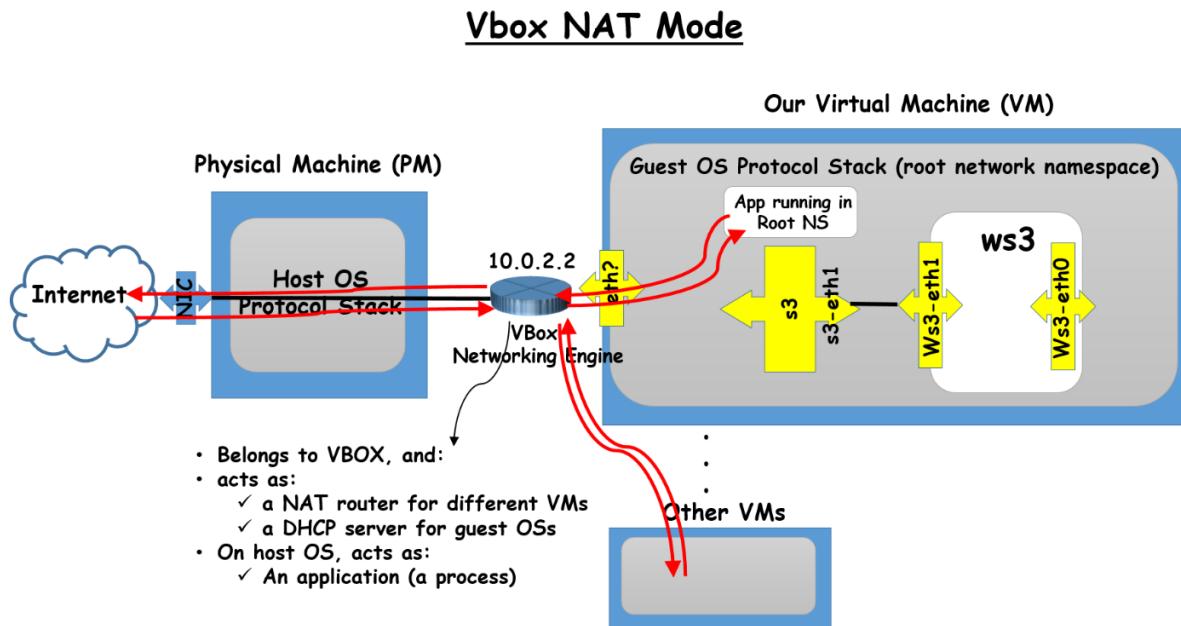
#### ب) قرض گرفتن اتصال مبتنی بر آدرس IP واقعی

همانطور که قبلاً گفتیم، برای اتصال به اینترنت، نیازمند یک آدرس IP معتبر هستیم. در این بخش، فرض می‌کنیم کامپیوتری که در حال حاضر روی آن کار می‌کنید (یعنی: ماشین میزبان VM)، دارای اتصال به اینترنت می‌باشد. برای اینکه بتوانیم اتصال IP کامپیوتر خود را برای h3 قرض بگیریم، دو گام زیر باید طی شوند:

- ۱) ایجاد یک پل (bridge) مابین ارتباطی که با ماشین میزبان داریم و اینترفیس h3-eth1 از h3.
- ۲) یافتن و تنظیم یک آدرس IP مناسب برای h3-eth1

## ب-۱) ایجاد پل بین اینترفیس مجازی eth1 و اتصال به بیرون

شکل ۲ به صورت شماتیک، وضعیت ارتباط مابین دو دنیای واقعی و مجازی را قبل از فرآیند پل‌سازی (bridging) نشان می‌دهد.



شکل ۲- اتصال بین دنیای مجازی و اینترنت

- به منظور پل‌سازی، ابتدا باید تعیین کنیم که VM ما از کدام اینترفیس برای دسترسی به اینترنت استفاده می‌کند.
- برنامه WireShark را روی VM اجرا کنید. روی تک تک اینترفیس‌ها capture کنید و با پینگ کردن google.com، تشخیص دهید که کدام اینترفیس برای دسترسی به اینترنت استفاده می‌شود. از این پس، این اینترفیس را با ? eth نشان می‌دهیم.

## سوال ۲- آدرس IP مورد استفاده برای دسترسی به اینترنت چیست؟

- حال که می‌دانیم کدام اینترفیس توسط VM استفاده می‌شود، گام بعدی، ایجاد یک پل بین ? eth و h3-eth1 است.

- در Mininet، ما با استفاده از برنامه OVS (OpenVSwitch) می‌توانیم پل تولید کنیم.
- اسکریپت lab4.py را اجرا نمایید. این اسکریپت همان تopoلوجی شکل ۱ را ایجاد می‌کند (همراه با بیشتر پیکربندی‌های لازم و ضمناً یک OVS اضافی به نام s3 هم برای انجام عملیات پل تولید می‌نماید).
- از خط دستور Mininet، با استفاده از دستور ovs-vsctl و add-port، اینترفیس ? eth را به پل s3 اضافه کنید. این کار باعث می‌شود تا ? eth به جای اتصال سابقش به پشتۀ IP stack (IP stack)، به پل متصل شود.

- با اجرای دستور `ovs-vsctl show` در فضای نام ریشه، می‌توان چک کرد که آیا `eth`? به عنوان یک پورت به `s3` اضافه شده است یا خیر.
- دیگر از جنس اینترفیس یک `host` نیست؛ بلکه عملاً پورتی از سوییچ `s3` است و ما به طور مستقیم از طریق `eth`? به اینترنت متصل نمی‌شویم. ضمناً، ما به پورت یک سوییچ IP نمی‌دهیم. بنابراین، با استفاده از دستور `ip addr flush dev eth`? آدرس IP این اینترفیس (پورت) را حذف کنید.
- به پنجه‌ر ترمینال `h3` رفته و دستور مقابل را اجرا نمایید: `dhclient h3-eth1`. این دستور یک آدرس IP قابل استفاده برای اینترفیس `h3-eth1` از `h3` ایجاد می‌کند و اجازه می‌دهد که از طریق پلی که ایجاد کردیم، به اینترنت دسترسی داشته باشد.
- پیکربندی ایجاد شده را با پینگ کردن گوگل از داخل `h3` تست نمایید.

## ب-۲) پیکربندی NAT

- در این بخش، قصد داریم ارتباط `h1` و `h2` با اینترنت را فراهم نماییم.
- ابتدا بررسی کنید که وقتی از `h1` (بر اساس همان آدرس‌های IP اولیه خودش) به سوی گوگل پینگ می‌کنید، چه اتفاقی می‌افتد؟ برای این منظور، با استفاده از `WireShark` روی اینترفیس `h1` و همچنین، روی اینترفیس `h3-eth1` از گوش دهید و از `h1`، نام دامنه گوگل را (یک مرتبه) پینگ نمایید.

سوال ۳- با تحلیل بسته‌های صادره از سوی `h1` توضیح دهید که چرا نتوانستید با سرور گوگل ارتباط برقرار نمایید.

    - حال، از آدرس IP گوگل را (یک مرتبه) پینگ نمایید.

سوال ۴- در `h3-eth1` چه مشاهده می‌شود؟ علت عدم موفقیت چیست؟

سوال ۵- دستور `iptables -t nat` مناسب برای ایجاد یک پیکربندی NAT مناسب در `h3` را بیان کنید.

- پس از اجرای دستور مورد نظر در سوال ۵، مجدداً از `h1`، ارتباط اینترنتی خود را با پینگ کردن Google (بر اساس نام دامنه) آزمایش کنید. با مشاهده پیام‌های ICMP و DNS، مطمئن شوید که به طور موفقیت‌آمیزی توانسته‌اید روتر `h3` را به عنوان یک NAT router پیکربندی نمایید.

- سوال ۶- هنگام پینگ کردن از طرف `h1` پیکربندی NAT در `h3` بر چه اساسی پاسخ‌های ICMP دریافتی را برای `h1` می‌فرستد؟ (راهنمایی: فکر می‌کنید چه فیلدی در بسته‌های درخواست/پاسخ ICMP برای شناسایی سورس حقیقی بسته‌ها مورد استفاده قرار می‌گیرد؟)

## گزارشکار آزمایش ۵: اتصال محیط مجازی به دنیای واقعی با استفاده از N

سیده شکیبا انارکی فیروز —

بهاره کاووسی نژاد — ۹۹۴۴۲۰۴۷

سوال ۱- فرض کنید تنها یک ارتباط واقعی با دنیای بیرون داریم (یعنی: تنها یک آدرس IP معتبر در اختیار داریم). میخواهیم برای دو کلاینت h2 و h1 دسترسی اینترنت فراهم کنیم. از چه راهکاری برای حل این مسأله باید بهره برداری کرد؟ توضیح دهید که این راهکار چگونه مشکل را برطرف خواهد نمود.

مراحل برطرف کردن مشکل:

۱. به دلیل داشتن یک آدرس IP، بهترین روش این است که router h3 را به شبکه اینترنت وصل کنیم. برای این منظور می توانیم یک آدرس IP واقعی برای interface h3-eth1 مد نظر قرار دهیم تا در شبکه اینترنت این interface مشخص باشد. (بسته های ارسالی از طریق router h3 host ها به router h3 می رسند)

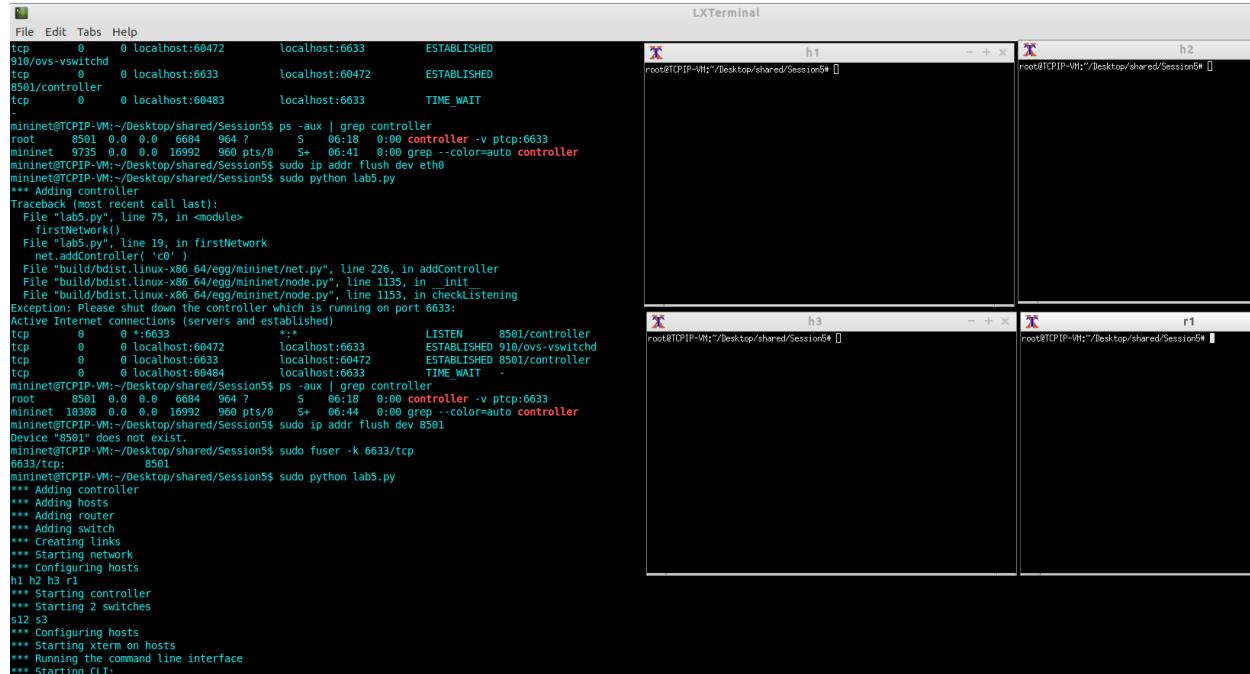
۲. در مرحله دوم پیدا کردن یک روش برای مسیریابی packet ها و ترافیک ارسالی از این دو host است که می توان از IP iptable در Masquerading استفاده کرد. (بسته های واقعی ایجاد می شوند)

۳. گام بعد استفاده از bridging است. باید یک پل بین interface h3-eth1 و interface h3-eth1 واقعی ایجاد کنیم. (بسته های واقعی توسط پل به خارج از شبکه local ارسال می شوند و فرایند برعکس نیز اعمال می شود)

۴. مرحله آخر پیدا کردن یک IP Address خوب و set کردن آن روی این interface است.

پیش از اجرای کد به **error** زیر بخوردیم که برای در اجرا بودن یک **controller** بود. با استفاده از دستور زیر این **controller** را از حالت اجرا خارج کردیم و پس از آن توانستیم با موفقیت فایل پایتون را اجرا کنیم.

`sudo fuser -k 6633/tcp`



```

File Edit Tabs Help
tcp 0 0 localhost:60472 localhost:6633 ESTABLISHED
910/ovs-vsctld
tcp 0 0 localhost:6633 localhost:60472 ESTABLISHED
8501/controller
tcp 0 0 localhost:60483 localhost:6633 TIME_WAIT
.
.
.
mininet@TCPPIP-VM:~/Desktop/shared/Session5$ ps -aux | grep controller
root 8501 0.0 0.8 6684 964 ? S 06:18 0:00 controller -v ptcp:6633
mininet 9735 0.0 0.8 16992 960 pts/0 S+ 06:41 0:00 grep --color=auto controller
mininet@TCPPIP-VM:~/Desktop/shared/Session5$ sudo ip addr flush dev eth0
mininet@TCPPIP-VM:~/Desktop/shared/Session5$ sudo python lab5.py
*** Adding controller
Traceback (most recent call last):
  File "lab5.py", line 75, in <module>
    firstNetwork()
  File "lab5.py", line 19, in firstNetwork
    net.addController('c0')
  File "/build/dhclient/linux-x86_64/egg/mininet/net.py", line 226, in addController
    File "/build/dhclient/linux-x86_64/egg/mininet/node.py", line 1135, in __init__
    File "/build/dhclient/linux-x86_64/egg/mininet/node.py", line 1153, in checkListening
Exception: Please shut down the controller which is running on port 6633:
Active Internet connections (servers and established)
tcp 0 0 6633 * LISTEN 8501/controller
tcp 0 0 localhost:60472 localhost:6633 ESTABLISHED 910/ovs-vsctld
tcp 0 0 localhost:6633 localhost:60472 ESTABLISHED 8501/controller
tcp 0 0 localhost:60484 localhost:6633 TIME_WAIT -
.
.
.
mininet@TCPPIP-VM:~/Desktop/shared/Session5$ ps -aux | grep controller
root 8501 0.0 0.8 6684 964 ? S 06:18 0:00 controller -v ptcp:6633
mininet 10388 0.0 0.8 16992 960 pts/0 S+ 06:44 0:00 grep --color=auto controller
mininet@TCPPIP-VM:~/Desktop/shared/Session5$ sudo ip addr flush dev 8501
Device '8501' does not exist.
mininet@TCPPIP-VM:~/Desktop/shared/Session5$ sudo fuser -k 6633/tcp
6633/tcp: 8501
mininet@TCPPIP-VM:~/Desktop/shared/Session5$ sudo python lab5.py
*** Adding controller
*** Adding hosts
*** Adding router
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2 h3 r1
*** Starting xterm on hosts
*** Running the command line interface
*** Starting CLI:
mininet> _

```

```

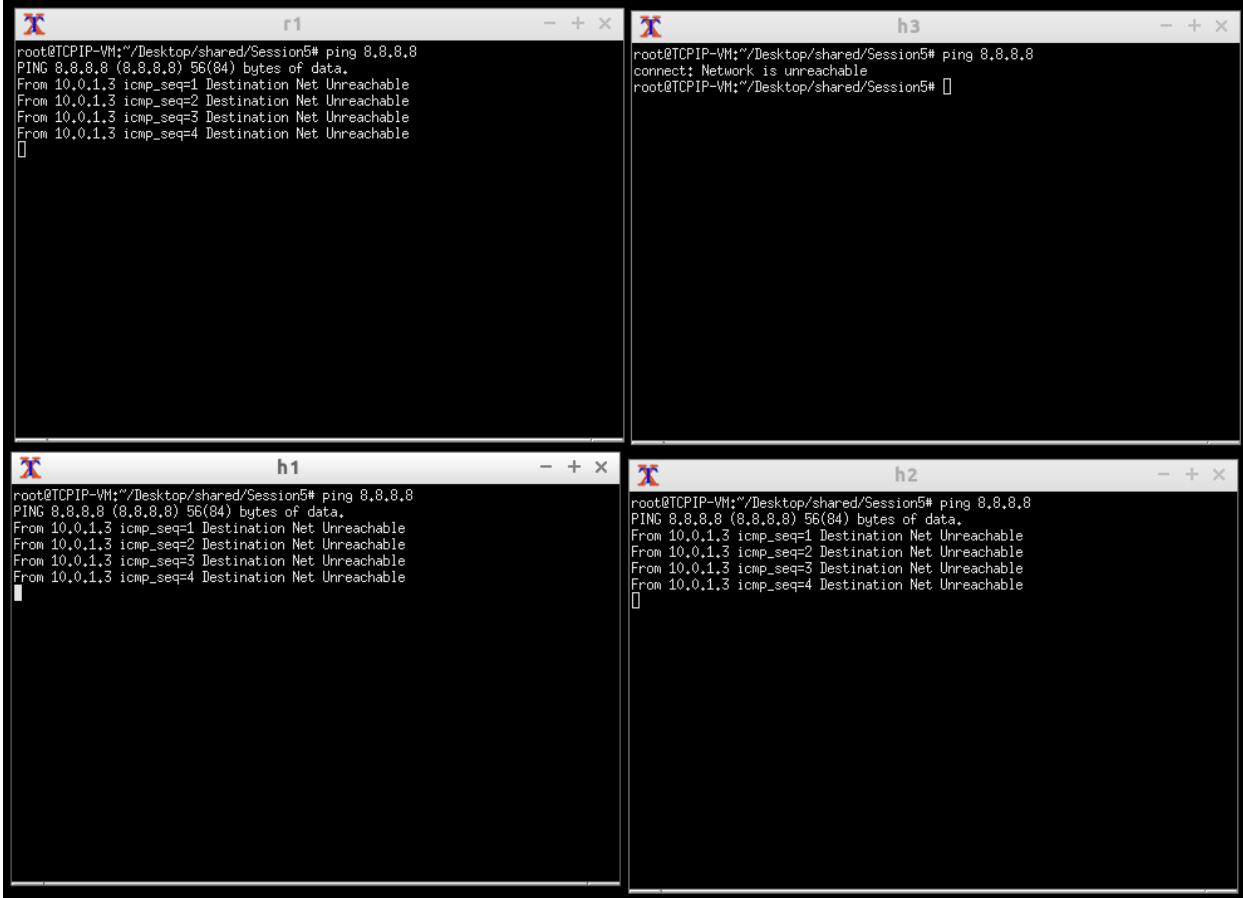
mininet> sh ovs-vsctl show
38f09442-9c39-4e0d-8c48-747b22b3fcfa
    Bridge "s12"
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
            fail_mode: secure
        Port "s12-eth3"
            Interface "s12-eth3"
        Port "s12"
            Interface "s12"
            type: internal
        Port "s12-eth2"
            Interface "s12-eth2"
        Port "s12-eth1"
            Interface "s12-eth1"
    Bridge "s3"
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
            fail_mode: secure
        Port "s3"
            Interface "s3"
            type: internal
        Port "s3-eth1"
            Interface "s3-eth1"
    ovs_version: "2.0.2"
mininet>

```

در قسمت network تنظیمات ماشین مجازی، 8.8.8.8 اتصال به اینترنت را بررسی می کنیم؛ سپس با دستور زیر interface IP را حذف می کنیم.

```
mininet@TCPIP-VM:~/Desktop/shared/Session5$ sudo ip addr flush dev eth0
mininet@TCPIP-VM:~/Desktop/shared/Session5$
```

پس از اجرای تپولوژی متوجه می شویم که اگر در h3 گوگل را ping کنیم پیام Network is unreachable نمایش داده می شود.



اکنون با دستور ovs, interface eth0 را به پل s3 اضافه می کنیم.

```
root@TCPIP-VM:~/Desktop/shared/Session5# ping 8.8.8.8
connect: Network is unreachable
root@TCPIP-VM:~/Desktop/shared/Session5# ovs-vsctl
lNo command 'ovs-vsctl' found, did you mean:
  Command 'ovs-vsctl' from package 'openvswitch-switch' (main)
ovs-vsctl: command not found
root@TCPIP-VM:~/Desktop/shared/Session5# sudo ovs-vsctl add-port s3 eth0
```

اکنون با استفاده از دستور dhclient یک IP Address قابل استفاده برای h3 interface eth1 برای h3 interface IP می دهیم و سپس دوباره گوگل را پینگ می کنیم و می بینیم که ارتباط برقرار است:

```

mininet> h3 dhclient h3-eth1
mininet> h3 ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=296 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=341 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=494 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=52 time=329 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=52 time=458 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=52 time=430 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=52 time=535 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=52 time=611 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=52 time=680 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=52 time=611 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=52 time=472 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=52 time=784 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=52 time=97.7 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=52 time=67.2 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=52 time=59.2 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=52 time=76.1 ms
64 bytes from 8.8.8.8: icmp_seq=17 ttl=52 time=66.4 ms
64 bytes from 8.8.8.8: icmp_seq=18 ttl=52 time=68.2 ms
64 bytes from 8.8.8.8: icmp_seq=19 ttl=52 time=60.4 ms
64 bytes from 8.8.8.8: icmp_seq=20 ttl=52 time=71.1 ms
64 bytes from 8.8.8.8: icmp_seq=21 ttl=52 time=71.3 ms

```

### ب - (2) پیکربندی NAT

سوال 3- با تحلیل بسته های صادره از سوی h1 توضیح دهید که چرا توانستید با سرور گوگل ارتباط برقرار نمایید.

سوال 4- در h3-eth1 چه مشاهده می شود؟ علت عدم موفقیت چیست؟

اگر از ترمینال h1 گوگل را پینگ کنیم همه پکت ها loss می شوند به این دلیل که network لokaL است و باید برای درخواست دادن به گوگل پکت ها را تغییر دهیم.

```

root@TCPPIP-VM:~/Desktop/shared/Session5# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
11 packets transmitted, 0 received, 100% packet loss, time 10082ms
root@TCPPIP-VM:~/Desktop/shared/Session5#

```

سوال 5- دستور iptables -t nat مناسب برای ایجاد یک پیکربندی NAT مناسب در h3 را بیان کنید

با دستور زیر h3 را به عنوان NAT router تنظیم کردیم تا ارتباط با شبکه اینترنت برای h1 و h2 برقرار شود.

iptables -t nat -A POSTROUTING -o h3-eth1 -j MASQUERADE

```

root@TCPPIP-VM:~/Desktop/shared/Session5# iptables -t nat -A POSTROUTING -o h3-eth1 -j MASQUERADE
root@TCPPIP-VM:~/Desktop/shared/Session5#

```

h1

```
root@TCPiP-VM:~/Desktop/shared/Session5# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=50 time=50.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=50 time=49.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=50 time=67.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=50 time=56.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=50 time=67.4 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=50 time=63.4 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=50 time=43.9 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=50 time=70.7 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=50 time=61.0 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=50 time=57.7 ms
^C
--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 43.911/58.846/70.774/8.364 ms
root@TCPiP-VM:~/Desktop/shared/Session5#
```

h1

```
root@TCPiP-VM:~/Desktop/shared/Session5# sudo echo nameserver 8.8.8.8 > /etc/resolve.conf
root@TCPiP-VM:~/Desktop/shared/Session5# ping www.google.com -c 1
PING www.google.com (216.239.38.120) 56(84) bytes of data.
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=1 ttl=51 time=94.2 ms
--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 94.273/94.273/94.273/0.000 ms
root@TCPiP-VM:~/Desktop/shared/Session5#
```

سوال 6 - هنگام پینگ کردن از طرف h3 بر چه اساسی پاسخ های ICMP دریافتی را برای h1 می فرستد؟  
 (راهنمایی: فکر می کنید چه فیلدی در بسته های درخواست/پاسخ ICMP برای شناسایی سورس حقیقی بسته ها مورد استفاده قرار می گیرد؟)  
 با توجه به انجام عمل NAT در h3 یک جدول به نام NAT-table ایجاد می شود که در آن برای h1 و h2 شناسه هایی در نظر گرفته می شود که بتواند برای آی پی private آنها را یک آی پی public در نظر بگیرد بدین ترتیب بسته هایی که از طرف h1 ارسال می شوند مقدار آنها به یک به یک آی پی پابلیک تغییر پیدا می کند تا در شبکه اینترنت درآپ نشود همچنین هنگام دریافت شدن بسته نیز h3 این شناسه را با توجه به جدول تغییر می دهد و آی پی مقصد متناظر با آی پی پرایویت h1 می شود تا پیام به مقصد ارسال شود.

## تمرین سری پنجم

آزمایشگاه شبکه های کامپیوتر

بکتاش انصاری 99521082

پوریا رحیمی 99521289

سوال ۱:

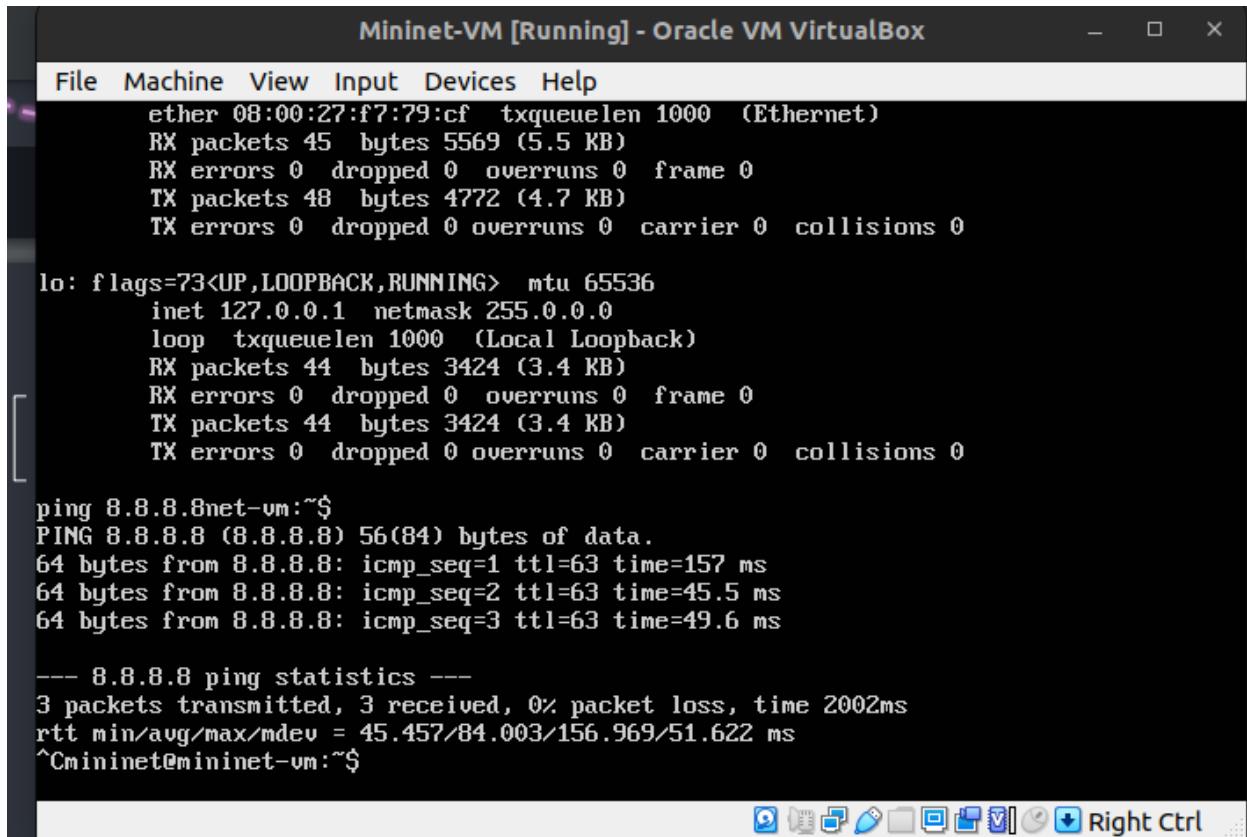
از آنجایی که ما تنها یک آدرس آیپی در اختیار داریم پس بهترین روش این است که راتر h3 را به شبکه اینترنت متصل کنیم. برای این کار کافیست یک آدرس آیپی واقعی برای اینترفیس h3-eht1 در نظر بگیریم تا این اینترفیس بتواند در شبکه اینترنت مشخص باشد. گام بعدی مورد نظر ما این است که چون h1 و h2 داخل شبکه ساختگی ما هستند باید به یک روشی پکت ها و ترافیک های ارسالی از این دو هاست را مسیریابی کنیم. که میتوانیم از IP iptable و Masquerading استفاده کنیم.

گام بعدی استفاده از Bridging میباشد. ما باید یک پل میان اینترفیس h3-eht1 و اینترفیس و درنهایت یک ip آدرس خوب پیدا کنیم و آن را روی این اینترفیس ست کنیم.

با این کار پکت های ارسالی از طرف هاست ها به راتر h3 میرستند و توسط پکت های واقعی ایجاد میشوند و توسط brdg به خارج از شبکه لوکال ارسال میشوند و فرآیند برعکس نیز اعمال میشود. بدین شکل شبکه لوکال ما به اینترنت متصل میشود.

## سوال (۲)

ابتدا **network** ویرچوال ماشین را روی NAT قرار میدهیم.  
حال وقتی ۸.۸.۸.۸ را پینگ کنیم میبینیم که ارتباط برقرار است.



```
Mininet-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
ether 08:00:27:f7:79:cf txqueuelen 1000 (Ethernet)
RX packets 45 bytes 5569 (5.5 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 48 bytes 4772 (4.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 44 bytes 3424 (3.4 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 44 bytes 3424 (3.4 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

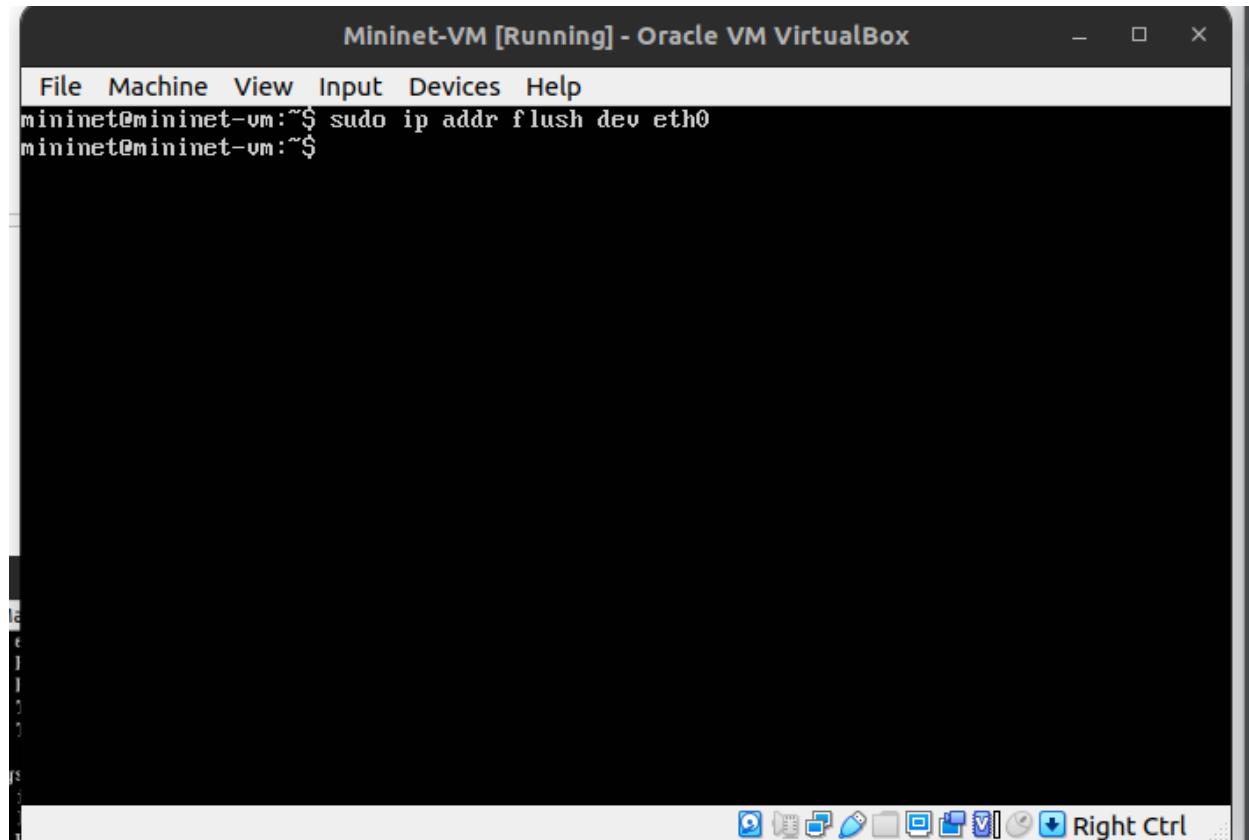
ping 8.8.8.8net-vm:~$ 
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=157 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=45.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=49.6 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 45.457/84.003/156.969/51.622 ms
^Cmininet@mininet-vm:~$
```

حال آیپی اینترفیس را حذف میکنیم.

Mininet-VM [Running] - Oracle VM VirtualBox

```
File Machine View Input Devices Help
mininet@mininet-vm:~$ sudo ip addr flush dev eth0
mininet@mininet-vm:~$
```



حال توپولوژی را اجرا میکنیم.

همانطور که میبینید در توپولوژی ساخته شده اگر از h3 گوگل را پینگ بگیریم پیام **network is unreachable** نمایش داده میشود.

Mininet-VM [Running] - Oracle VM VirtualBox

```
File Machine View Input Devices Help
*** Adding hosts
*** Adding router
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2 h3 r1
*** Starting controller
c0
*** Starting 2 switches
s12 s3 ...
*** Configuring hosts
*** Starting xterm on hosts
*** Running the command line interface
*** Starting CLI:
mininet> h3 ping 8.8.8.8
Warning: This program is an suid-root program or is being run by the root user.
The full text of the error or warning message cannot be safely formatted
in this environment. You may get a more descriptive message by running the
program as a non-root user or by removing the suid bit on the executable.
xterm: Xt error: Can't open display: :0
xterm: DISPLAY is not set
ping: connect: Network is unreachable
mininet>
```



Right Ctrl

حال با دستور `ovs eth0` اینترفیس `eth0` را به پل `3` اضافه میکنیم.

```
        is_connected: true
    fail_mode: secure
    Port s3-eth1
        Interface s3-eth1
    Port eth0
        Interface eth0
    Port s3
        Interface s3
        type: internal
Bridge s12
    Controller "tcp:127.0.0.1:6653"
        is_connected: true
    fail_mode: secure
    Port s12-eth1
        Interface s12-eth1
    Port s12-eth3
        Interface s12-eth3
    Port s12
        Interface s12
        type: internal
    Port s12-eth2
        Interface s12-eth2
    ovs_version: "2.13.1"
mininet>
```

حال با استفاده از دستور `dhclient` یک آیپی آدرس قابل استفاده برای اینترفیس `eth1` برای قرار میدهیم.

حال بعد از این کار 8.8.8 را پینگ میگیریم و متوجه میشویم ارتباط برقرار است.

```
mininet> h3 dhclient h3-eth1
mininet> h3 ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=54.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=47.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=58.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=72.6 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=63 time=129 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 47.534/72.570/129.075/29.409 ms
mininet>
```

ب) پیکربندی NAT

ابتدا از h1 گوگل را پینگ میکنیم.

هنگامی که پینگ میکنیم ریسپانسی دریافت نمیکنیم و تمام پکت ها loss میشوند. دلیل این هم این است که پکت ها درون شبکه لوکال ما معنا پیدا میکنند و ما باید این پکت ها را جعل کنیم.

```
mininet> h1 ping 8.8.8.8
Warning: This program is an suid-root program or is being run by the root user.
The full text of the error or warning message cannot be safely formatted
in this environment. You may get a more descriptive message by running the
program as a non-root user or by removing the suid bit on the executable.
xterm: Xt error: Can't open display: %s
xterm: DISPLAY is not set
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
49 packets transmitted, 0 received, 100% packet loss, time 49152ms
mininet> _
```

برای فرآیند جعل پکت یا **ip masquerade** استفاده میکنیم تا این جدول را در **h3** تغییر دهیم.

پس از این کار میبینیم که میتوانیم با استفاده از **h1** گوگل را پینگ کنیم.

```
mininet> h3 iptables -t nat -A POSTROUTING -o h3-eth1 -j MASQUERADE
mininet> h1 ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=59 time=865 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=59 time=79.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=59 time=47.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=59 time=43.4 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 43.435/259.027/865.438/350.393 ms
mininet>
```

حال در نهايٰت ميتوانيم dns سرور را نيز کانفيگ کنيم.

```
mininet> h1 sudo echo nameserver 8.8.8.8 > /etc/resolv.conf
mininet> h1 ping google.com
PING google.com (216.239.38.120) 56(84) bytes of data.
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=1 ttl=59 time=469 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=2 ttl=59 time=73.9 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=3 ttl=59 time=205 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=4 ttl=59 time=153 ms
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=5 ttl=59 time=112 ms
^C
--- google.com ping statistics ---
5 packets transmitted, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 73.921/202.527/468.638/139.987 ms
mininet>
```

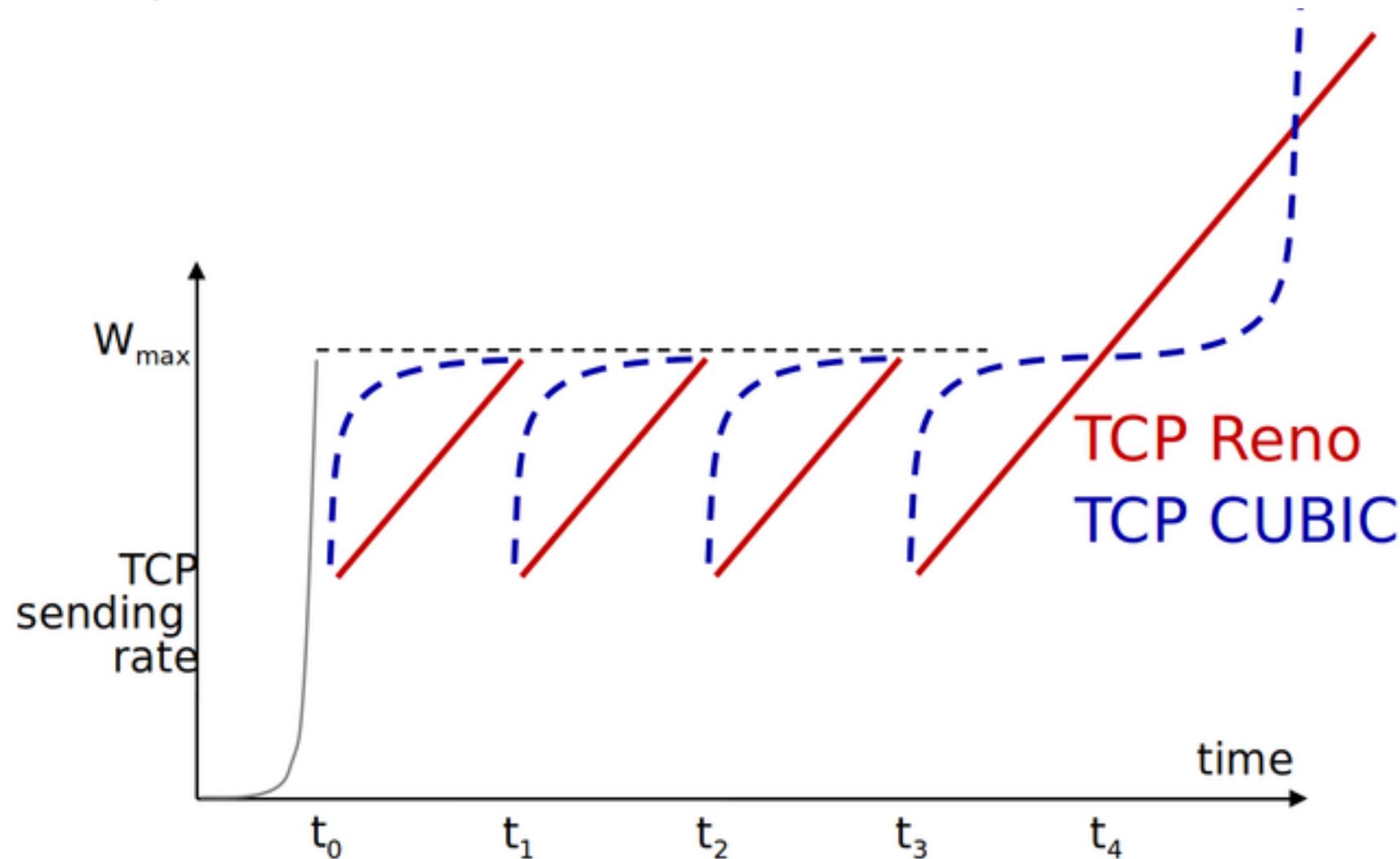
حال میبینیم که با استفاده از نام دامنه نیز میتوان 8.8.8.8 را پینگ کرد.

# Congestion Control

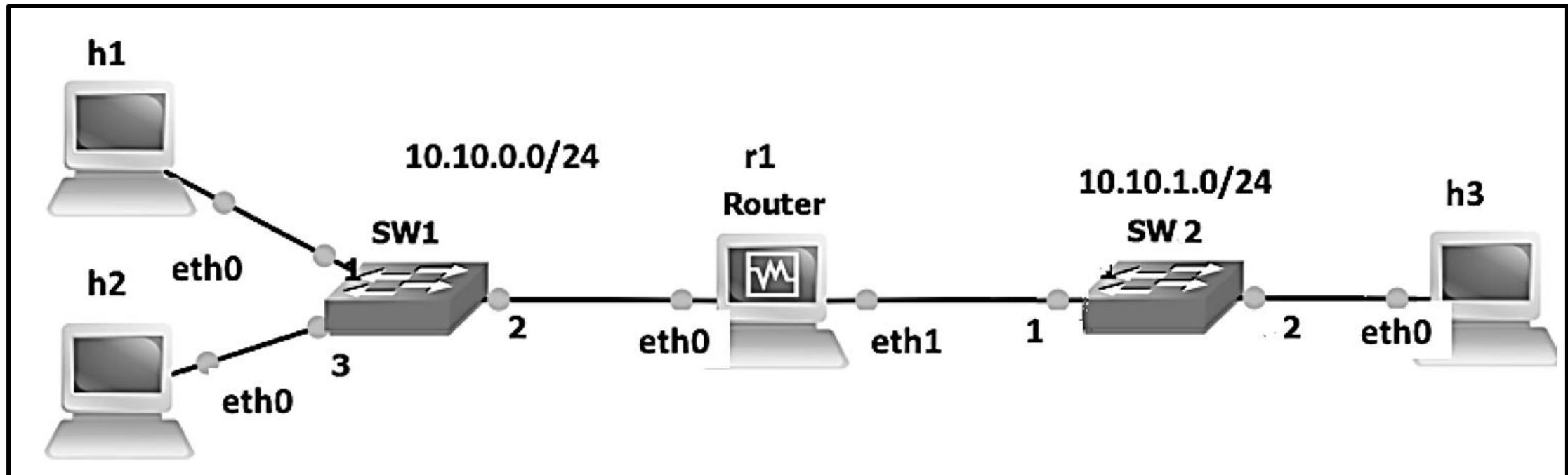
# Transport layer protocols

- Transmission Control Protocol (TCP)
  - Connection-oriented
  - Three-way handshake
  - Congestion control
  - Flow control
  - Acknowledgement mechanism
  - Slow
- User Datagram Protocol (UDP)
  - Connection-less
  - No handshake mechanism
  - No congestion control
  - No flow control
  - No acknowledgement
  - Fast

# TCP congestion control



# lab6.py



# UDP connection

```
"host: h3" x
root@mininet-vm:/home/mininet/Downloads/lab5# cd udp
root@mininet-vm:/home/mininet/Downloads/lab5/udp# ./udpserver 10000
```

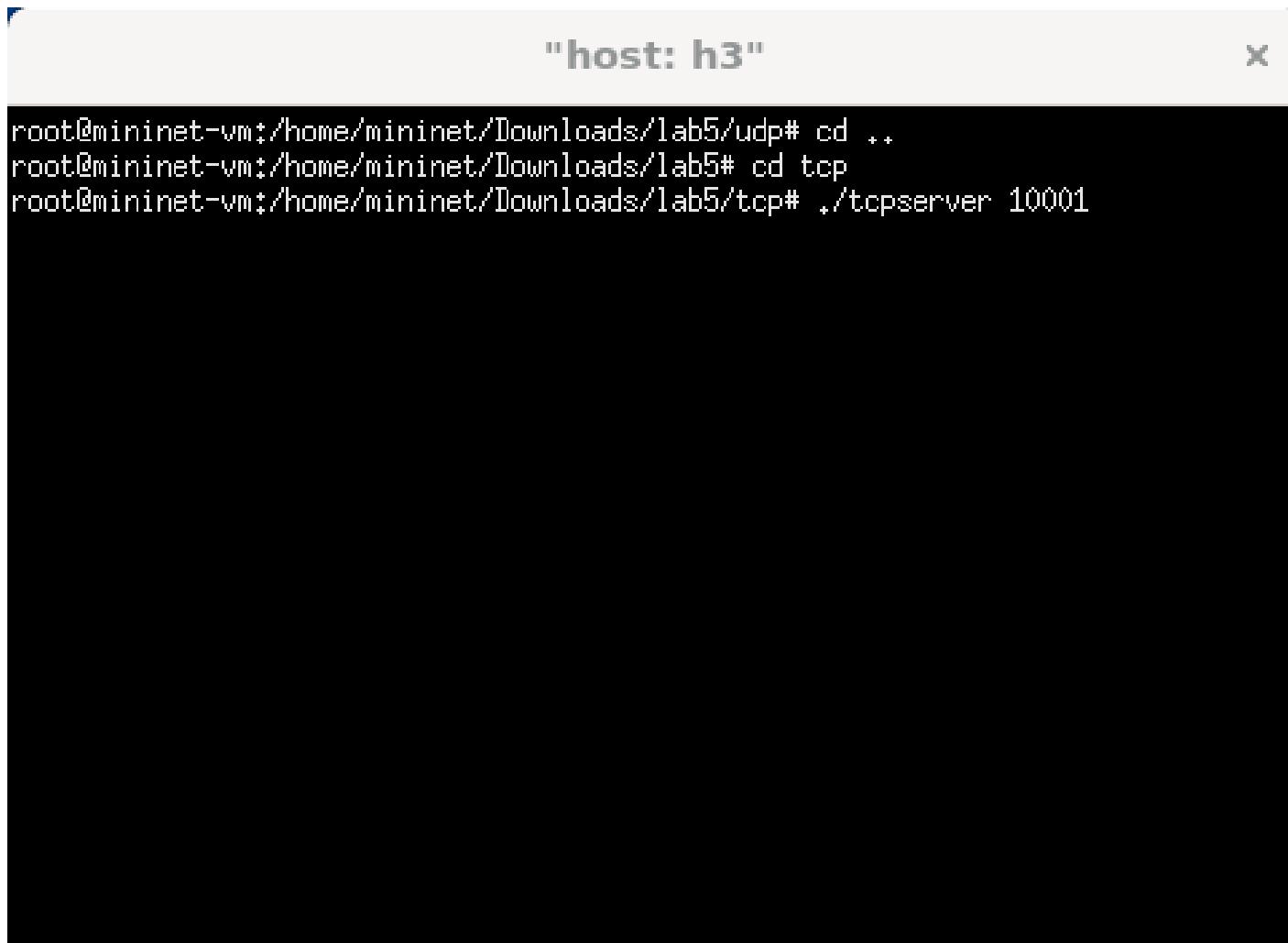
## "host: h1"

```
root@mininet-vm:/home/mininet/Downloads/lab5# cd udp
root@mininet-vm:/home/mininet/Downloads/lab5/udp# ./udpclient 10.10.1.3 10000 100
1.0s - sent: 14 pkts, 107.6 kbytes/s
2.1s - sent: 27 pkts, 100.0 kbytes/s
3.1s - sent: 40 pkts, 99.8 kbytes/s
4.2s - sent: 53 pkts, 102.6 kbytes/s
5.2s - sent: 66 pkts, 100.0 kbytes/s
6.2s - sent: 79 pkts, 100.1 kbytes/s
7.3s - sent: 92 pkts, 100.3 kbytes/s
8.3s - sent: 105 pkts, 100.1 kbytes/s
9.4s - sent: 118 pkts, 100.2 kbytes/s
10.4s - sent: 131 pkts, 100.1 kbytes/s
11.4s - sent: 144 pkts, 100.1 kbytes/s
12.5s - sent: 157 pkts, 100.2 kbytes/s
13.5s - sent: 170 pkts, 100.2 kbytes/s
14.6s - sent: 183 pkts, 100.1 kbytes/s
15.6s - sent: 196 pkts, 100.1 kbytes/s
16.6s - sent: 209 pkts, 100.2 kbytes/s
17.7s - sent: 222 pkts, 100.3 kbytes/s
18.7s - sent: 235 pkts, 99.9 kbytes/s
19.8s - sent: 248 pkts, 100.1 kbytes/s
20.8s - sent: 261 pkts, 100.1 kbytes/s
21.8s - sent: 274 pkts, 100.3 kbytes/s
```

## "host: h3"

```
root@mininet-vm:/home/mininet/Downloads/lab5# cd udp
root@mininet-vm:/home/mininet/Downloads/lab5/udp# ./udpserver 10000
 0.0s - received:  1/ sent:  1 pkts (loss 0.000%),  0.2 kbit/s
 47.0s - received: 14/ sent: 14 pkts (loss 0.000%), 100.3 kbit/s
 48.0s - received: 27/ sent: 27 pkts (loss 0.000%), 100.0 kbit/s
 49.0s - received: 40/ sent: 40 pkts (loss 0.000%), 100.0 kbit/s
 50.1s - received: 53/ sent: 53 pkts (loss 0.000%), 100.0 kbit/s
 51.1s - received: 66/ sent: 66 pkts (loss 0.000%), 100.0 kbit/s
 52.2s - received: 79/ sent: 79 pkts (loss 0.000%), 100.0 kbit/s
 53.2s - received: 92/ sent: 92 pkts (loss 0.000%), 99.9 kbit/s
 54.2s - received: 105/ sent: 105 pkts (loss 0.000%), 100.1 kbit/s
 55.3s - received: 118/ sent: 118 pkts (loss 0.000%), 100.0 kbit/s
 56.3s - received: 131/ sent: 131 pkts (loss 0.000%), 100.0 kbit/s
 57.4s - received: 144/ sent: 144 pkts (loss 0.000%), 100.0 kbit/s
 58.4s - received: 157/ sent: 157 pkts (loss 0.000%), 100.0 kbit/s
 59.4s - received: 170/ sent: 170 pkts (loss 0.000%), 100.0 kbit/s
 60.5s - received: 183/ sent: 183 pkts (loss 0.000%), 100.0 kbit/s
 61.5s - received: 196/ sent: 196 pkts (loss 0.000%), 100.0 kbit/s
 62.6s - received: 209/ sent: 209 pkts (loss 0.000%), 100.0 kbit/s
 63.6s - received: 222/ sent: 222 pkts (loss 0.000%), 99.8 kbit/s
 64.6s - received: 235/ sent: 235 pkts (loss 0.000%), 100.2 kbit/s
 65.7s - received: 248/ sent: 248 pkts (loss 0.000%), 100.0 kbit/s
 66.7s - received: 261/ sent: 261 pkts (loss 0.000%), 100.0 kbit/s
 67.8s - received: 274/ sent: 274 pkts (loss 0.000%), 100.0 kbit/s
```

# TCP connection



The image shows a terminal window titled "host: h3". The terminal window has a white header bar with the title and a black body. The command history is as follows:

```
root@mininet-vm:/home/mininet/Downloads/lab5/udp# cd ..
root@mininet-vm:/home/mininet/Downloads/lab5# cd tcp
root@mininet-vm:/home/mininet/Downloads/lab5/tcp# ./tcpserver 10001
```

## "host: h1"

```
root@mininet-vm:/home/mininet/Downloads/lab5/udp# cd ..
root@mininet-vm:/home/mininet/Downloads/lab5# cd tcp
root@mininet-vm:/home/mininet/Downloads/lab5/tcp# ./tcpclient 10.10.1.3 10001
 0.0:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 10 rtt  1.9ms
 0.0:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 18 rtt  8.8ms
 0.0:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 30 rtt 11.5ms
 0.0:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 44 rtt 17.0ms
 0.2:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 13 rtt 22.1ms
 0.3:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 19 rtt 20.1ms
 0.4:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 23 rtt 25.1ms
 0.5:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 15 rtt 16.1ms
 0.6:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 20 rtt 21.5ms
 0.7:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 24 rtt 26.3ms
 0.8:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 16 rtt 16.7ms
 1.0:  0.0kbps avg (  0.0[inst],  0.0[mov,avg]) cwnd 21 rtt 22.6ms
 1.1: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 12 rtt 26.6ms
 1.2: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 20 rtt 21.7ms
 1.3: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 12 rtt 25.8ms
 1.5: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 19 rtt 20.2ms
 1.6: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 24 rtt 25.9ms
 1.7: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 16 rtt 16.7ms
 1.8: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 21 rtt 22.6ms
 1.9: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 12 rtt 25.3ms
 2.0: 9195.5kbps avg ( 9195.5[inst], 9195.5[mov,avg]) cwnd 18 rtt 18.9ms
```

Capturing from r1-eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
2	0.007003000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
3	0.016072000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
4	0.023665000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
5	0.031216000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
6	0.040005000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
7	0.048301000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
8	0.055022000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
9	0.063939000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
10	0.071510000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp

► Frame 1: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits) on interface 0

► Ethernet II, Src: 92:ce:13:1f:54:f1 (92:ce:13:1f:54:f1), Dst: c6:4e:62:e9:95:e1 (c6:4e:62:e9:95:e1)

► Internet Protocol Version 4, Src: 10.10.0.1 (10.10.0.1), Dst: 10.10.1.3 (10.10.1.3)

► User Datagram Protocol, Src Port: 58293 (58293), Dst Port: ndmp (10000)

► Data (1000 bytes)

0000	c6 4e 62 e9 95 e1 92 ce 13 1f 54 f1 08 00 45 00	.Nb.....T...E.
0010	04 04 21 1a 40 00 40 11 00 b8 0a 0a 00 01 0a 0a	...!@. @. ....
0020	01 03 e3 b5 27 10 03 f0 19 19 32 33 38 33 37 0a	.....'....23837.
0030	00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01	.....
0040	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01	.....
0050	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01	.....

Frame (frame), 1042 bytes

Packets: 6212 · Di... Profile: Default

Capturing from r1-eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
2	0.007003000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
3	0.016072000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
4	0.023665000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
5	0.031216000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
6	0.040005000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
7	0.048301000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
8	0.055022000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
9	0.063939000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
10	0.071510000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp

► Frame 1: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits) on interface 0

► Ethernet II, Src: 92:ce:13:1f:54:f1 (92:ce:13:1f:54:f1), Dst: c6:4e:62:e9:95:e1 (c6:4e:62:e9:95:e1)

► Internet Protocol Version 4, Src: 10.10.0.1 (10.10.0.1), Dst: 10.10.1.3 (10.10.1.3)

► User Datagram Protocol, Src Port: 58293 (58293), Dst Port: ndmp (10000)

► Data (1000 bytes)

0000	c6 4e 62 e9 95 e1 92 ce 13 1f 54 f1 08 00 45 00	.Nb.....T...E.
0010	04 04 21 1a 40 00 40 11 00 b8 0a 0a 00 01 0a 0a	...!@. @. ....
0020	01 03 e3 b5 27 10 03 f0 19 19 32 33 38 33 37 0a	....'....23837.
0030	00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01	.....
0040	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01	.....
0050	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01	.....

Ethernet (eth), 14 bytes

Packets: 6212 · Di... Profile: Default

Capturing from r1-eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
2	0.007003000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
3	0.016072000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
4	0.023665000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
5	0.031216000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
6	0.040005000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
7	0.048301000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
8	0.055022000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
9	0.063939000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
10	0.071510000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp

► Frame 1: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits) on interface 0

► Ethernet II, Src: 92:ce:13:1f:54:f1 (92:ce:13:1f:54:f1), Dst: c6:4e:62:e9:95:e1 (c6:4e:62:e9:95:e1)

► Internet Protocol Version 4, Src: 10.10.0.1 (10.10.0.1), Dst: 10.10.1.3 (10.10.1.3)

► User Datagram Protocol, Src Port: 58293 (58293), Dst Port: ndmp (10000)

► Data (1000 bytes)

0000 c6 4e 62 e9 95 e1 92 ce 13 1f 54 f1 08 00 45 00 .Nb.....T...E.  
0010 04 04 21 1a 40 00 40 11 00 b8 0a 0a 00 01 0a 0a ...!@. ....  
0020 01 03 e3 b5 27 10 03 f0 19 19 32 33 38 33 37 0a ....'....23837.  
0030 00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....  
0040 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....  
0050 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....

Internet Protocol Version 4 (ip), 20 bytes Packets: 6212 · Di... Profile: Default

Capturing from r1-eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
2	0.007003000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
3	0.016072000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
4	0.023665000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
5	0.031216000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
6	0.040005000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
7	0.048301000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
8	0.055022000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
9	0.063939000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
10	0.071510000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp

► Frame 1: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits) on interface 0

► Ethernet II, Src: 92:ce:13:1f:54:f1 (92:ce:13:1f:54:f1), Dst: c6:4e:62:e9:95:e1 (c6:4e:62:e9:95:e1)

► Internet Protocol Version 4, Src: 10.10.0.1 (10.10.0.1), Dst: 10.10.1.3 (10.10.1.3)

► User Datagram Protocol, Src Port: 58293 (58293), Dst Port: ndmp (10000)

► Data (1000 bytes)

0020 01 03 e3 b5 27 10 03 f0 19 19 32 33 38 33 37 0a . . . . . 23837.

0030 00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....

0040 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....

0050 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....

0060 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....

0070 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....

○ User Datagram Protocol (udp), 8 bytes

Packets: 6212 · Di... Profile: Default

Capturing from r1-eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
2	0.007003000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
3	0.016072000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
4	0.023665000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
5	0.031216000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
6	0.040005000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
7	0.048301000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
8	0.055022000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
9	0.063939000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp
10	0.071510000	10.10.0.1	10.10.1.3	UDP	1042	Source port: 58293 Destination port: ndmp

► Frame 1: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits) on interface 0

► Ethernet II, Src: 92:ce:13:1f:54:f1 (92:ce:13:1f:54:f1), Dst: c6:4e:62:e9:95:e1 (c6:4e:62:e9:95:e1)

► Internet Protocol Version 4, Src: 10.10.0.1 (10.10.0.1), Dst: 10.10.1.3 (10.10.1.3)

► User Datagram Protocol, Src Port: 58293 (58293), Dst Port: ndmp (10000)

► Data (1000 bytes)

0020 01 03 e3 b5 27 10 03 f0 19 19 32 33 38 33 37 0a ....'....23837.  
0030 00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....  
0040 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....  
0050 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....  
0060 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....  
0070 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....

○ Data (data), 1000 bytes

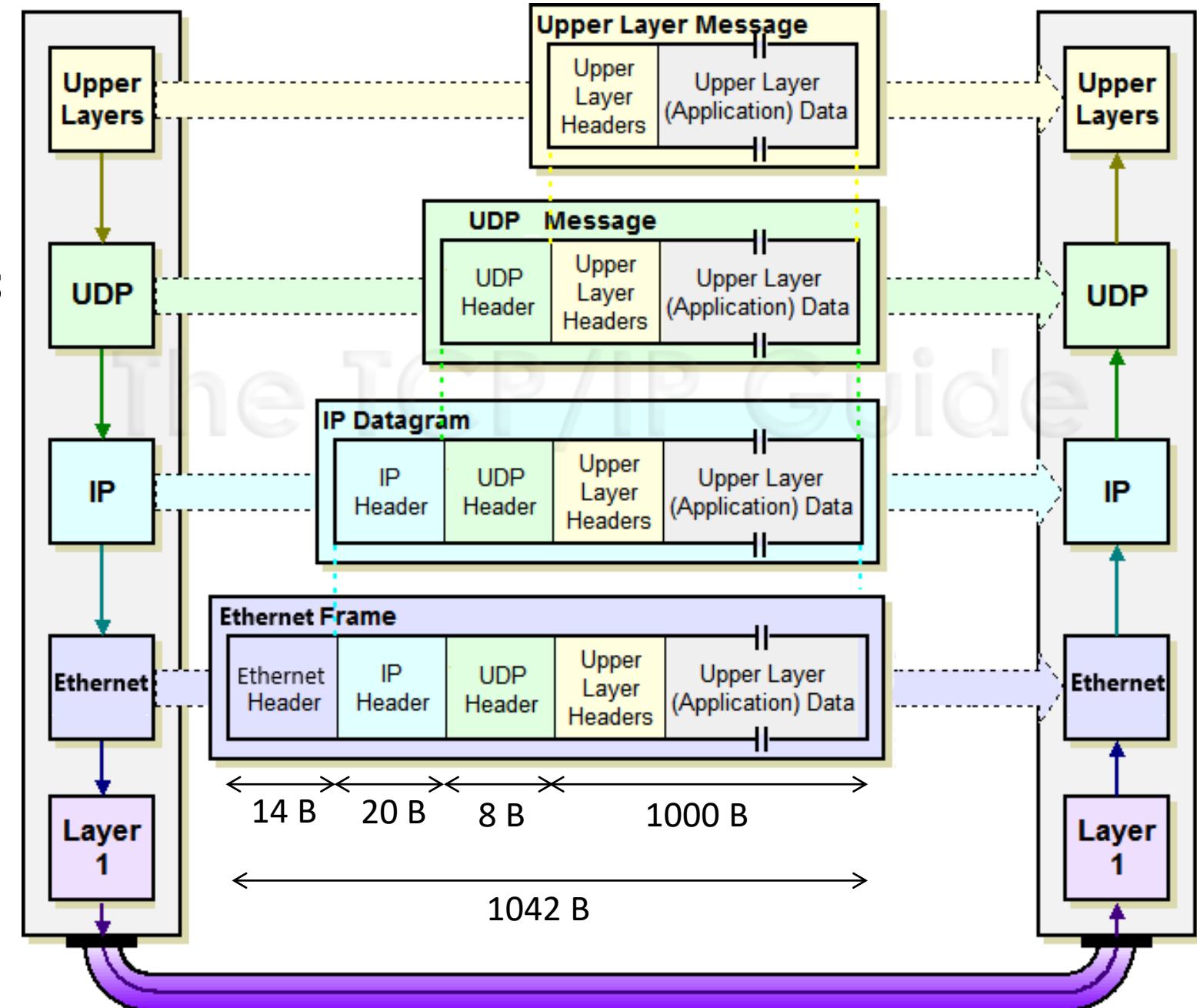
Packets: 6212 · Di... Profile: Default

# UDP connection

Max total rate = 3 Mbps

Goodput:

$$3 * \frac{1000}{1042} = 2879 \text{ kbps}$$



Capturing from r1-eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
2	0.000025000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1489993 Ack=1 Win=58 Len=1448 T
3	0.000084000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=1449 Win=16320 Len=0 TSva
4	0.000098000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1491441 Ack=1 Win=58 Len=1448 T
5	0.000120000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=2897 Win=16318 Len=0 TSva
6	0.000133000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1492889 Ack=1 Win=58 Len=1448 T
7	0.0001375000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=4345 Win=16322 Len=0 TSva
8	0.001398000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1494337 Ack=1 Win=58 Len=1448 T
9	0.002566000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=5793 Win=16322 Len=0 TSva
10	0.002584000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1495785 Ack=1 Win=58 Len=1448 T
11	0.002750000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=7241 Win=16322 Len=0 TSva

► Frame 2: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0

► Ethernet II, Src: 92:ce:13:1f:54:f1 (92:ce:13:1f:54:f1), Dst: c6:4e:62:e9:95:e1 (c6:4e:62:e9:95:e1)

► Internet Protocol Version 4, Src: 10.10.0.1 (10.10.0.1), Dst: 10.10.1.3 (10.10.1.3)

► Transmission Control Protocol, Src Port: 37126 (37126), Dst Port: scp-config (10001), Seq: 1489993, Ack: 1, Len: 1448

► Data (1448 bytes)

0000	c6 4e 62 e9 95 e1 92 ce	13 1f 54 f1 08 00 45 00	.Nb..... T...E.
0010	05 dc 16 b6 40 00 40 06	09 4f 0a 0a 00 01 0a 0a	....@. @. 0.....
0020	01 03 91 06 27 11 3e fc	ab a3 5d 79 fa 70 80 10	....'.> ..]y.p..
0030	00 3a 1a e6 00 00 01 01	08 0a 00 58 57 2d 00 58	..... XW- .X
0040	57 2d 00 00 00 00 00 00	00 00 00 00 00 00 00 00	W-.....
0050	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....

Frame (frame), 1514 bytes

Packets: 13250 · Displayed: 13250 (100%)

Profile: Default

Capturing from r1-eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
2	0.000025000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1489993 Ack=1 Win=58 Len=1448 T
3	0.000084000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=1449 Win=16320 Len=0 TSva
4	0.000098000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1491441 Ack=1 Win=58 Len=1448 T
5	0.000120000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=2897 Win=16318 Len=0 TSva
6	0.000133000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1492889 Ack=1 Win=58 Len=1448 T
7	0.0001375000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=4345 Win=16322 Len=0 TSva
8	0.001398000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1494337 Ack=1 Win=58 Len=1448 T
9	0.002566000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=5793 Win=16322 Len=0 TSva
10	0.002584000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1495785 Ack=1 Win=58 Len=1448 T
11	0.002750000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=7241 Win=16322 Len=0 TSva

► Frame 2: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0

► Ethernet II, Src: 92:ce:13:1f:54:f1 (92:ce:13:1f:54:f1), Dst: c6:4e:62:e9:95:e1 (c6:4e:62:e9:95:e1)

► Internet Protocol Version 4, Src: 10.10.0.1 (10.10.0.1), Dst: 10.10.1.3 (10.10.1.3)

► Transmission Control Protocol, Src Port: 37126 (37126), Dst Port: scp-config (10001), Seq: 1489993, Ack: 1, Len: 1448

► Data (1448 bytes)

0000	c6 4e 62 e9 95 e1 92 ce	13 1f 54 f1 08 00 45 00	.Nb..... T...E.
0010	05 dc 16 b6 40 00 40 06	09 4f 0a 0a 00 01 0a 0a	....@. @. 0.....
0020	01 03 91 06 27 11 3e fc	ab a3 5d 79 fa 70 80 10	....'.>. .]y.p..
0030	00 3a 1a e6 00 00 01 01	08 0a 00 58 57 2d 00 58	.:..... XW-. X
0040	57 2d 00 00 00 00 00 00	00 00 00 00 00 00 00 00	W-.....
0050	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....

Ethernet (eth), 14 bytes

Packets: 13250 · Displayed: 13250 (100%)

Profile: Default

Capturing from r1-eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
2	0.000025000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1489993 Ack=1 Win=58 Len=1448 T
3	0.000084000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=1449 Win=16320 Len=0 TSva
4	0.000098000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1491441 Ack=1 Win=58 Len=1448 T
5	0.000120000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=2897 Win=16318 Len=0 TSva
6	0.000133000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1492889 Ack=1 Win=58 Len=1448 T
7	0.0001375000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=4345 Win=16322 Len=0 TSva
8	0.001398000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1494337 Ack=1 Win=58 Len=1448 T
9	0.002566000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=5793 Win=16322 Len=0 TSva
10	0.002584000	10.10.0.1	10.10.1.3	TCP	1514	37126 > scp-config [ACK] Seq=1495785 Ack=1 Win=58 Len=1448 T
11	0.002750000	10.10.1.3	10.10.0.1	TCP	78	scp-config > 37126 [ACK] Seq=1 Ack=7241 Win=16322 Len=0 TSva

► Frame 2: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0

► Ethernet II, Src: 92:ce:13:1f:54:f1 (92:ce:13:1f:54:f1), Dst: c6:4e:62:e9:95:e1 (c6:4e:62:e9:95:e1)

► Internet Protocol Version 4, Src: 10.10.0.1 (10.10.0.1), Dst: 10.10.1.3 (10.10.1.3)

► Transmission Control Protocol, Src Port: 37126 (37126), Dst Port: scp-config (10001), Seq: 1489993, Ack: 1, Len: 1448

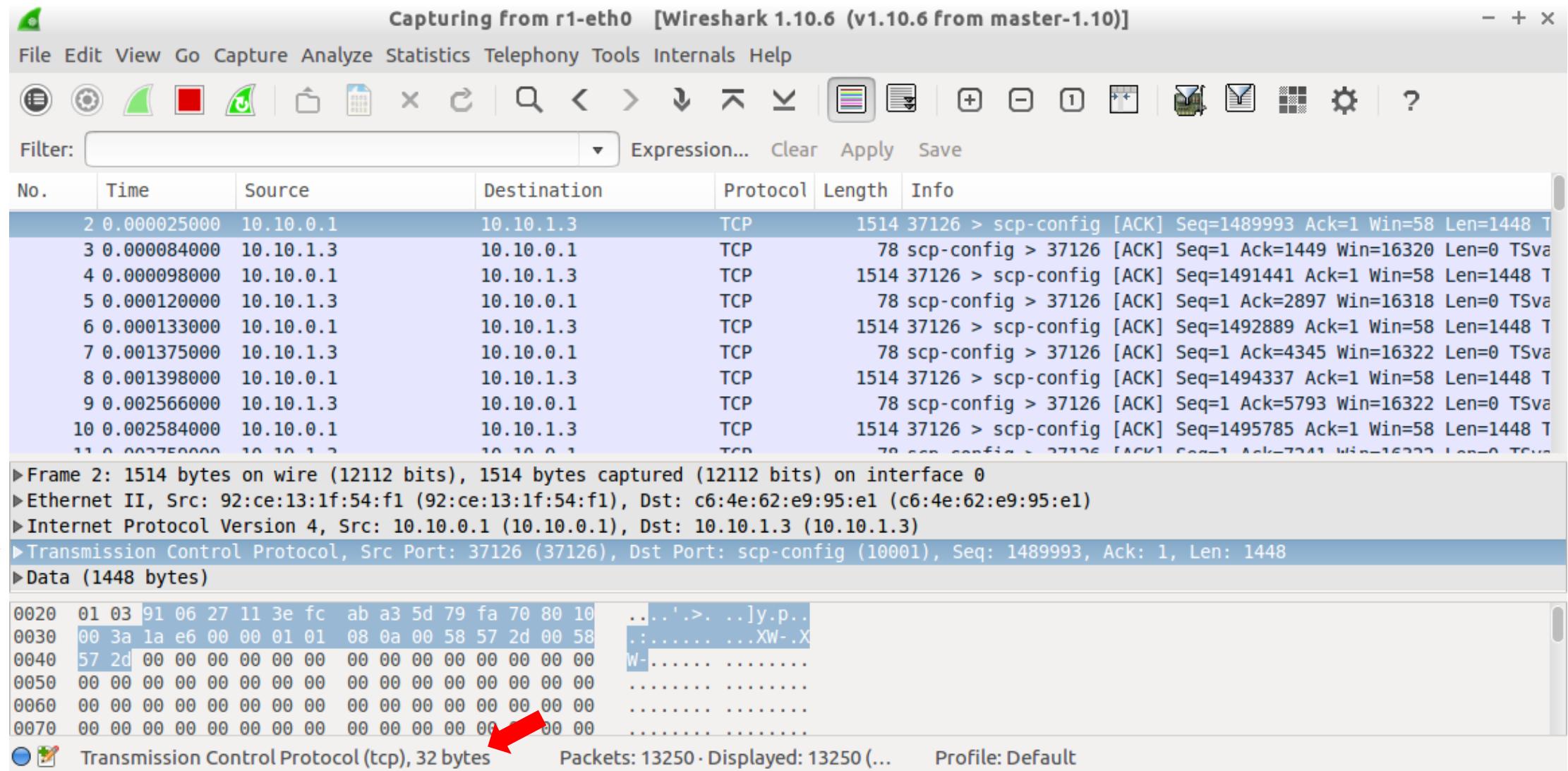
► Data (1448 bytes)

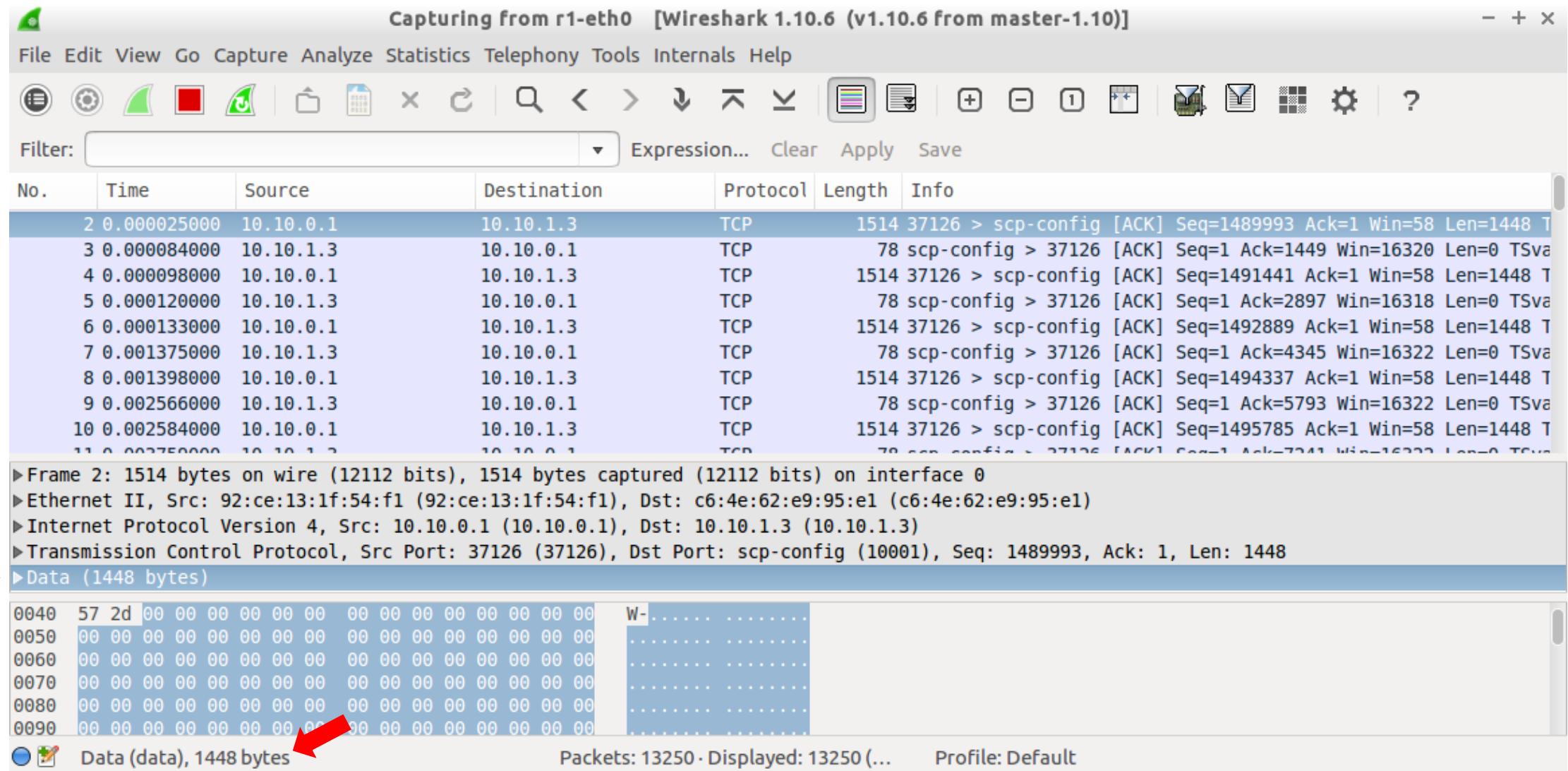
Hex	Dec	ASCII
0000	c6 4e 62 e9 95 e1 92 ce 13 1f 54 f1 08 00 45 00	.Nb..... T...E.
0010	05 dc 16 b6 40 00 40 06 09 4f 0a 0a 00 01 0a 0a	....@. @. .0.....
0020	01 03 91 06 27 11 3e fc ab a3 5d 79 fa 70 80 10	....'.>. .]y.p..
0030	00 3a 1a e6 00 00 01 01 08 0a 00 58 57 2d 00 58	.:..... . .XW-.X
0040	57 2d 00 00 00 00 00 00 00 00 00 00 00 00 00 00	W-.....
0050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

Internet Protocol Version 4 (ip), 20 bytes

Packets: 13250 · Displayed: 13250 (100%)

Profile: Default



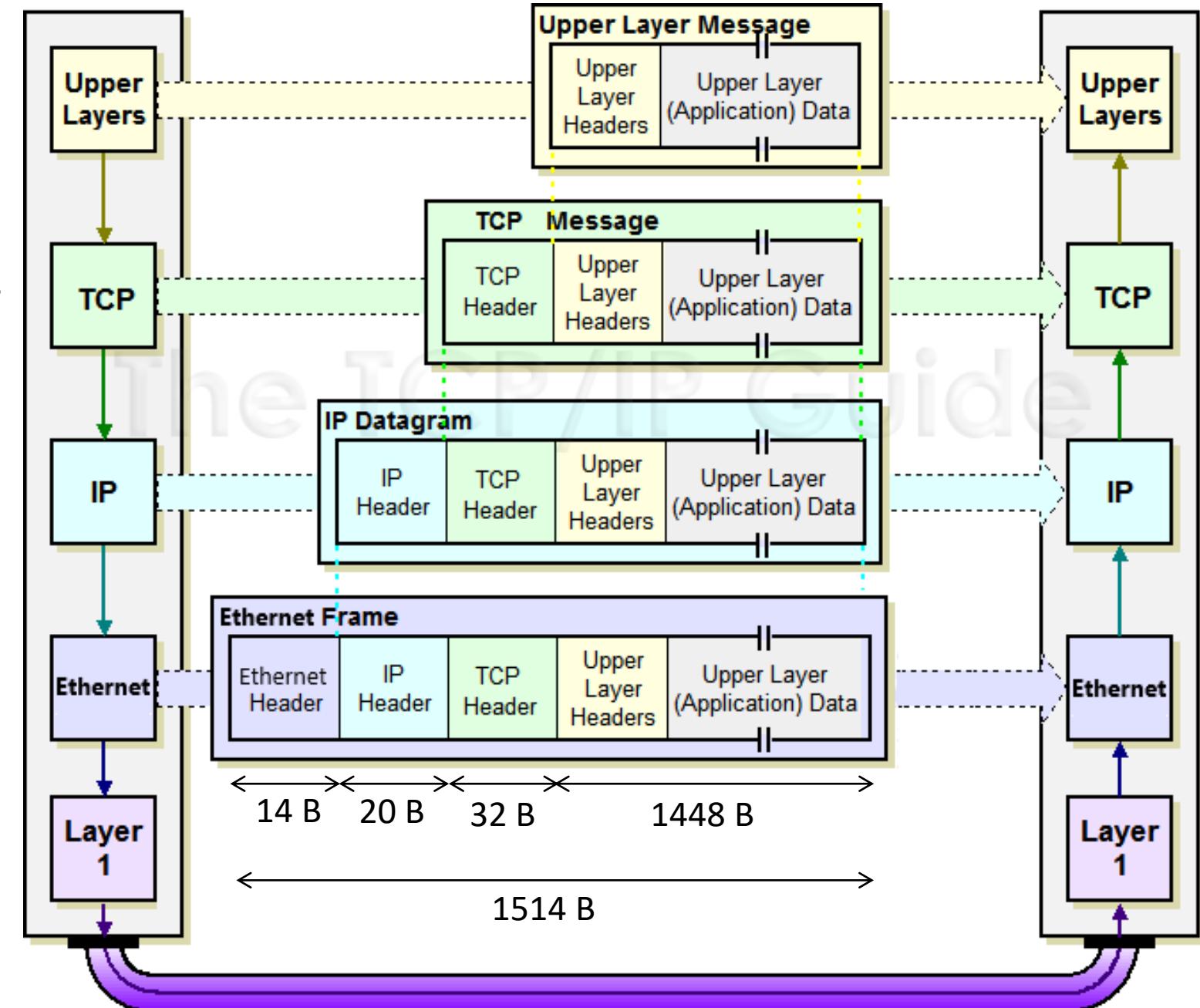


# TCP connection

Max total rate = 3 Mbps

Goodput:

$$3 * \frac{1448}{1514} = 2869 \text{ kbps}$$



## آزمایشگاه شبکه

### آزمایش ۶: کنترل ازدحام، UDP و TCP

در این آزمایش، مقدمات لازم برای راهاندازی و پیکربندی نشستهای ارتباطی UDP و TCP را فراهم می‌کنیم. در آزمایش‌های بعدی، مکانیزم کنترل ازدحام TCP را مورد بررسی قرار می‌دهیم.

#### الف) روال آزمایش

برنامه‌های مورد نیاز برای اجرای این آزمایش از فolder lab5 در دسترس هستند. این فolder شامل موارد زیر است:

a. فolder lab5 حاوی اسکریپت‌های مورد نیاز برای ساخت توپولوژی‌های مورد نظر می‌باشد.

b. فolders‌های lab5/udp و lab5/tcp حاوی کلاینت‌ها و سرورهای TCP و UDP هستند که برای اندازه‌گیری<sup>۱</sup> هر جریان مورد استفاده قرار خواهند گرفت.

ماشین مجازی را راهاندازی کرده و در صورت لزوم، با ورود به دایرکتوری lab5، با اجرای دستور زیر، برنامه‌ها را به حالت «اجرایی» درآورید:

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

مکانیزم کنترل ازدحام در ماشین مجازی را بررسی کنید. روی یک ماشین لینوکس، می‌توانید با تایپ دستور زیر بررسی کنید که چه مکانیزمی مورد استفاده است:

```
cat /proc/sys/net/ipv4/tcp_congestion_control
```

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

```
sudo su
```

```
echo reno >/proc/sys/net/ipv4/tcp_congestion_control
```

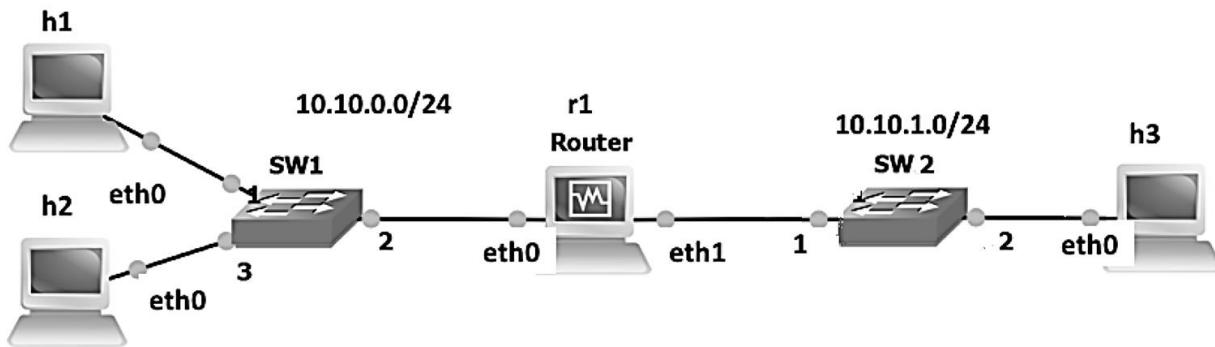
راهکار دیگر این است که مستقیماً محتوای فایل فوق را به reno تغییر دهید.

---

<sup>۱</sup> برای یک جریان از داده‌ها، goodput عبارتست از نرخ داده‌های application (یعنی، داده‌های مفید) که به طور موفقیت‌آمیز ارسال می‌شوند. برای سؤالاتی که در بخش‌های آتی مواجه می‌شوید، باید در نظر بگیرید که بسته‌ها علاوه بر داده‌های application دارای سرآیند (هدر) هم هستند.

## ب) جریان‌های TCP و UDP

اسکریپت موجود در فolder5، lab5.py، توپولوژی نمایش داده شده در شکل ۱ را می‌سازد.



شکل ۱- توپولوژی متشکل از سه PC و یک روتر

فایل lab5.py را طوری تکمیل کنید که جداول مسیریابی و آدرس‌های IP طبق روش آدرس‌دهی زیر پیکربندی شوند:

- سابنتِ ماشین‌های h1 و روتر r1 آدرس 10.10.0/24 است. آدرس‌های h2، h1 و روتر هم به ترتیب به صورت 10.10.0.1، 10.10.0.2 و 10.10.0.10 می‌باشد.
- سابنتِ ماشین h3 و روتر دارای آدرس 10.10.1.0/24 است. آدرس‌های h3 و روتر هم به ترتیب، 10.10.1.10 و 10.10.1.3 می‌باشد.

یک ترمینال در ماشین خود باز کنید و اسکریپت lab5.py را اجرا نمایید. پس از اجرای توپولوژی، با استفاده از دستور pingall، پیکربندی خود را آزمایش نمایید.

### ب-۱) تست اتصال بین کلاینت و سرورهای TCP و UDP

دایرکتوری lab5/udp حاوی دو برنامه است: udpclient و udpserver. نحوه استفاده از این برنامه‌ها به صورت زیر است:

```
# ./udpserver (PORT)  
# ./udpclient (IP_SERVER) (PORT) (RATE)
```

برای سرور، PORT همان شماره پورتی است که روی آن سرور مشغول گوش کردن می‌باشد. برای کلاینت IP\_SERVER و PORT به ترتیب، آدرس IP و شماره پورت ماشینی را مشخص می‌کند که بسته‌ها به مقصد آن ارسال می‌شوند و RATE هم نرخی را مشخص می‌کند که با آن داده‌ها توسط کلاینت‌ها ارسال می‌شوند (برحسب کیلوبیت در

ثانیه). اگر نرخ کمتر از 50 kbps باشد، کلاینیت‌ها بسته‌های به اندازه 125 بایتی ارسال می‌کنند و گرنه بسته‌هایی با اندازه 1000 بایت می‌فرستند.

خروجی کلاینیت UDP دارای فرمت زیر است<sup>۲</sup>:

```
4.0s - sent: 503 pkts, 1000.0 kbytes/s
5.0s - sent: 629 pkts, 1000.6 kbytes/s
```

به عنوان مثال، مقادیر خط دوم را می‌توان به این نحو تفسیر کرد که:

- ۵.۰ عبارتست از تعداد ثانیه‌های سپری شده از لحظه شروع به کار کلاینیت
- ۶۲۹ یعنی تعداد کل بسته‌های ارسالی توسط کلاینیت
- ۱۰۰۰.۶ نرخ ارسال طی آخرین ثانیه است (برحسب kbit/s).

خروجی سرور UDP دارای فرمت زیر است:

```
169.5s - received: 723/ sent: 741 pkts (loss 2.429%), 959.6 kbytes/s
170.5s - received: 843/ sent: 867 pkts (loss 2.768%), 957.7 kbytes/s
```

به عنوان مثال، مقادیر خط دوم را می‌توان به این نحو تفسیر کرد که:

- ۱۷۰.۵ زمانی است که از راهاندازی سرور می‌گذرد.
- ۸۴۳ یعنی تعداد کل بسته‌های دریافتی توسط سرور
- ۸۶۷ یعنی تعداد کل بسته‌های ارسالی توسط کلاینیت
- ۲.۷۶۸ یعنی درصد بسته‌هایی که از بین رفته‌اند
- ۹۵۷.۷ نرخی است که طی آخرین ثانیه، بسته‌ها با آن دریافت شده‌اند. به این مقدار، اصطلاحاً goodput طی آخرین ثانیه گفته می‌شود.

یک سرور UDP در h3 راهاندازی کنید که روی پورت 10000 گوش می‌کند.

\* نکته: با توجه به اینکه Mininet یک امولاتور است، اگر یک آزمایش را چندین مرتبه اجرا کنید، نتایج هر بار می‌تواند متفاوت باشد. به همین مناسبت، توصیه می‌شود که هر آزمایش را دو مرتبه اجرا نمایید و مقادیر میانگین را به عنوان جواب ارائه دهید.

<sup>۲</sup> برای کلیه موارد این آزمایش، مقادیر نرخ چاپ شده، از نوع داده‌های application هستند و حجم سرآیندها (هدرها) در آنها لحاظ نشده است.

**سؤال 1: روی ماشین h1 یک کلاینت UDP اجرا کنید که داده‌ها را برای سرور h3 با نرخ 100 kbps ارسال می‌کند. احتمال loss و goodput را مشاهده می‌شود، چقدر است؟**

**سؤال 2: آزمایش را با نرخ‌های 1 Gbps، 10 Mbps، 100 Mbps و 1 Mbps تکرار کنید. مقادیر goodput و احتمال‌های loss چقدر است؟ به ازای چه نرخی، احتمال loss بالاتر از 1% است؟ آیا می‌توانید نتایج را با توجه به گذرنامه lab5.py توجیه نمایید؟**

دایرکتوری lab5/tcp حاوی دو برنامه است: tcpserver و tcpclient. نحوه استفاده از آنها مشابه udpserver و udpclient است به جز اینکه ما نرخی را برای کلاینت مقرر نمی‌نماییم. در واقع، فرض می‌شود که کلاینت دارای حجم نامتناهی داده برای ارسال است و از مکانیزم کنترل ازدحام TCP برای کنترل نرخ ارسال داده‌ها به سوی سرور استفاده می‌شود.

```
# ./tcpserver (PORT)  
# ./tcpclient (IP_SERVER) (PORT)
```

خروجی کلاینت TCP چیزی شبیه به شکل زیر است:

```
6.3: 854.0 kbps avg ( 944.5[inst], 926.5[mov.avg] ) cwnd 9 rtt 83.9ms  
7.3: 862.4 kbps avg ( 914.6[inst], 925.3[mov.avg] ) cwnd 9 rtt 86.8ms
```

به عنوان مثال، مقادیر خط دوم را می‌توان به صورت زیر تفسیر کرد:

- 7.3 نمایانگر زمان است
- 862.4 میانگین نرخ کلاینت است: یعنی، کل حجم داده‌ای که به طور موفقیت‌آمیزی توسط کلاینت منتقل شده است تقسیم بر کل زمان (مقدار حاصل را می‌توان به عنوان مقدار میانگین goodput برای TCP نیز تعبیر نمود).
- 914.6 همان نرخ آنی است (تقریباً طی ثانیه آخر).
- 925.3 نیز مقدار میانگین متحرک (moving average) نرخ است.
- مقدار 9 اندازه «پنجره ازدحام» TCP را نشان می‌دهد.
- 86.8 یعنی RTT اندازه‌گیری شده توسط الگوریتم کنترل ازدحام TCP.
- یک سرور TCP روی ماشین h3 راهاندازی کنید که روی پورت 10001 گوش می‌دهد.<sup>۳</sup>

<sup>۳</sup> قبل از راهاندازی هر سرور و کلاینت جدید، کلیه کلاینت‌ها و سرورهای قبلی را kill کنید. این کار باعث می‌شود تا مقادیر متوسطی که چاپ می‌شوند، reset شوند.

### سؤال ۳: یک کلاینت TCP روی ماشین h1 اجرا نمایید که برای سرور h3 داده ارسال می‌کند. ارتباط چقدر است؟

\* نکته: برای تمامی آزمایش‌ها باید صبر کنید تا مقادیر چاپ شده به حالت پایداری برسند. این امر به خصوص برای TCP خیلی اهمیت دارد. نرخ ارسال داده‌ها در TCP به وقوع loss‌ها ربط دارد که آنها هم به صورت تصادفی رخ می‌دهند. به همین دلیل، برای دستیابی به مقادیر قطعی، باید صبر کنید تا نرخ متوسط به پایداری برسد (برای اغلب سناریوها حدود ۲ دقیقه کفایت می‌کند). ضمن اینکه هر آزمایش ترجیحاً باید دوبار تکرار شود.

#### ب-۲) محدودسازی پهنهای باند روتر

برای انجام آزمایش‌هایی که در آنها کارایی شبکه بابت ظرفیت برخی لینک‌ها دچار محدودیت است، در این بخش، پهنهای باند بعضی از اینترفیس‌ها را بیشتر محدود می‌نماییم.

مثالاً، به عنوان یک راهکار می‌توان از امکانات کلاس TCLink در Mininet و پارامترهایی که برای محدودسازی پهنهای باند در اختیار می‌گذارد، بهره گرفت. به طور مشخص، دستور زیر:

```
net.addLink(h1, h2, bw=5, delay='2ms', max_queue_size=1000, loss=1)
```

یک لینک دو طرفه بین ماشین‌های h1 و h2 با پهنهای باند 5 Mbps، تأخیر 2ms، احتمال loss برابر با 1% و حداکثر سایز صفحه 1000 بسته ایجاد می‌کند. پارامتر bw عددی را بر حسب Mbps مشخص می‌کند؛ تأخیر به صورت یک رشته با واحد (مانند '5ms' یا '100us' یا '1s') بیان می‌شود؛ احتمال loss بر حسب درصد (بین ۰ تا ۱۰۰) معین شده و max\_queue\_size هم بر حسب تعداد بسته‌ها تعیین می‌شود. برای اطلاعات بیشتر می‌توانید به آدرس [http://mininet.org/api/classmininet\\_1\\_1link\\_1\\_1TCLink.html](http://mininet.org/api/classmininet_1_1link_1_1TCLink.html) مراجعه نمایید.

به عنوان راهکار دیگر، در اسکریپت lab5.py که در بخش قبلی آن را تکمیل و اجرا نمودید، دستوری به صورت زیر وجود دارد:

```
link_r1sw2.intf1.config(bw=10)
```

این دستور صرفاً پهنهای باند اینترفیس eth1 از روتر را روی 10 Mbps تنظیم می‌کند (به جای اینکه پهنهای باند کل لینک را محدود سازد). با این وجود، حداکثر ترافیکی که قابل انتقال خواهد بود، مشابه زمانی است که شما پهنهای باند کل لینک را به 10 Mbps تنظیم نمایید (مثلاً از طریق دستور `link_r1sw2=net.addLink(r1, sw2, bw=10)` که در زمان اجرای یک لینک قابل استفاده است).

- پهنهای باند اینترفیس eth1 از روتر را به 3 Mbps محدود سازید.

\* توجه: برای محقق شدن تغییر پهنهای باند، باید از Mininet خارج شده، توپولوژی قبلی را clean-up کرده و سپس مجدداً اسکریپت lab5.py را اجرا نمایید.

\* توجه: محدودیت 3 Mbps برای بسته‌های وارد و هم خارجه از اینترفیس eth1 روتر اعمال خواهد شد چراکه کلاس TCLink، اینترفیس‌های متقارن ایجاد می‌کند. در آزمایش‌های آتی، توضیح خواهیم داد که چگونه با استفاده از دستورات لینوکس می‌توان محدودیت‌های پهنهای باند را به صورت نامتقارن بر مبنای زمان‌بندی CBQ<sup>4</sup> ایجاد کرد.

## ب-۲) تست UDP

یه یاد داریم که کلاینت UDP در شرایطی که RATE را روی مقداری بزرگتر از 50 kbps تنظیم کنیم، بسته‌هایی هر کدام به اندازه 1000 بایت ارسال خواهد نمود.

**سؤال ۴:** به لحاظ تئوری، انتظار داریم اندازه (بر حسب بایت) فریم‌های Ethernet که برای ارسال داده‌های کلاینت استفاده می‌شوند، به صورت زیر باشد:

هدر (سرآیند) UDP به میزان ۱ بایت، هدر IP به مقدار ۲۰ بایت، هدر Ethernet به میزان ۱۴ بایت و اندازه داده‌های Application هم که ۱۰۰۰ بایت، پس، مجموعاً ۱۰۴۲ بایت.

با استفاده از WireShark همخوانی واقعیت با این مقدار تئوری را کنترل کنید.

**سؤال ۵:** پس از اعمال محدودیت 3 Mbps در پهنهای باند، روتر در توپولوژی شکل ۱ تبدیل به گلوگاه (bottleneck) شبکه می‌شود. به لحاظ تئوری، حداقل مقدار قابل دستیابی برای گذردهی داده‌های کاربردی (همان goodput) چقدر خواهد بود؟ محاسبه کنید.

• یک سرور UDP روی ماشین h3 راهاندازی کنید که روی پورت 10000 گوش می‌کند.

**سؤال ۶:** یک کلاینت UDP روی ماشین h1 راهاندازی نمایید که داده‌ها را با نرخ 100 kbps ارسال می‌کند. مقدار احتمال loss و همچنین goodput مشاهده شده در h3 چقدر است؟

**سؤال ۷:** عملیات مورد نظر در سؤال ۶ را برای نرخ‌های 3 Mbps و 10 Mbps انجام دهید. مقادیر goodput و احتمالات loss چقدر می‌شود؟ مقادیر حاصل برای goodput در همه این موارد را با مقداری که انتظار دارید (و در سؤال ۵ محاسبه کردید)، مقایسه نمایید.

<sup>4</sup> Class-Based Queuing (CBQ)

## ب-۲-۲) تست TCP

سؤال ۸: انتظار داریم اندازه (بر حسب بایت) فریم‌های اترنتی که برای ارسال داده‌ها توسط ارتباط TCP استفاده می‌شوند برابر با ۱۵۱۴ بایت باشد چراکه: با توجه به MTU داده‌های برنامه کاربردی ۱۴۴۸ بایت، هدیر IP ۲۰ بایت، هدیر اترنت ۱۴ بایت و هدیر TCP برابر با ۳۲ بایت است. این فرضیه را از طریق گوش دادن به بسته‌ها در سمت سرور بررسی کنید.

\* توجه: اگر قابلیت TCP segmentation offload در یک ماشین فعال باشد (که گاهی به آن، TSO یا LSO نیز گفته می‌شود)، سیستم‌عامل بسته‌های بزرگتر از MTU را به کارت شبکه می‌دهد و درایور کارت شبکه نیز برای جا دادن بسته‌ها در MTU، آنها را خُرد می‌نماید. اگر بتوان بسته‌ها را مستقیماً از رسانه شنود کرد (به جای اینکه از endpoint ارتباط شنود کنیم)، خواهیم دید که بسته‌ها با اندازه درست (همان ۱۵۱۴ بایت) ارسال می‌شوند. قابلیت TSO نوعی بهبود روی عملکرد TCP است ولی امکان غیرفعال کردن آن نیز وجود دارد تا سیستم‌عامل، دیگر فریم‌های oversized تولید نکند. برای این منظور در h1 دستور زیر را اجرا نمایید:

```
ethtool -K h1-eth0 tx off sg off tso off
```

سؤال ۹: به لحاظ تئوری، حداکثر مقدار مورد انتظار برای goodput داده‌های کاربردی چقدر است؟ نحوه محاسبه خود را تشریح کنید.

• یک سرور TCP روی ماشین h3 راه‌اندازی کنید که روی پورت ۱۰۰۰۱ گوش می‌کند.

سؤال ۱۰: یک کلاینت TCP روی ماشین h1 راه‌اندازی کنید که داده‌هایی را برای سرور واقع در h3 می‌فرستد. ارتباط چقدر است؟ آن را با مقدار تئوری مورد سؤال ۹ مقایسه نمایید.

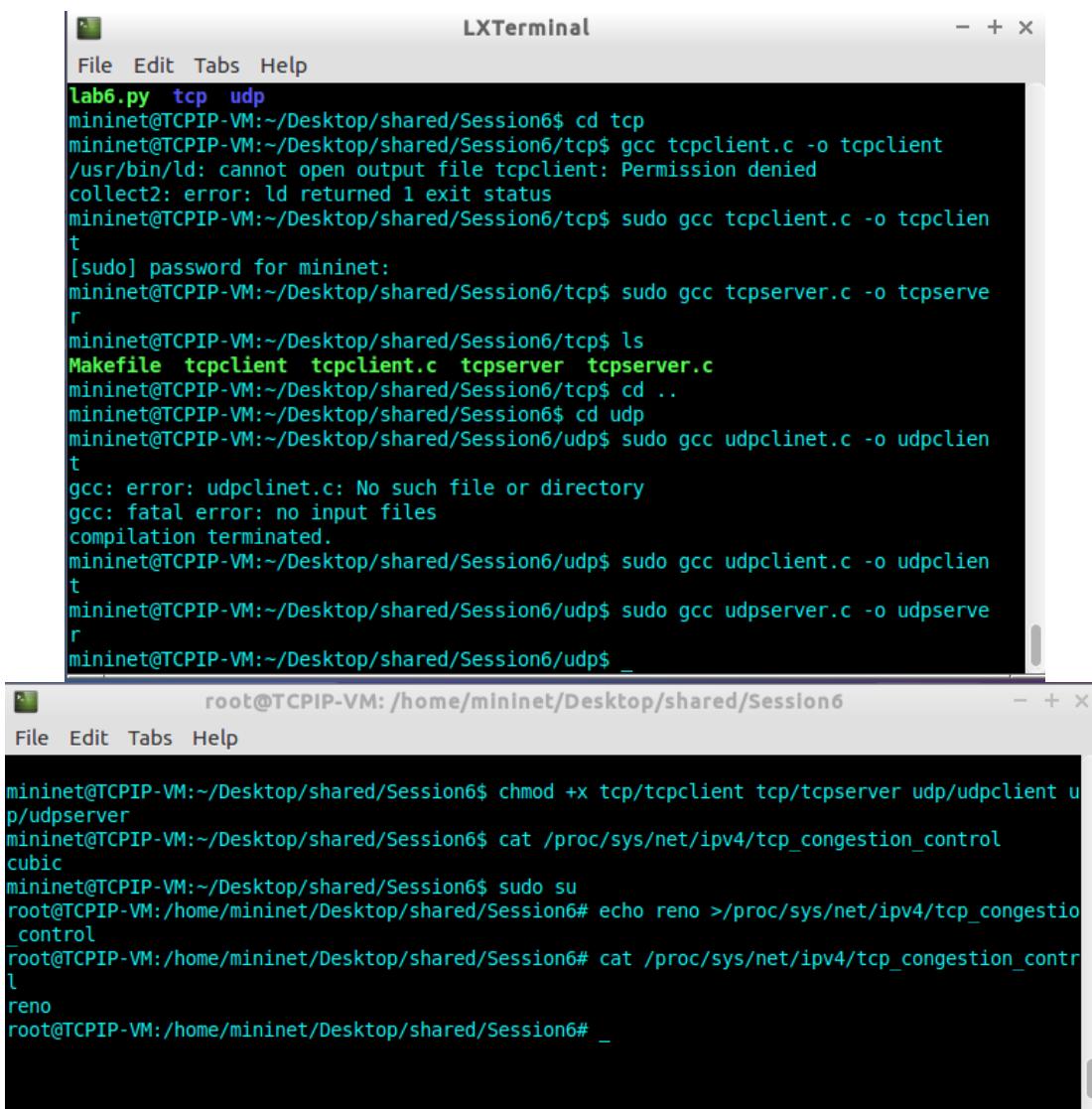
به نام خدا

## گزارشکار آزمایش 6: کنترل ازدحام TCP و UDP

بهاره کاووسی نژاد – 99431217

سیده شکیبا انارکی فیروز – 99442047

الف) روال آزمایش



```
File Edit Tabs Help
lab6.py tcp udp
mininet@TCPIP-VM:~/Desktop/shared/Session6$ cd tcp
mininet@TCPIP-VM:~/Desktop/shared/Session6/tcp$ gcc tcpclient.c -o tcpclient
/usr/bin/ld: cannot open output file tcpclient: Permission denied
collect2: error: ld returned 1 exit status
mininet@TCPIP-VM:~/Desktop/shared/Session6/tcp$ sudo gcc tcpclient.c -o tcpclient
[sudo] password for mininet:
mininet@TCPIP-VM:~/Desktop/shared/Session6/tcp$ sudo gcc tcpserver.c -o tcpserver
mininet@TCPIP-VM:~/Desktop/shared/Session6/tcp$ ls
Makefile tcpclient tcpclient.c tcpserver tcpserver.c
mininet@TCPIP-VM:~/Desktop/shared/Session6/tcp$ cd ..
mininet@TCPIP-VM:~/Desktop/shared/Session6$ cd udp
mininet@TCPIP-VM:~/Desktop/shared/Session6/udp$ sudo gcc udpclient.c -o udpclient
gcc: error: udpclient.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.
mininet@TCPIP-VM:~/Desktop/shared/Session6/udp$ sudo gcc udpclient.c -o udpclient
mininet@TCPIP-VM:~/Desktop/shared/Session6/udp$ sudo gcc udpserver.c -o udpserver
mininet@TCPIP-VM:~/Desktop/shared/Session6/udp$ _

root@TCPIP-VM: /home/mininet/Desktop/shared/Session6
File Edit Tabs Help
mininet@TCPIP-VM:~/Desktop/shared/Session6$ chmod +x tcp/tcpclient tcp/tcpserver udp/udpclient udp/udpserver
mininet@TCPIP-VM:~/Desktop/shared/Session6$ cat /proc/sys/net/ipv4/tcp_congestion_control
cubic
mininet@TCPIP-VM:~/Desktop/shared/Session6$ sudo su
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# echo reno >/proc/sys/net/ipv4/tcp_congestion_control
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# cat /proc/sys/net/ipv4/tcp_congestion_control
reno
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# _
```

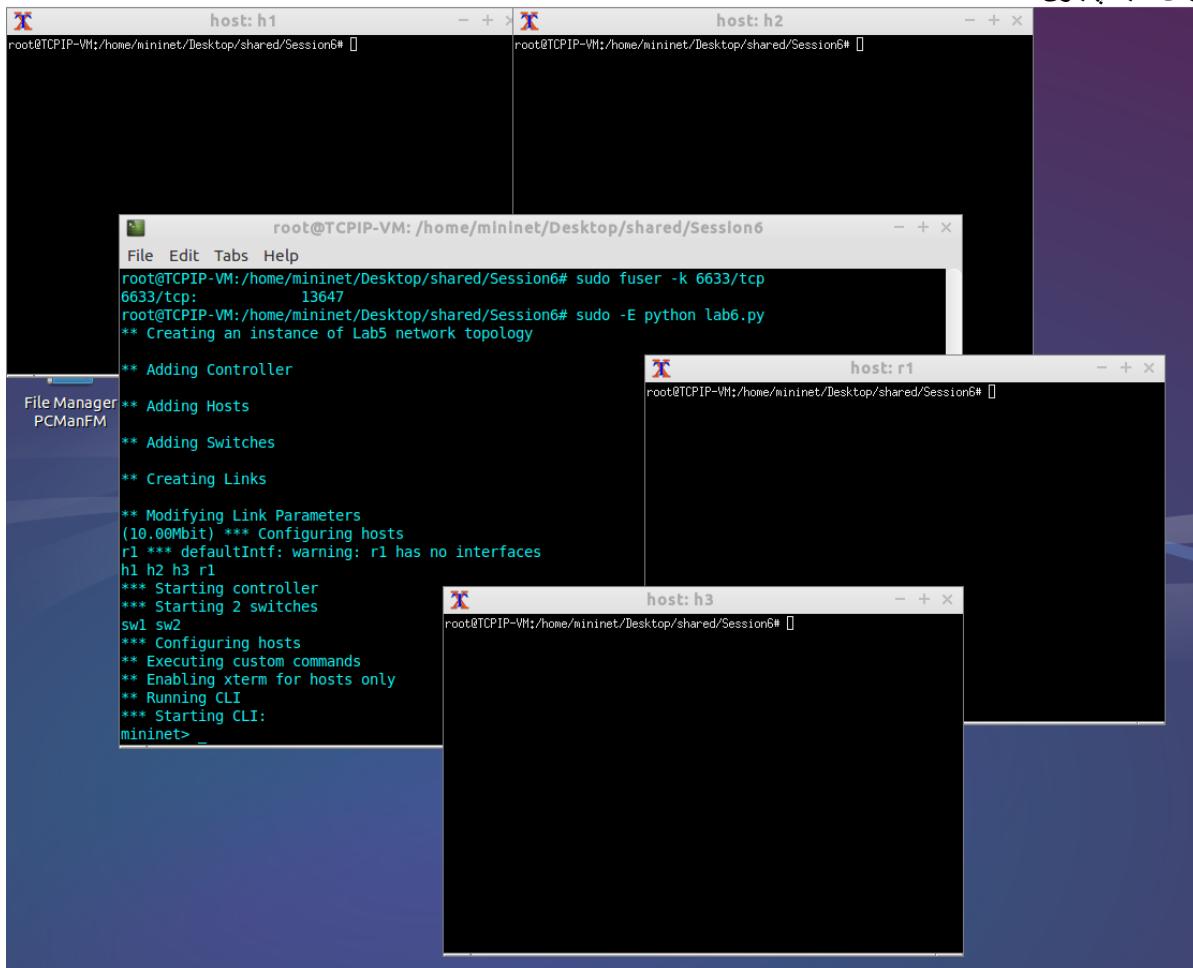
## ب) جریان های TCP و UDP

های Configuration Configuration را به شکل زیر در فایل پایتون انجام می دهیم:

```
info('** Adding Hosts\n')
r1 = net.addHost('r1', ip='10.10.0.10/24', hostname='r1')
# Space to add any commands for configuring the IP
h1 = net.addHost('h1', ip='10.10.0.1/24')
h2 = net.addHost('h2', ip='10.10.0.2/24')
h3 = net.addHost('h3', ip='10.10.1.3/24')
r1 = net.addHost('r1', ip='10.10.0.10/24')

info('*** Configuring hosts\n')
r1.cmd('ifconfig r1-eth1 10.10.1.10 netmask 255.255.255.0')
# Space to add commands for configuring routing tables and default gateway
r1.cmd ('ifconfig r1-eth1 10.10.1.10 netmask 255.255.255.0')
r1.cmd ('echo 1 >/proc/sys/net/ipv4/ip_forward')
h1.cmd ('ip route add default via 10.10.0.10')
h2.cmd ('ip route add default via 10.10.0.10')
h3.cmd ('ip route add default via 10.10.1.10')
```

اجرای فایل پایتون:



## پس از اجرای دستور :pingall

```
root@TCPIP-VM: /home/mininet/Desktop/shared/Session6
File Edit Tabs Help
r1 -> X X X X
h1 -> *** defaultIntf: warning: r1 has no interfaces
Traceback (most recent call last):
  File "lab6.py", line 147, in <module>
    run()
  File "lab6.py", line 127, in run
    CLI(net)
  File "build/bdist.linux-x86_64/egg/mininet/cli.py", line 67, in __init__
  File "/usr/lib/python2.7/cmd.py", line 142, in cmdloop
    stop = self.onecmd(line)
  File "/usr/lib/python2.7/cmd.py", line 221, in onecmd
    return func(arg)
  File "build/bdist.linux-x86_64/egg/mininet/cli.py", line 156, in do_pingall
  File "build/bdist.linux-x86_64/egg/mininet/net.py", line 592, in pingAll
  File "build/bdist.linux-x86_64/egg/mininet/net.py", line 506, in ping
  File "build/bdist.linux-x86_64/egg/mininet/node.py", line 456, in IP
AttributeError: 'NoneType' object has no attribute 'IP'
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# sudo fuser -K 6633/tcp
6633/tcp:      21133
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# sudo -E python lab6.py
** Creating an instance of Lab5 network topology

** Adding Controller

** Adding Hosts

** Adding Switches

** Creating Links

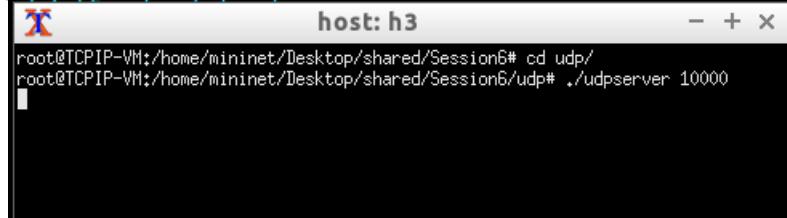
** Modifying Link Parameters
(3.00Mbit) *** Configuring hosts
h1 h2 h3 r1
*** Starting controller
*** Starting 2 switches
sw1 sw2
*** Configuring hosts
** Executing custom commands
** Enabling xterm for hosts only
** Running CLI
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 r1
h2 -> h1 h3 r1
h3 -> h1 h2 r1
r1 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> _
```

ب-1) تست اتصال بین کلاینت و سرورهای TCP و UDP

یک سرور UDP در h3 راه اندازی کنید که روی پورت 10000 گوش می کند.

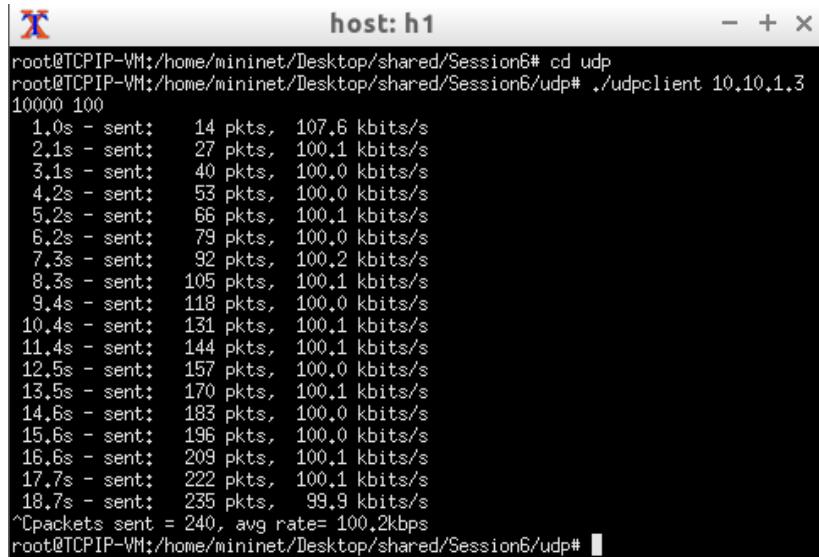
سؤال 1: روی ماشین h1 یک کلاینت UDP اجرا کنید که داده ها را برای سرور h3 با نرخ 100 kbps ارسال می کند. احتمال loss و goodput مشاهده می شود، چقدر است؟

یک سرور UDP در h3 که روی پورت 10000 گوش می کند، راه اندازی کرده و روی h1 یک client ایجاد می کنیم که برای این port از h3 داده می فرستد. برای افزایش اطمینان این آزمایش را دوبار انجام می دهیم:

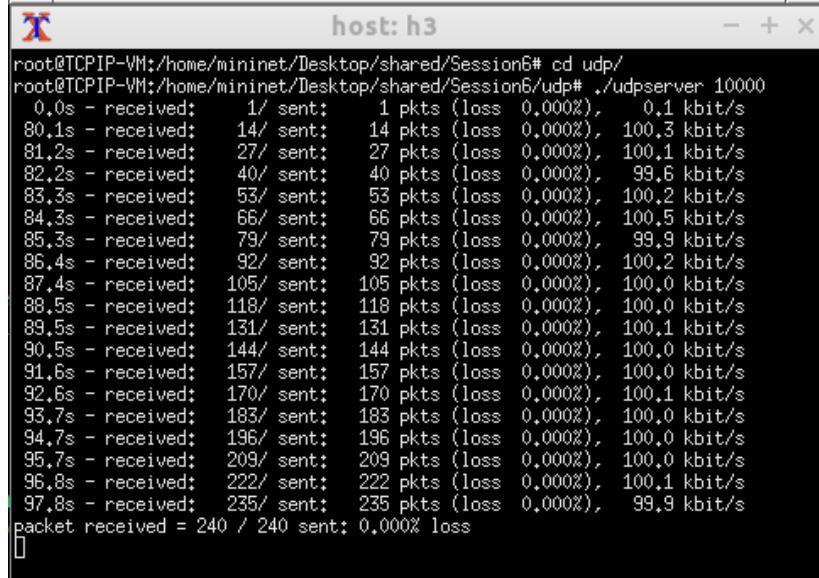


```
host: h3
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# cd udp/
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpserver 10000
```

نتایج انجام آزمایش برای بار اول:

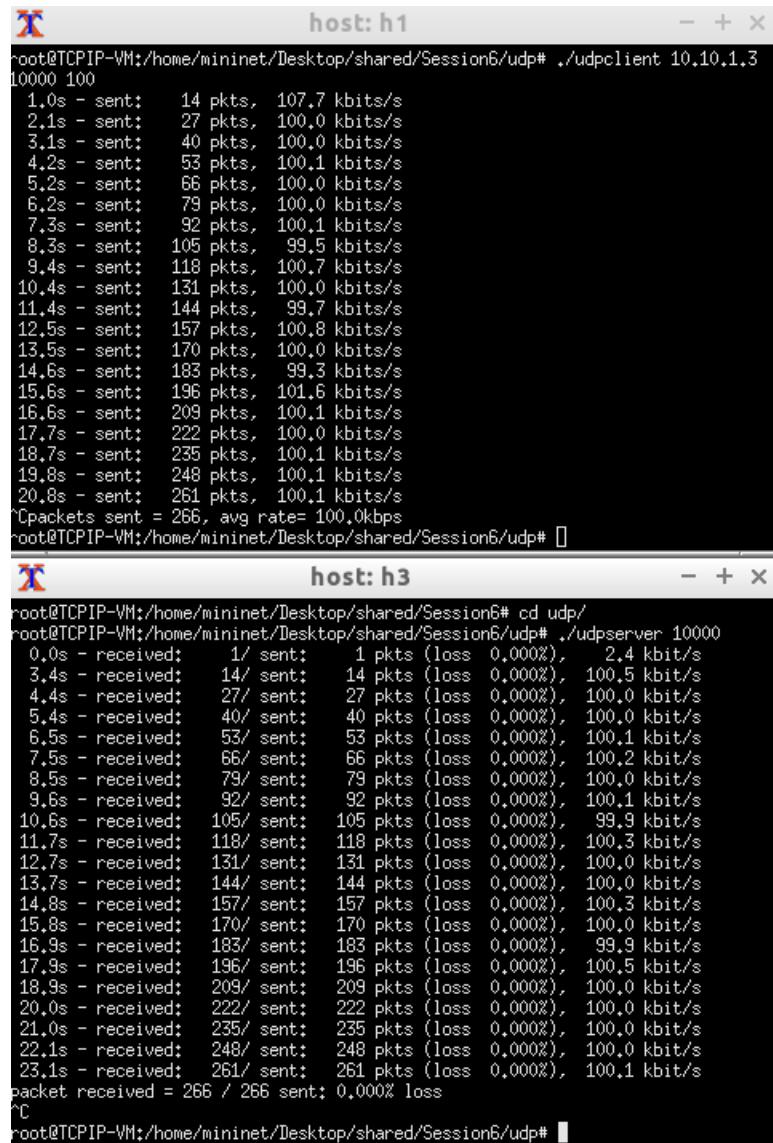


```
host: h1
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# cd udp/
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpclient 10.10.1.3 10000 100
 1.0s - sent: 14 pkts, 107.6 kbytes/s
 2.1s - sent: 27 pkts, 100.1 kbytes/s
 3.1s - sent: 40 pkts, 100.0 kbytes/s
 4.2s - sent: 53 pkts, 100.0 kbytes/s
 5.2s - sent: 66 pkts, 100.1 kbytes/s
 6.2s - sent: 79 pkts, 100.0 kbytes/s
 7.3s - sent: 92 pkts, 100.2 kbytes/s
 8.3s - sent: 105 pkts, 100.1 kbytes/s
 9.4s - sent: 118 pkts, 100.0 kbytes/s
10.4s - sent: 131 pkts, 100.1 kbytes/s
11.4s - sent: 144 pkts, 100.1 kbytes/s
12.5s - sent: 157 pkts, 100.0 kbytes/s
13.5s - sent: 170 pkts, 100.1 kbytes/s
14.6s - sent: 183 pkts, 100.0 kbytes/s
15.6s - sent: 196 pkts, 100.0 kbytes/s
16.6s - sent: 209 pkts, 100.1 kbytes/s
17.7s - sent: 222 pkts, 100.1 kbytes/s
18.7s - sent: 235 pkts, 99.9 kbytes/s
^Cpackets sent = 240, avg rate= 100.2kbytes/s
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp#
```



```
host: h3
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# cd udp/
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpserver 10000
 0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 0.1 kbit/s
80.1s - received: 14/ sent: 14 pkts (loss 0.000%), 100.3 kbit/s
81.2s - received: 27/ sent: 27 pkts (loss 0.000%), 100.1 kbit/s
82.2s - received: 40/ sent: 40 pkts (loss 0.000%), 99.6 kbit/s
83.3s - received: 53/ sent: 53 pkts (loss 0.000%), 100.2 kbit/s
84.3s - received: 66/ sent: 66 pkts (loss 0.000%), 100.5 kbit/s
85.3s - received: 79/ sent: 79 pkts (loss 0.000%), 99.9 kbit/s
86.4s - received: 92/ sent: 92 pkts (loss 0.000%), 100.2 kbit/s
87.4s - received: 105/ sent: 105 pkts (loss 0.000%), 100.0 kbit/s
88.5s - received: 118/ sent: 118 pkts (loss 0.000%), 100.0 kbit/s
89.5s - received: 131/ sent: 131 pkts (loss 0.000%), 100.1 kbit/s
90.5s - received: 144/ sent: 144 pkts (loss 0.000%), 100.0 kbit/s
91.6s - received: 157/ sent: 157 pkts (loss 0.000%), 100.0 kbit/s
92.6s - received: 170/ sent: 170 pkts (loss 0.000%), 100.1 kbit/s
93.7s - received: 183/ sent: 183 pkts (loss 0.000%), 100.0 kbit/s
94.7s - received: 196/ sent: 196 pkts (loss 0.000%), 100.0 kbit/s
95.7s - received: 209/ sent: 209 pkts (loss 0.000%), 100.0 kbit/s
96.8s - received: 222/ sent: 222 pkts (loss 0.000%), 100.1 kbit/s
97.8s - received: 235/ sent: 235 pkts (loss 0.000%), 99.9 kbit/s
packet received = 240 / 240 sent: 0.000% loss
```

## نتایج انجام آزمایش برای بار دوم:



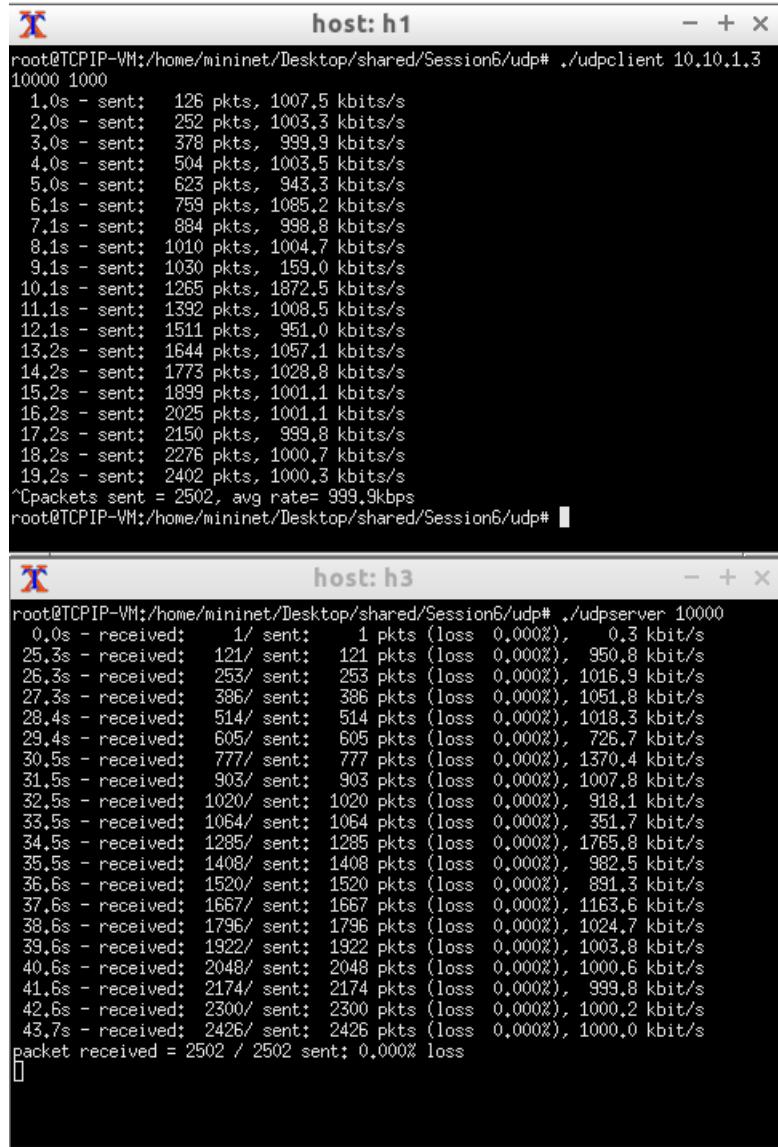
```
host: h1
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpclient 10.10.1.3 10000 100
1.0s - sent: 14 pkts, 107.7 kbytes/s
2.1s - sent: 27 pkts, 100.0 kbytes/s
3.1s - sent: 40 pkts, 100.0 kbytes/s
4.2s - sent: 53 pkts, 100.1 kbytes/s
5.2s - sent: 66 pkts, 100.0 kbytes/s
6.2s - sent: 79 pkts, 100.0 kbytes/s
7.3s - sent: 92 pkts, 100.1 kbytes/s
8.3s - sent: 105 pkts, 99.5 kbytes/s
9.4s - sent: 118 pkts, 100.7 kbytes/s
10.4s - sent: 131 pkts, 100.0 kbytes/s
11.4s - sent: 144 pkts, 99.7 kbytes/s
12.5s - sent: 157 pkts, 100.8 kbytes/s
13.5s - sent: 170 pkts, 100.0 kbytes/s
14.6s - sent: 183 pkts, 99.3 kbytes/s
15.6s - sent: 196 pkts, 101.6 kbytes/s
16.6s - sent: 209 pkts, 100.1 kbytes/s
17.7s - sent: 222 pkts, 100.0 kbytes/s
18.7s - sent: 235 pkts, 100.1 kbytes/s
19.8s - sent: 248 pkts, 100.1 kbytes/s
20.8s - sent: 261 pkts, 100.1 kbytes/s
^Cpackets sent = 266, avg rate= 100.0 kbytes/s
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# 

host: h3
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6# cd udp/
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpserver 10000
0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 2.4 kbit/s
3.4s - received: 14/ sent: 14 pkts (loss 0.000%), 100.5 kbit/s
4.4s - received: 27/ sent: 27 pkts (loss 0.000%), 100.0 kbit/s
5.4s - received: 40/ sent: 40 pkts (loss 0.000%), 100.0 kbit/s
6.5s - received: 53/ sent: 53 pkts (loss 0.000%), 100.1 kbit/s
7.5s - received: 66/ sent: 66 pkts (loss 0.000%), 100.2 kbit/s
8.5s - received: 79/ sent: 79 pkts (loss 0.000%), 100.0 kbit/s
9.6s - received: 92/ sent: 92 pkts (loss 0.000%), 100.1 kbit/s
10.6s - received: 105/ sent: 105 pkts (loss 0.000%), 99.9 kbit/s
11.7s - received: 118/ sent: 118 pkts (loss 0.000%), 100.3 kbit/s
12.7s - received: 131/ sent: 131 pkts (loss 0.000%), 100.0 kbit/s
13.7s - received: 144/ sent: 144 pkts (loss 0.000%), 100.0 kbit/s
14.8s - received: 157/ sent: 157 pkts (loss 0.000%), 100.3 kbit/s
15.8s - received: 170/ sent: 170 pkts (loss 0.000%), 100.0 kbit/s
16.9s - received: 183/ sent: 183 pkts (loss 0.000%), 99.9 kbit/s
17.9s - received: 196/ sent: 196 pkts (loss 0.000%), 100.5 kbit/s
18.9s - received: 209/ sent: 209 pkts (loss 0.000%), 100.0 kbit/s
20.0s - received: 222/ sent: 222 pkts (loss 0.000%), 100.0 kbit/s
21.0s - received: 235/ sent: 235 pkts (loss 0.000%), 100.0 kbit/s
22.1s - received: 248/ sent: 248 pkts (loss 0.000%), 100.0 kbit/s
23.1s - received: 261/ sent: 261 pkts (loss 0.000%), 100.1 kbit/s
packet received = 266 / 266 sent: 0.000% loss
^C
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# 
```

در هر دو مرتبه تکرار آزمایش مشاهده شد که مقدار loss در h3 صفر بوده و مقدار goodput در بسیاری از موارد طبق مقدار مشخص شده (یعنی همان 100 kbps) می باشد.

سؤال 2: آزمایش را با نرخ های 1 Gbps، 100 Mbps، 10 Mbps و 1 Mbps تکرار کنید. مقدار goodput و احتمال های loss چقدر است؟ به ازای چه نرخی، احتمال loss بالاتر از 1% است؟ آیا می توانید نتایج را با توجه به گذشته در فایل py5.lab توجیه نمایید؟

• نرخ 1 Mbps



host: h1

```
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpclient 10.10.1.3
10000 1000
 1.0s - sent: 126 pkts, 1007.5 kbytes/s
 2.0s - sent: 252 pkts, 1003.3 kbytes/s
 3.0s - sent: 378 pkts, 999.9 kbytes/s
 4.0s - sent: 504 pkts, 1003.5 kbytes/s
 5.0s - sent: 623 pkts, 943.3 kbytes/s
 6.1s - sent: 759 pkts, 1085.2 kbytes/s
 7.1s - sent: 884 pkts, 998.8 kbytes/s
 8.1s - sent: 1010 pkts, 1004.7 kbytes/s
 9.1s - sent: 1030 pkts, 159.0 kbytes/s
10.1s - sent: 1265 pkts, 1872.5 kbytes/s
11.1s - sent: 1392 pkts, 1008.5 kbytes/s
12.1s - sent: 1511 pkts, 951.0 kbytes/s
13.2s - sent: 1644 pkts, 1057.1 kbytes/s
14.2s - sent: 1773 pkts, 1028.8 kbytes/s
15.2s - sent: 1899 pkts, 1001.1 kbytes/s
16.2s - sent: 2025 pkts, 1001.1 kbytes/s
17.2s - sent: 2150 pkts, 999.8 kbytes/s
18.2s - sent: 2276 pkts, 1000.7 kbytes/s
19.2s - sent: 2402 pkts, 1000.3 kbytes/s
^Cpackets sent = 2502, avg rate= 999.9kbps
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/udp#
```

host: h3

```
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpserver 10000
 0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 0.3 kbit/s
25.3s - received: 121/ sent: 121 pkts (loss 0.000%), 950.8 kbit/s
26.3s - received: 253/ sent: 253 pkts (loss 0.000%), 1016.9 kbit/s
27.3s - received: 386/ sent: 386 pkts (loss 0.000%), 1051.8 kbit/s
28.4s - received: 514/ sent: 514 pkts (loss 0.000%), 1018.3 kbit/s
29.4s - received: 605/ sent: 605 pkts (loss 0.000%), 726.7 kbit/s
30.5s - received: 777/ sent: 777 pkts (loss 0.000%), 1370.4 kbit/s
31.5s - received: 903/ sent: 903 pkts (loss 0.000%), 1007.8 kbit/s
32.5s - received: 1020/ sent: 1020 pkts (loss 0.000%), 918.1 kbit/s
33.5s - received: 1064/ sent: 1064 pkts (loss 0.000%), 351.7 kbit/s
34.5s - received: 1285/ sent: 1285 pkts (loss 0.000%), 1765.8 kbit/s
35.5s - received: 1408/ sent: 1408 pkts (loss 0.000%), 982.5 kbit/s
36.6s - received: 1520/ sent: 1520 pkts (loss 0.000%), 891.3 kbit/s
37.6s - received: 1667/ sent: 1667 pkts (loss 0.000%), 1163.6 kbit/s
38.6s - received: 1796/ sent: 1796 pkts (loss 0.000%), 1024.7 kbit/s
39.6s - received: 1922/ sent: 1922 pkts (loss 0.000%), 1003.8 kbit/s
40.6s - received: 2048/ sent: 2048 pkts (loss 0.000%), 1000.6 kbit/s
41.6s - received: 2174/ sent: 2174 pkts (loss 0.000%), 999.8 kbit/s
42.6s - received: 2300/ sent: 2300 pkts (loss 0.000%), 1000.2 kbit/s
43.7s - received: 2426/ sent: 2426 pkts (loss 0.000%), 1000.0 kbit/s
packet received = 2502 / 2502 sent: 0.000% loss
```

نرخ 10 Mbps •

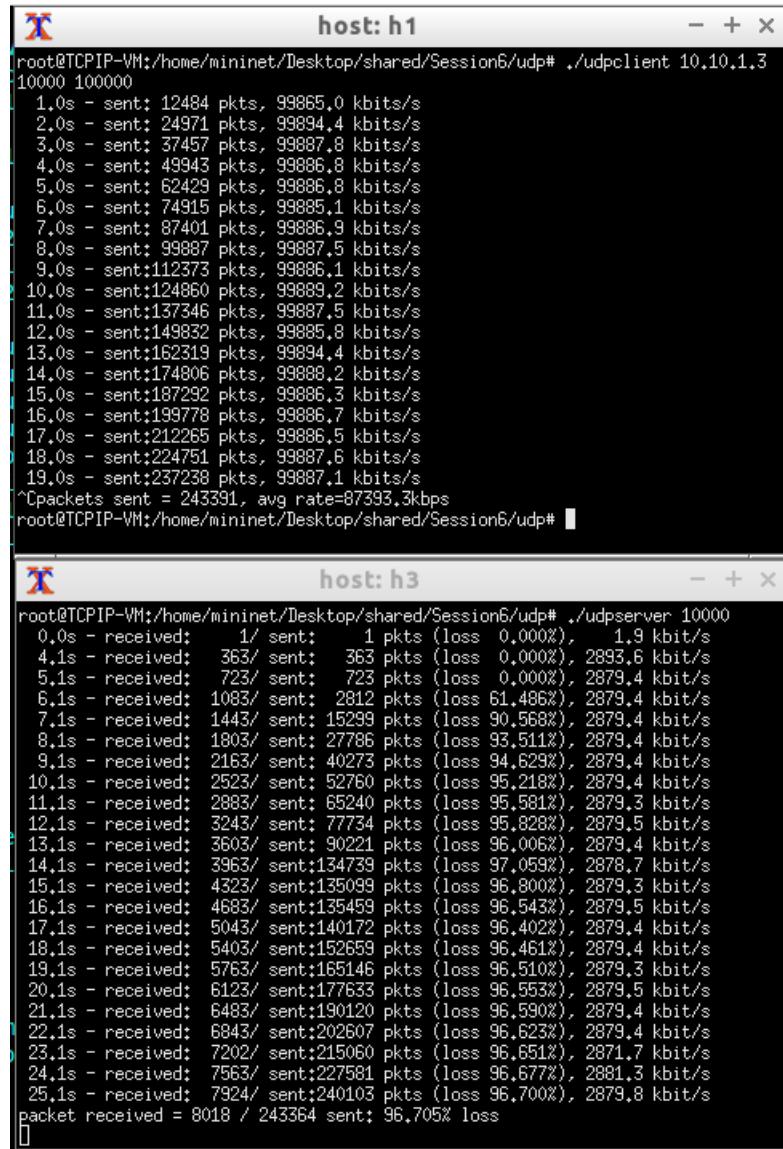
X host: h1

```
root@TCPiP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpclient 10.10.1.3
10000 10000
1.0s - sent: 1252 pkts, 10007.4 kbytes/s
2.0s - sent: 2503 pkts, 10004.8 kbytes/s
3.0s - sent: 3754 pkts, 10003.7 kbytes/s
4.0s - sent: 5005 pkts, 10003.6 kbytes/s
5.0s - sent: 6256 pkts, 10003.9 kbytes/s
6.0s - sent: 7507 pkts, 10004.1 kbytes/s
7.0s - sent: 8758 pkts, 10005.7 kbytes/s
8.0s - sent: 10009 pkts, 10003.1 kbytes/s
9.0s - sent: 11260 pkts, 10002.1 kbytes/s
10.0s - sent: 12512 pkts, 10009.8 kbytes/s
11.0s - sent: 13763 pkts, 10004.1 kbytes/s
12.0s - sent: 15014 pkts, 10003.0 kbytes/s
13.0s - sent: 16266 pkts, 10013.4 kbytes/s
14.0s - sent: 17517 pkts, 10003.1 kbytes/s
15.0s - sent: 18768 pkts, 10003.8 kbytes/s
16.0s - sent: 20019 pkts, 10003.8 kbytes/s
17.0s - sent: 21270 pkts, 10000.9 kbytes/s
18.0s - sent: 22521 pkts, 10008.0 kbytes/s
19.0s - sent: 23772 pkts, 10003.5 kbytes/s
^Cpackets sent = 24522, avg rate=8759.7kbps
root@TCPiP-VM:/home/mininet/Desktop/shared/Session6/udp#
```

X host: h3

```
root@TCPiP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpserver 10000
0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 0.9 kbit/s
8.4s - received: 363/ sent: 363 pkts (loss 0.000%), 2893.5 kbit/s
9.4s - received: 723/ sent: 723 pkts (loss 0.000%), 2879.3 kbit/s
10.4s - received: 1083/ sent: 1083 pkts (loss 0.000%), 2879.5 kbit/s
11.4s - received: 1443/ sent: 1532 pkts (loss 5.809%), 2879.4 kbit/s
12.4s - received: 1803/ sent: 2783 pkts (loss 35.214%), 2879.0 kbit/s
13.4s - received: 2163/ sent: 4034 pkts (loss 46.381%), 2879.7 kbit/s
14.4s - received: 2523/ sent: 5284 pkts (loss 52.252%), 2879.6 kbit/s
15.4s - received: 2883/ sent: 6539 pkts (loss 55.884%), 2879.6 kbit/s
16.4s - received: 3240/ sent: 7778 pkts (loss 58.344%), 2848.3 kbit/s
17.4s - received: 3601/ sent: 9032 pkts (loss 60.131%), 2880.3 kbit/s
18.4s - received: 3961/ sent: 11048 pkts (loss 64.147%), 2879.3 kbit/s
19.4s - received: 4320/ sent: 11540 pkts (loss 62.565%), 2867.4 kbit/s
20.4s - received: 4681/ sent: 12794 pkts (loss 63.413%), 2881.5 kbit/s
21.4s - received: 5041/ sent: 14045 pkts (loss 64.108%), 2879.4 kbit/s
22.4s - received: 5402/ sent: 15304 pkts (loss 64.702%), 2880.2 kbit/s
23.4s - received: 5762/ sent: 16554 pkts (loss 65.193%), 2879.5 kbit/s
24.4s - received: 6122/ sent: 17805 pkts (loss 65.616%), 2879.4 kbit/s
25.4s - received: 6482/ sent: 19056 pkts (loss 65.984%), 2879.1 kbit/s
26.4s - received: 6843/ sent: 20310 pkts (loss 66.307%), 2880.4 kbit/s
27.4s - received: 7203/ sent: 21560 pkts (loss 66.591%), 2879.4 kbit/s
28.4s - received: 7563/ sent: 22811 pkts (loss 66.845%), 2879.5 kbit/s
29.4s - received: 7924/ sent: 24065 pkts (loss 67.073%), 2878.0 kbit/s
packet received = 8055 / 24520 sent: 67.149% loss
```

نرخ 100 Mbps •

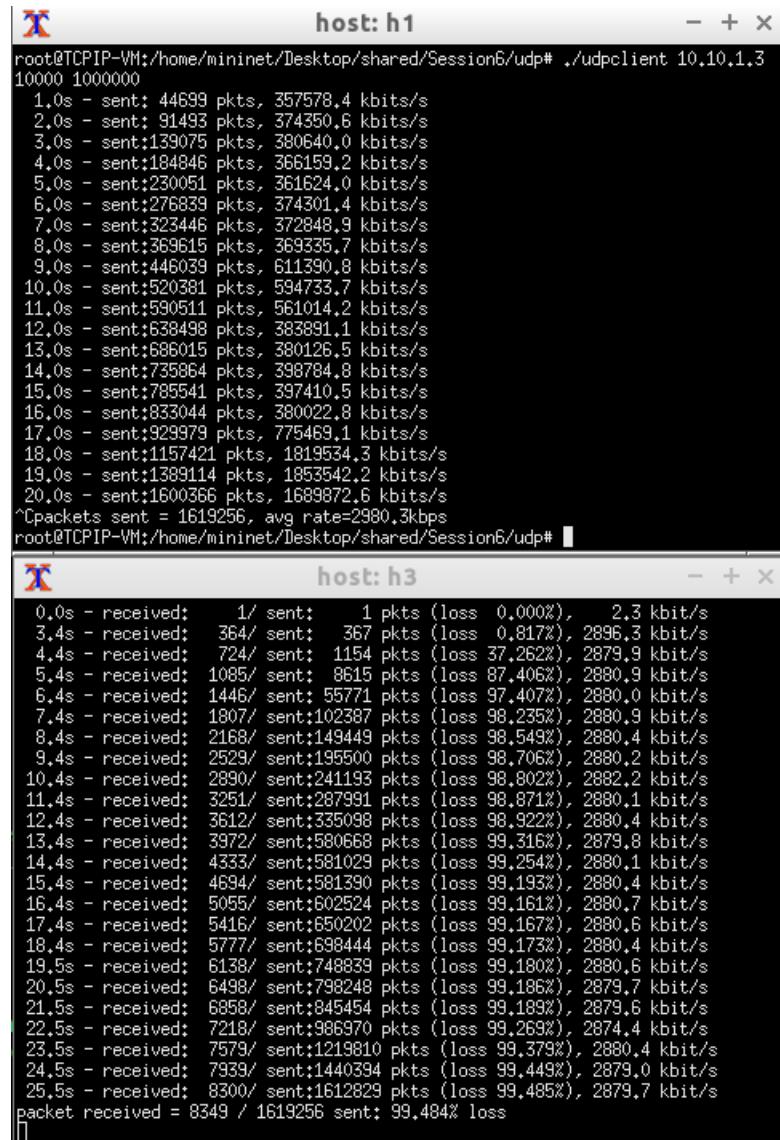


```
host: h1
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpclient 10.10.1.3
10000 100000
 1.0s - sent: 12484 pkts, 99865.0 kbit/s
 2.0s - sent: 24971 pkts, 99894.4 kbit/s
 3.0s - sent: 37457 pkts, 99887.8 kbit/s
 4.0s - sent: 49943 pkts, 99886.8 kbit/s
 5.0s - sent: 62429 pkts, 99886.8 kbit/s
 6.0s - sent: 74915 pkts, 99885.1 kbit/s
 7.0s - sent: 87401 pkts, 99886.9 kbit/s
 8.0s - sent: 99887 pkts, 99887.5 kbit/s
 9.0s - sent:112373 pkts, 99886.1 kbit/s
10.0s - sent:1124860 pkts, 99889.2 kbit/s
11.0s - sent:1137346 pkts, 99887.5 kbit/s
12.0s - sent:1149832 pkts, 99885.8 kbit/s
13.0s - sent:1162319 pkts, 99894.4 kbit/s
14.0s - sent:1174806 pkts, 99888.2 kbit/s
15.0s - sent:1187292 pkts, 99886.3 kbit/s
16.0s - sent:1199778 pkts, 99886.7 kbit/s
17.0s - sent:1212265 pkts, 99886.5 kbit/s
18.0s - sent:1224751 pkts, 99887.6 kbit/s
19.0s - sent:1237238 pkts, 99887.1 kbit/s
^Cpackets sent = 243391, avg rate=87393.3kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp#
```

```
host: h3
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpserver 10000
0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 1.9 kbit/s
4.1s - received: 363/ sent: 363 pkts (loss 0.000%), 2893.6 kbit/s
5.1s - received: 723/ sent: 723 pkts (loss 0.000%), 2879.4 kbit/s
6.1s - received: 1083/ sent: 2812 pkts (loss 61.486%), 2879.4 kbit/s
7.1s - received: 1443/ sent: 15299 pkts (loss 90.568%), 2879.4 kbit/s
8.1s - received: 1803/ sent: 27786 pkts (loss 93.511%), 2879.4 kbit/s
9.1s - received: 2163/ sent: 40273 pkts (loss 94.629%), 2879.4 kbit/s
10.1s - received: 2523/ sent: 52760 pkts (loss 95.218%), 2879.4 kbit/s
11.1s - received: 2883/ sent: 65240 pkts (loss 95.581%), 2879.3 kbit/s
12.1s - received: 3243/ sent: 77734 pkts (loss 95.828%), 2879.5 kbit/s
13.1s - received: 3603/ sent: 90221 pkts (loss 96.006%), 2879.4 kbit/s
14.1s - received: 3963/ sent:134739 pkts (loss 97.059%), 2878.7 kbit/s
15.1s - received: 4323/ sent:135099 pkts (loss 96.800%), 2879.3 kbit/s
16.1s - received: 4683/ sent:135459 pkts (loss 96.543%), 2879.5 kbit/s
17.1s - received: 5043/ sent:140172 pkts (loss 96.402%), 2879.4 kbit/s
18.1s - received: 5403/ sent:152659 pkts (loss 96.461%), 2879.4 kbit/s
19.1s - received: 5763/ sent:165146 pkts (loss 96.510%), 2879.3 kbit/s
20.1s - received: 6123/ sent:177633 pkts (loss 96.553%), 2879.5 kbit/s
21.1s - received: 6483/ sent:190120 pkts (loss 96.590%), 2879.4 kbit/s
22.1s - received: 6843/ sent:202607 pkts (loss 96.623%), 2879.4 kbit/s
23.1s - received: 7202/ sent:215060 pkts (loss 96.651%), 2871.7 kbit/s
24.1s - received: 7563/ sent:227581 pkts (loss 96.677%), 2881.3 kbit/s
25.1s - received: 7924/ sent:240103 pkts (loss 96.700%), 2879.8 kbit/s
packet received = 8018 / 243364 sent: 96.705% loss
```

• نرخ 1 Gbps :



```
host: h1
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpclient 10.10.1.3
10000 1000000
1.0s - sent: 44699 pkts, 357578.4 kbytes/s
2.0s - sent: 91493 pkts, 374350.6 kbytes/s
3.0s - sent:139075 pkts, 380640.0 kbytes/s
4.0s - sent:184846 pkts, 366159.2 kbytes/s
5.0s - sent:230051 pkts, 361624.0 kbytes/s
6.0s - sent:276839 pkts, 374301.4 kbytes/s
7.0s - sent:323446 pkts, 372848.9 kbytes/s
8.0s - sent:369615 pkts, 369335.7 kbytes/s
9.0s - sent:446039 pkts, 611390.8 kbytes/s
10.0s - sent:520381 pkts, 594733.7 kbytes/s
11.0s - sent:590511 pkts, 561014.2 kbytes/s
12.0s - sent:638498 pkts, 383891.1 kbytes/s
13.0s - sent:686015 pkts, 380126.5 kbytes/s
14.0s - sent:735864 pkts, 398784.8 kbytes/s
15.0s - sent:785541 pkts, 397410.5 kbytes/s
16.0s - sent:833044 pkts, 380022.8 kbytes/s
17.0s - sent:929979 pkts, 775469.1 kbytes/s
18.0s - sent:1157421 pkts, 1819534.3 kbytes/s
19.0s - sent:1389114 pkts, 1853542.2 kbytes/s
20.0s - sent:1600366 pkts, 1689872.6 kbytes/s
^Cpackets sent = 1619256, avg rate=2980.3kbytes/s
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp#
```

```
host: h3
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpreceiver 10.10.1.3
0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 2.3 kbit/s
3.4s - received: 364/ sent: 367 pkts (loss 0.817%), 2896.3 kbit/s
4.4s - received: 724/ sent: 1154 pkts (loss 37.262%), 2879.9 kbit/s
5.4s - received: 1085/ sent: 8615 pkts (loss 87.406%), 2880.9 kbit/s
6.4s - received: 1446/ sent: 55771 pkts (loss 97.407%), 2880.0 kbit/s
7.4s - received: 1807/ sent:102387 pkts (loss 98.235%), 2880.9 kbit/s
8.4s - received: 2168/ sent:149449 pkts (loss 98.549%), 2880.4 kbit/s
9.4s - received: 2529/ sent:195500 pkts (loss 98.706%), 2880.2 kbit/s
10.4s - received: 2890/ sent:241193 pkts (loss 98.802%), 2882.2 kbit/s
11.4s - received: 3251/ sent:287991 pkts (loss 98.871%), 2880.1 kbit/s
12.4s - received: 3612/ sent:335098 pkts (loss 98.922%), 2880.4 kbit/s
13.4s - received: 3972/ sent:580668 pkts (loss 99.316%), 2879.8 kbit/s
14.4s - received: 4333/ sent:581029 pkts (loss 99.284%), 2880.1 kbit/s
15.4s - received: 4694/ sent:581390 pkts (loss 99.193%), 2880.4 kbit/s
16.4s - received: 5055/ sent:602524 pkts (loss 99.161%), 2880.7 kbit/s
17.4s - received: 5416/ sent:650202 pkts (loss 99.167%), 2880.6 kbit/s
18.4s - received: 5777/ sent:698444 pkts (loss 99.173%), 2880.4 kbit/s
19.5s - received: 6138/ sent:748839 pkts (loss 99.180%), 2880.6 kbit/s
20.5s - received: 6498/ sent:798248 pkts (loss 99.186%), 2879.7 kbit/s
21.5s - received: 6858/ sent:845454 pkts (loss 99.189%), 2879.6 kbit/s
22.5s - received: 7218/ sent:896970 pkts (loss 99.269%), 2874.4 kbit/s
23.5s - received: 7579/ sent:1219810 pkts (loss 99.379%), 2880.4 kbit/s
24.5s - received: 7939/ sent:1440394 pkts (loss 99.449%), 2879.0 kbit/s
25.5s - received: 8300/ sent:1612829 pkts (loss 99.485%), 2879.7 kbit/s
packet received = 8349 / 1619256 sent: 99.484% loss
```

همانطور که در تصاویر دیده می شود مشاهده می کنیم که در این پروتکل به ازای نرخ 10Mbps loss جهش زیادی پیدا می کند و طبق فایل topo.py به نظر می رسد که این افزایش مقدار loss به خاطر پر شدن ظرفیت لینک بین r1 و sw2 است. با افزایش نرخ ارسال، میزان ترافیک در این لینک نیز افزایش می یابد و از ظرفیت آن فراتر می رود. این موضوع باعث دراپ شدن بسته می شود، زیرا روتر قادر به پردازش تمام بسته های ورودی نیست.

برای رفع این مشکل، می توان به عنوان مثال، پهنه ای باند لینک r1 و sw2 را افزایش داد که به آن اجازه می دهد تا

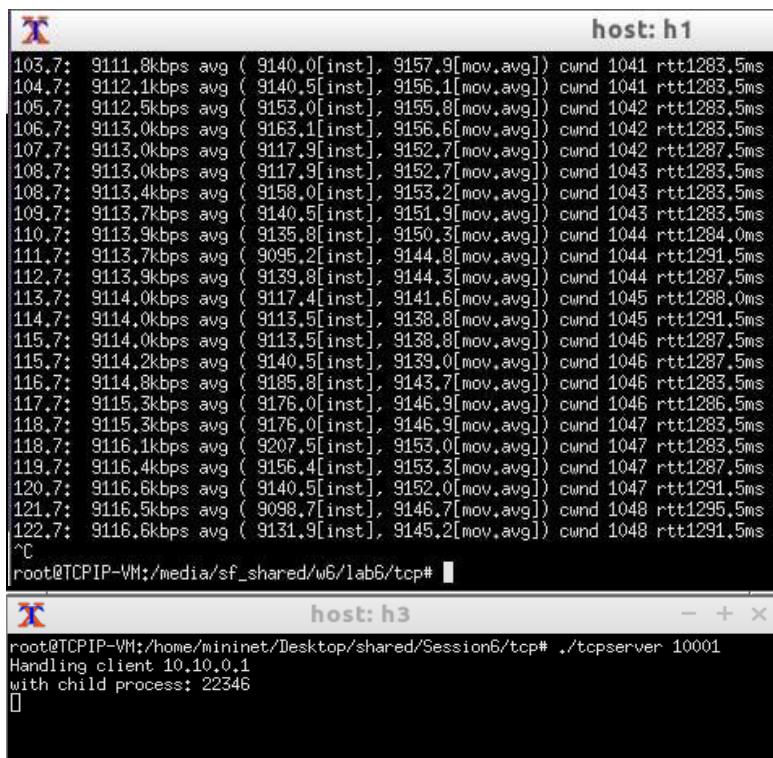
ترافیک بیشتری را مدیریت کند و اتلاف بسته را کاهش دهد. برای این کار می توان در فایل topo.py یک متغیر به اسم res ایجاد کرد و برای آن مقدار bw را set کرد و تا این مشکل کاهش پیدا کند :

```
res = net.addLink( sw2, r1, intfName2='r1-eth1' )
```

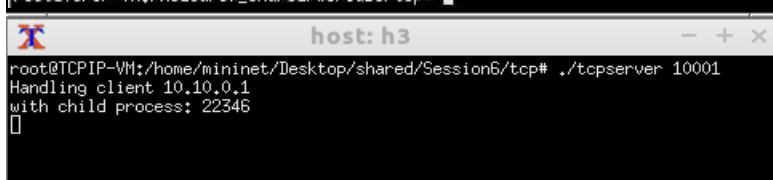
```
info( '*** Starting network\n' )
res.intf1.config(bw=30)
```

سؤال 3: یک کلاینت TCP روی ماشین h1 اجرا نمایید که برای سرور h3 داده ارسال می کند. Goodput ارتباط چقدر است؟

نتایج انجام آزمایش برای بار اول:

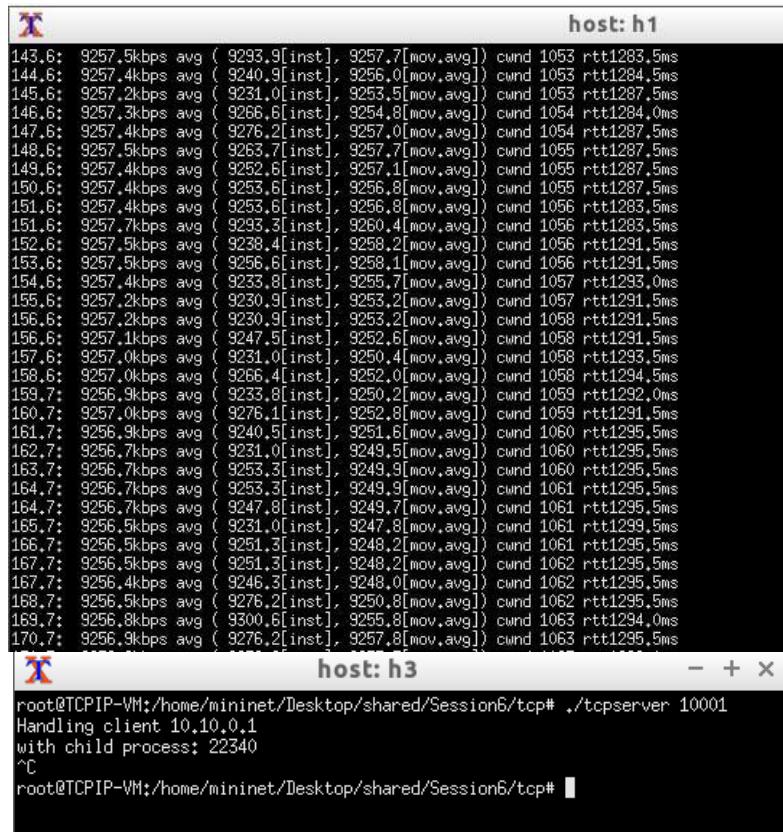


```
host: h1
103.7: 9111.8kbps avg ( 9140.0[inst], 9157.9[mov,avg]) cwnd 1041 rtt1283.5ms
104.7: 9112.1kbps avg ( 9140.5[inst], 9156.1[mov,avg]) cwnd 1041 rtt1283.5ms
105.7: 9112.5kbps avg ( 9153.0[inst], 9155.8[mov,avg]) cwnd 1042 rtt1283.5ms
106.7: 9113.0kbps avg ( 9163.1[inst], 9156.6[mov,avg]) cwnd 1042 rtt1283.5ms
107.7: 9113.0kbps avg ( 9117.9[inst], 9152.7[mov,avg]) cwnd 1042 rtt1287.5ms
108.7: 9113.0kbps avg ( 9117.9[inst], 9152.7[mov,avg]) cwnd 1043 rtt1283.5ms
108.7: 9113.4kbps avg ( 9158.0[inst], 9153.2[mov,avg]) cwnd 1043 rtt1283.5ms
109.7: 9113.7kbps avg ( 9140.5[inst], 9151.9[mov,avg]) cwnd 1043 rtt1283.5ms
110.7: 9113.9kbps avg ( 9135.8[inst], 9150.3[mov,avg]) cwnd 1044 rtt1284.0ms
111.7: 9113.7kbps avg ( 9095.2[inst], 9144.8[mov,avg]) cwnd 1044 rtt1291.5ms
112.7: 9113.9kbps avg ( 9139.8[inst], 9144.3[mov,avg]) cwnd 1044 rtt1287.5ms
113.7: 9114.0kbps avg ( 9117.4[inst], 9141.6[mov,avg]) cwnd 1045 rtt1288.0ms
114.7: 9114.0kbps avg ( 9113.5[inst], 9138.8[mov,avg]) cwnd 1045 rtt1291.5ms
115.7: 9114.0kbps avg ( 9113.5[inst], 9138.8[mov,avg]) cwnd 1046 rtt1287.5ms
115.7: 9114.2kbps avg ( 9140.5[inst], 9139.0[mov,avg]) cwnd 1046 rtt1287.5ms
116.7: 9114.8kbps avg ( 9185.8[inst], 9143.7[mov,avg]) cwnd 1046 rtt1283.5ms
117.7: 9115.3kbps avg ( 9176.0[inst], 9146.9[mov,avg]) cwnd 1046 rtt1286.5ms
118.7: 9115.3kbps avg ( 9176.0[inst], 9146.9[mov,avg]) cwnd 1047 rtt1283.5ms
118.7: 9116.1kbps avg ( 9207.5[inst], 9153.0[mov,avg]) cwnd 1047 rtt1283.5ms
119.7: 9116.4kbps avg ( 9156.4[inst], 9153.3[mov,avg]) cwnd 1047 rtt1287.5ms
120.7: 9116.6kbps avg ( 9140.5[inst], 9152.0[mov,avg]) cwnd 1047 rtt1291.5ms
121.7: 9116.5kbps avg ( 9098.7[inst], 9146.7[mov,avg]) cwnd 1048 rtt1295.5ms
122.7: 9116.6kbps avg ( 9131.9[inst], 9145.2[mov,avg]) cwnd 1048 rtt1291.5ms
^C
root@TCPPIP-VM:/media/sf_shared/w6/lab6/tcp#
```



```
host: h3
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/tcp# ./tcpserver 10001
Handling client 10.10.0.1
with child process: 22346
[]
```

نتایج انجام آزمایش برای بار دوم •



host: h1

```

143.6: 9257.5kbps avg ( 9233.9[inst], 9257.7[mov.avg]) cwnd 1053 rtt1283.5ms
144.6: 9257.4kbps avg ( 9240.9[inst], 9256.0[mov.avg]) cwnd 1053 rtt1284.5ms
145.6: 9257.2kbps avg ( 9231.0[inst], 9253.5[mov.avg]) cwnd 1053 rtt1287.5ms
146.6: 9257.3kbps avg ( 9266.6[inst], 9254.8[mov.avg]) cwnd 1054 rtt1284.0ms
147.6: 9257.4kbps avg ( 9276.2[inst], 9257.0[mov.avg]) cwnd 1054 rtt1287.5ms
148.6: 9257.5kbps avg ( 9263.7[inst], 9257.7[mov.avg]) cwnd 1055 rtt1287.5ms
149.6: 9257.4kbps avg ( 9252.6[inst], 9257.1[mov.avg]) cwnd 1055 rtt1287.0ms
150.6: 9257.4kbps avg ( 9253.6[inst], 9256.8[mov.avg]) cwnd 1055 rtt1287.5ms
151.6: 9257.4kbps avg ( 9253.6[inst], 9256.8[mov.avg]) cwnd 1056 rtt1283.5ms
151.6: 9257.7kbps avg ( 9293.3[inst], 9260.4[mov.avg]) cwnd 1056 rtt1283.5ms
152.6: 9257.5kbps avg ( 9238.4[inst], 9258.2[mov.avg]) cwnd 1056 rtt1291.5ms
153.6: 9257.5kbps avg ( 9256.6[inst], 9258.1[mov.avg]) cwnd 1056 rtt1291.5ms
154.6: 9257.4kbps avg ( 9233.8[inst], 9255.7[mov.avg]) cwnd 1057 rtt1293.0ms
155.6: 9257.2kbps avg ( 9230.9[inst], 9253.2[mov.avg]) cwnd 1057 rtt1291.5ms
156.6: 9257.2kbps avg ( 9230.9[inst], 9253.2[mov.avg]) cwnd 1058 rtt1291.5ms
156.6: 9257.1kbps avg ( 9247.5[inst], 9252.6[mov.avg]) cwnd 1058 rtt1291.5ms
157.6: 9257.0kbps avg ( 9231.0[inst], 9250.4[mov.avg]) cwnd 1058 rtt1293.5ms
158.6: 9257.0kbps avg ( 9266.4[inst], 9252.0[mov.avg]) cwnd 1058 rtt1294.5ms
159.7: 9256.9kbps avg ( 9233.8[inst], 9250.2[mov.avg]) cwnd 1059 rtt1292.0ms
160.7: 9257.0kbps avg ( 9276.1[inst], 9252.8[mov.avg]) cwnd 1059 rtt1291.5ms
161.7: 9256.9kbps avg ( 9240.5[inst], 9251.6[mov.avg]) cwnd 1060 rtt1295.5ms
162.7: 9256.7kbps avg ( 9231.0[inst], 9249.5[mov.avg]) cwnd 1060 rtt1295.5ms
163.7: 9256.7kbps avg ( 9253.3[inst], 9249.9[mov.avg]) cwnd 1060 rtt1295.5ms
164.7: 9256.7kbps avg ( 9253.3[inst], 9249.9[mov.avg]) cwnd 1061 rtt1295.5ms
164.7: 9256.7kbps avg ( 9247.8[inst], 9249.7[mov.avg]) cwnd 1061 rtt1295.5ms
165.7: 9256.5kbps avg ( 9231.0[inst], 9247.8[mov.avg]) cwnd 1061 rtt1299.5ms
166.7: 9256.5kbps avg ( 9251.3[inst], 9248.2[mov.avg]) cwnd 1061 rtt1295.5ms
167.7: 9256.5kbps avg ( 9251.3[inst], 9248.2[mov.avg]) cwnd 1062 rtt1295.5ms
167.7: 9256.4kbps avg ( 9246.3[inst], 9248.0[mov.avg]) cwnd 1062 rtt1295.5ms
168.7: 9256.5kbps avg ( 9276.2[inst], 9250.8[mov.avg]) cwnd 1062 rtt1295.5ms
169.7: 9256.8kbps avg ( 9300.6[inst], 9255.8[mov.avg]) cwnd 1063 rtt1294.0ms
170.7: 9256.9kbps avg ( 9276.2[inst], 9257.8[mov.avg]) cwnd 1063 rtt1295.5ms

```

host: h3

```

root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/tcp# ./tcpserver 10001
Handling client 10.10.0.1
with child process: 22340
^C
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/tcp# ■

```

همانطور که در تصویر مشخص می باشد در آزمایش مرتبه اول بعد از گذشت تقریبا 122 ثانیه مقدار goodput برای TCP به 9116 kbps می رسد و این آزمایش را برای بار دوم نیز تکرار می کنیم که بعد از گذشت تقریبا 150 ثانیه مقدار goodput برای TCP برابر است با 9257 kbps. می توان از میانگین گرفتن این دو عدد مقدار goodput را به صورت نظری بدست آورد.

## گزارشکار آزمایش ۶: کنترل ازدحام TCP و UDP – قسمت دوم

سیده شکیبا انارکی فیروز – ۹۹۴۴۲۰۴۷

بهاره کاوی نژاد – ۹۹۴۳۱۲۱۷

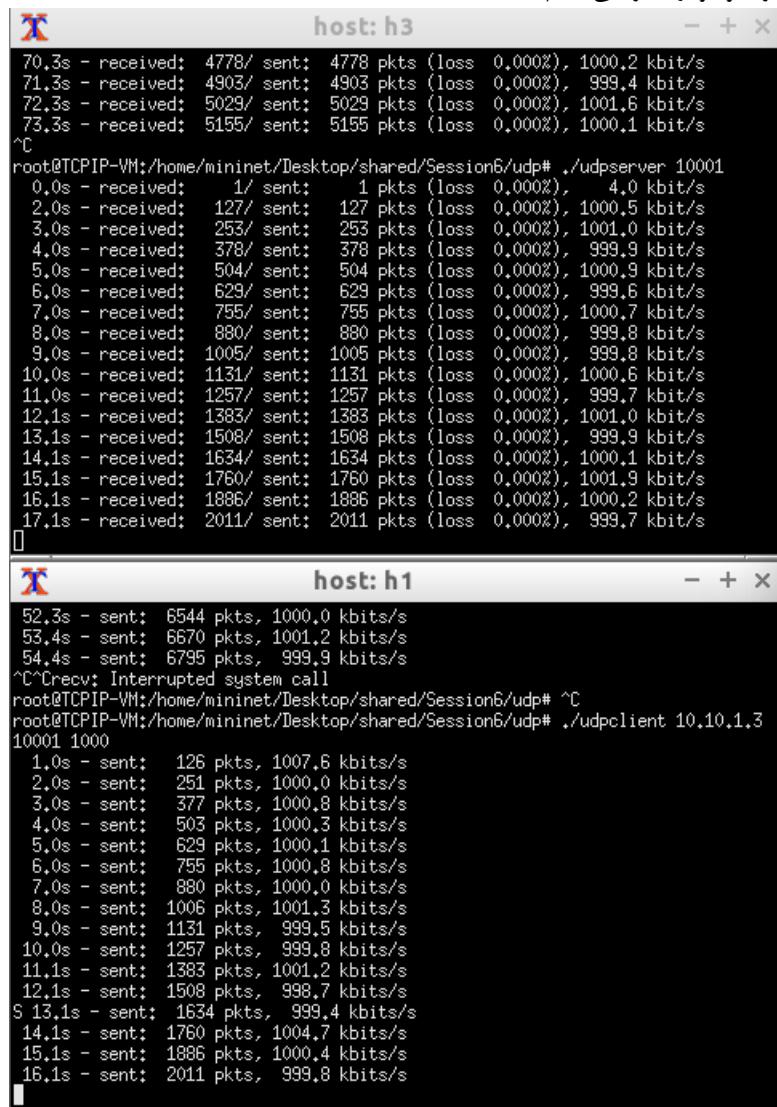
ب-۲) محدودسازی پهنای باند روتر

پهنای باند ایترفیس eth1 از روتر را به ۳ Mbps محدود سازید.

با استفاده از دستورات زیر در فایل پایتون پهنای باند ایترفیس eth1 از روتر را به ۳ Mbps محدود می کنیم.

```
res = net.addLink(sw2, r1, intfName2='r1-eth1')
info('*** Starting network\n')
res.intf1.config(bw=3)
```

سپس تپولوژی را clean کرده و دوباره اجرا می کیم.



```
host: h3
70.3s - received: 4778/ sent: 4778 pkts (loss 0.000%), 1000.2 kbit/s
71.3s - received: 4903/ sent: 4903 pkts (loss 0.000%), 999.4 kbit/s
72.3s - received: 5029/ sent: 5029 pkts (loss 0.000%), 1001.6 kbit/s
73.3s - received: 5155/ sent: 5155 pkts (loss 0.000%), 1000.1 kbit/s
^C
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpserver 10001
 0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 4.0 kbit/s
 2.0s - received: 127/ sent: 127 pkts (loss 0.000%), 1000.5 kbit/s
 3.0s - received: 253/ sent: 253 pkts (loss 0.000%), 1001.0 kbit/s
 4.0s - received: 378/ sent: 378 pkts (loss 0.000%), 999.9 kbit/s
 5.0s - received: 504/ sent: 504 pkts (loss 0.000%), 1000.9 kbit/s
 6.0s - received: 629/ sent: 629 pkts (loss 0.000%), 999.6 kbit/s
 7.0s - received: 755/ sent: 755 pkts (loss 0.000%), 1000.7 kbit/s
 8.0s - received: 880/ sent: 880 pkts (loss 0.000%), 999.8 kbit/s
 9.0s - received: 1005/ sent: 1005 pkts (loss 0.000%), 999.8 kbit/s
10.0s - received: 1131/ sent: 1131 pkts (loss 0.000%), 1000.6 kbit/s
11.0s - received: 1257/ sent: 1257 pkts (loss 0.000%), 999.7 kbit/s
12.1s - received: 1383/ sent: 1383 pkts (loss 0.000%), 1001.0 kbit/s
13.1s - received: 1508/ sent: 1508 pkts (loss 0.000%), 999.9 kbit/s
14.1s - received: 1634/ sent: 1634 pkts (loss 0.000%), 1000.1 kbit/s
15.1s - received: 1760/ sent: 1760 pkts (loss 0.000%), 1001.9 kbit/s
16.1s - received: 1886/ sent: 1886 pkts (loss 0.000%), 1000.2 kbit/s
17.1s - received: 2011/ sent: 2011 pkts (loss 0.000%), 999.7 kbit/s

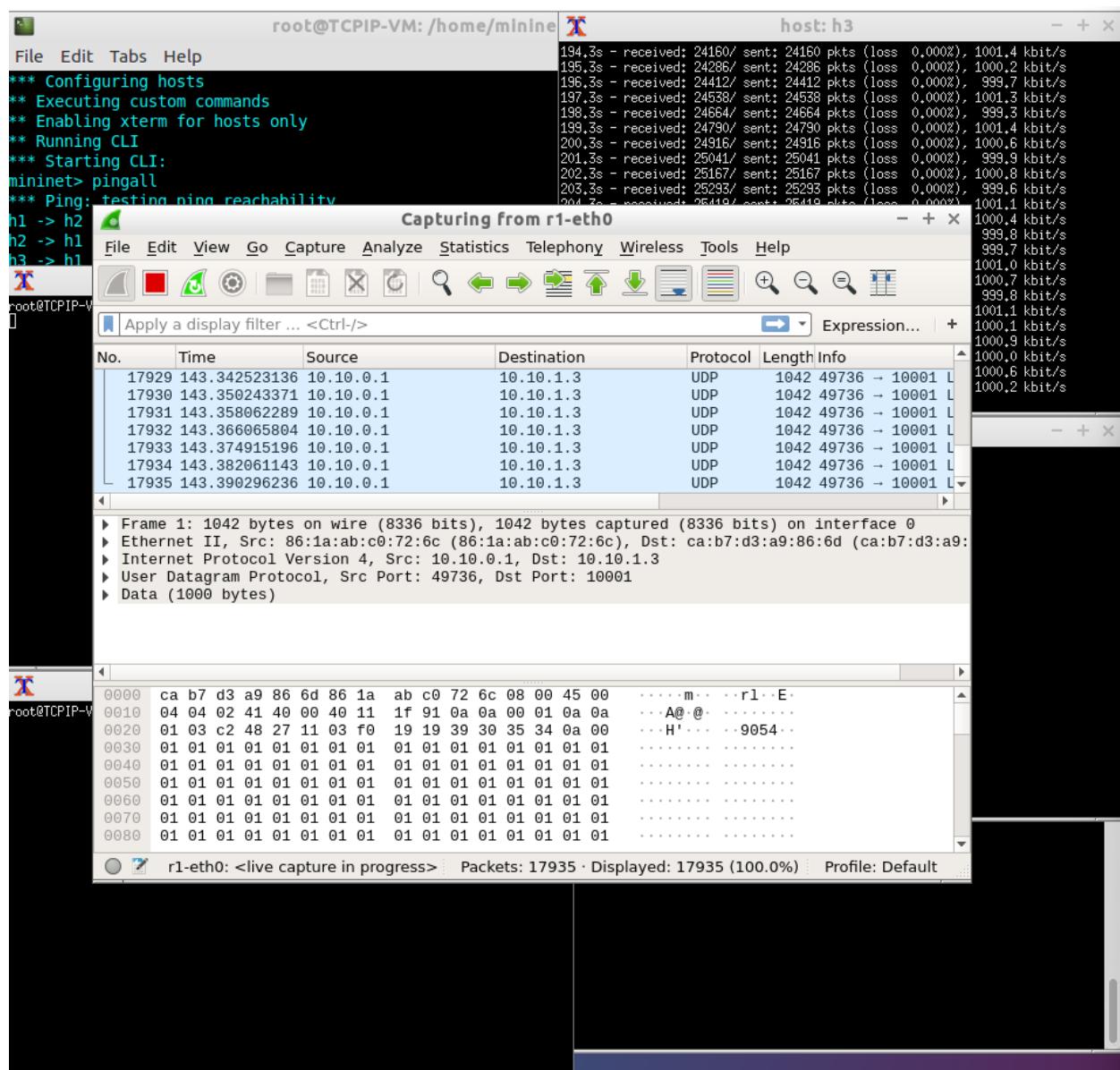
host: h1
52.3s - sent: 6544 pkts, 1000.0 kbytes/s
53.4s - sent: 6670 pkts, 1001.2 kbytes/s
54.4s - sent: 6795 pkts, 999.9 kbytes/s
^C^Crecv: Interrupted system call
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ^C
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpclient 10.10.1.3
10001 1000
 1.0s - sent: 126 pkts, 1007.6 kbytes/s
 2.0s - sent: 251 pkts, 1000.0 kbytes/s
 3.0s - sent: 377 pkts, 1000.8 kbytes/s
 4.0s - sent: 503 pkts, 1000.3 kbytes/s
 5.0s - sent: 629 pkts, 1000.1 kbytes/s
 6.0s - sent: 755 pkts, 1000.8 kbytes/s
 7.0s - sent: 880 pkts, 1000.0 kbytes/s
 8.0s - sent: 1006 pkts, 1001.3 kbytes/s
 9.0s - sent: 1131 pkts, 999.5 kbytes/s
10.0s - sent: 1257 pkts, 999.8 kbytes/s
11.1s - sent: 1383 pkts, 1001.2 kbytes/s
12.1s - sent: 1508 pkts, 998.7 kbytes/s
13.1s - sent: 1634 pkts, 999.4 kbytes/s
14.1s - sent: 1760 pkts, 1004.7 kbytes/s
15.1s - sent: 1886 pkts, 1000.4 kbytes/s
16.1s - sent: 2011 pkts, 999.8 kbytes/s
```

ب - 1 - 2

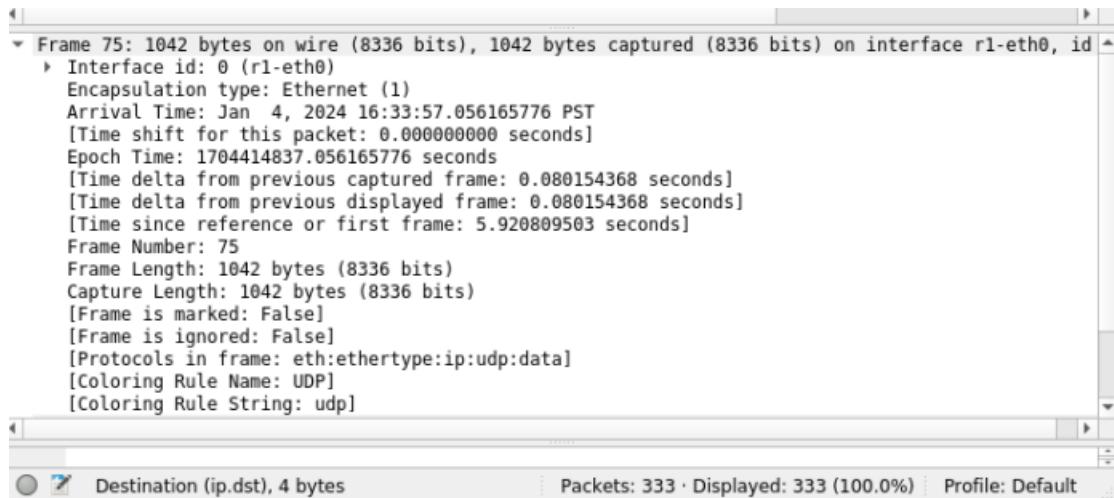
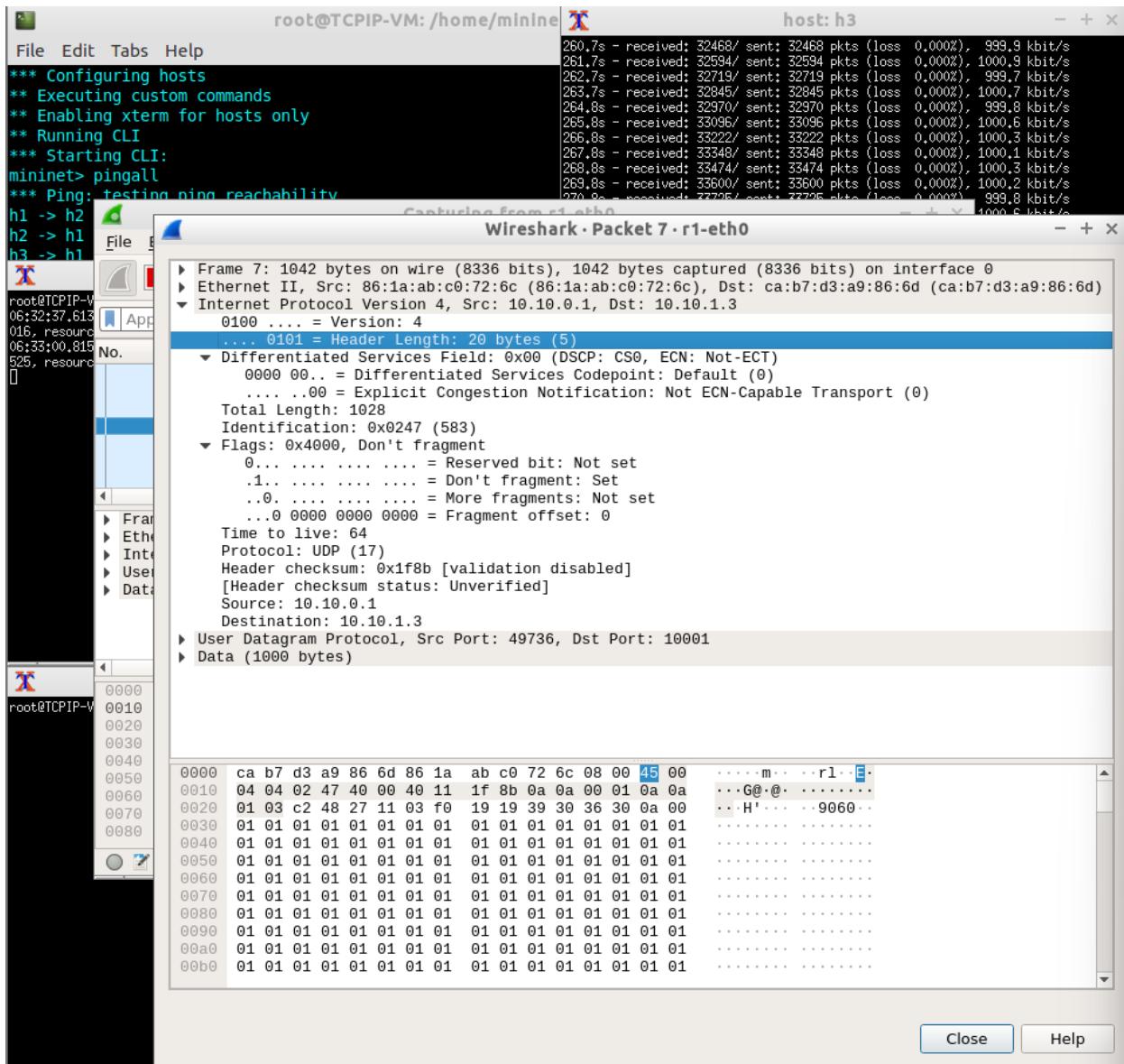
سؤال 4: به لحاظ تئوری، انتظار داریم اندازه (بر حسب بایت) فریم های Ethernet که برای ارسال داده های کلاینت استفاده می شوند، به صورت زیر باشد:

هدر (سرآیند) UDP به میزان 8 بایت، هدر IP به مقدار 20 بایت، هدر Ethernet به میزان 14 بایت و اندازه داده های Application هم که 1000 بایت؛ پس، مجموعاً 1042 بایت.

با استفاده از WireShark همخوانی واقعیت با این مقدار تئوری را کنترل کنید.  
sudo udpclient h3 و پورت 10000 اجرا می کنیم. سپس Wireshark را در r1 با دستور wireshark باز می کنیم.



با بررسی اندازه ها متوجه می شویم که تئوری با آنچه در عمل مشاهده شد مطابقت دارد.



سؤال 5: پس از اعمال محدودیت 3 Mbps در پهنهای باند، روتر در توپولوژی شکل 1 تبدیل به گلوگاه (bottleneck) شبکه می شود. به لحاظ تئوری، حداقل مقدار قابل دستیابی برای گذردهی داده های کاربردی (همان goodput) چقدر خواهد بود؟ محاسبه کنید.

$$2 * (1000 / 1042) = 2.879 \text{ Mbps}$$

سؤال 6: یک کلاینت UDP روی ماشین h1 راه اندازی نمایید که داده ها را با نرخ 100 kbps ارسال می کند. مقدار احتمال loss و همچنین goodput مشاهده شده در h3 چقدر است؟

مقدار loss و مقدار goodput برابر با 100 kbps است.

```

host: h3
43.2s - received: 339/ sent: 339 pkts (loss 0.000%), 100.0 kbit/s
44.2s - received: 352/ sent: 352 pkts (loss 0.000%), 100.0 kbit/s
45.2s - received: 365/ sent: 365 pkts (loss 0.000%), 100.0 kbit/s
46.3s - received: 378/ sent: 378 pkts (loss 0.000%), 99.9 kbit/s
47.3s - received: 391/ sent: 391 pkts (loss 0.000%), 100.1 kbit/s
48.4s - received: 404/ sent: 404 pkts (loss 0.000%), 100.0 kbit/s
49.4s - received: 417/ sent: 417 pkts (loss 0.000%), 100.0 kbit/s
50.4s - received: 430/ sent: 430 pkts (loss 0.000%), 100.0 kbit/s
51.5s - received: 443/ sent: 443 pkts (loss 0.000%), 100.0 kbit/s
52.5s - received: 456/ sent: 456 pkts (loss 0.000%), 99.9 kbit/s
53.6s - received: 469/ sent: 469 pkts (loss 0.000%), 100.2 kbit/s
54.6s - received: 482/ sent: 482 pkts (loss 0.000%), 100.0 kbit/s
55.6s - received: 495/ sent: 495 pkts (loss 0.000%), 100.0 kbit/s
56.7s - received: 508/ sent: 508 pkts (loss 0.000%), 99.9 kbit/s
57.7s - received: 521/ sent: 521 pkts (loss 0.000%), 100.1 kbit/s
58.8s - received: 534/ sent: 534 pkts (loss 0.000%), 100.0 kbit/s
59.8s - received: 547/ sent: 547 pkts (loss 0.000%), 100.0 kbit/s
60.8s - received: 560/ sent: 560 pkts (loss 0.000%), 100.0 kbit/s
61.9s - received: 573/ sent: 573 pkts (loss 0.000%), 100.1 kbit/s
62.9s - received: 586/ sent: 586 pkts (loss 0.000%), 100.0 kbit/s
64.0s - received: 599/ sent: 599 pkts (loss 0.000%), 100.0 kbit/s
65.0s - received: 612/ sent: 612 pkts (loss 0.000%), 100.0 kbit/s
packet received = 620 / 620 sent: 0.000% loss

host: h1
27.0s - sent: 339 pkts, 100.1 kbit/s
28.1s - sent: 352 pkts, 100.0 kbit/s
29.1s - sent: 365 pkts, 100.0 kbit/s
30.2s - sent: 378 pkts, 100.0 kbit/s
31.2s - sent: 391 pkts, 100.1 kbit/s
32.2s - sent: 404 pkts, 100.0 kbit/s
33.3s - sent: 417 pkts, 100.1 kbit/s
34.3s - sent: 430 pkts, 100.0 kbit/s
35.4s - sent: 443 pkts, 100.1 kbit/s
36.4s - sent: 456 pkts, 99.8 kbit/s
37.4s - sent: 469 pkts, 100.4 kbit/s
38.5s - sent: 482 pkts, 100.0 kbit/s
39.5s - sent: 495 pkts, 100.1 kbit/s
40.6s - sent: 508 pkts, 100.0 kbit/s
41.6s - sent: 521 pkts, 100.1 kbit/s
42.6s - sent: 534 pkts, 100.0 kbit/s
43.7s - sent: 547 pkts, 100.0 kbit/s
44.7s - sent: 560 pkts, 100.0 kbit/s
45.8s - sent: 573 pkts, 100.1 kbit/s
46.8s - sent: 586 pkts, 100.0 kbit/s
47.8s - sent: 599 pkts, 100.0 kbit/s
48.9s - sent: 612 pkts, 100.1 kbit/s
^Cpackets sent = 620, avg rate= 100.1kbit/s
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/udp# []

```

سؤال 7: عملیات مورد نظر در سؤال 6 را برای نرخ های 3 Mbps و 10 Mbps goodput و احتمالات loss چقدر می شود؟ مقادیر حاصل برای goodput در همه این موارد را با مقداری که انتظار دارید (و در سؤال 5 محاسبه کردید)، مقایسه نمایید.

- 3 Mbps

```
host: h3
0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 0.2 kbit/s
41.0s - received: 363/ sent: 363 pkts (loss 0.000%), 2888.0 kbit/s
42.0s - received: 723/ sent: 723 pkts (loss 0.000%), 2876.4 kbit/s
43.0s - received: 1084/ sent: 1084 pkts (loss 0.000%), 2880.3 kbit/s
44.0s - received: 1444/ sent: 1444 pkts (loss 0.000%), 2876.6 kbit/s
45.0s - received: 1805/ sent: 1805 pkts (loss 0.000%), 2883.2 kbit/s
46.0s - received: 2165/ sent: 2165 pkts (loss 0.000%), 2878.6 kbit/s
47.0s - received: 2526/ sent: 2526 pkts (loss 0.000%), 2880.1 kbit/s
48.0s - received: 2886/ sent: 2886 pkts (loss 0.000%), 2879.6 kbit/s
49.0s - received: 3246/ sent: 3246 pkts (loss 0.000%), 2879.3 kbit/s
50.0s - received: 3607/ sent: 3607 pkts (loss 0.000%), 2880.3 kbit/s
51.0s - received: 3967/ sent: 3967 pkts (loss 0.000%), 2879.4 kbit/s
52.0s - received: 4327/ sent: 4327 pkts (loss 0.000%), 2877.4 kbit/s
53.0s - received: 4688/ sent: 4688 pkts (loss 0.000%), 2881.1 kbit/s
54.0s - received: 5049/ sent: 5049 pkts (loss 0.000%), 2879.7 kbit/s
55.0s - received: 5409/ sent: 5409 pkts (loss 0.000%), 2879.9 kbit/s
56.0s - received: 5769/ sent: 5769 pkts (loss 0.000%), 2879.6 kbit/s
57.0s - received: 6129/ sent: 6129 pkts (loss 0.000%), 2879.6 kbit/s
58.0s - received: 6489/ sent: 6489 pkts (loss 0.000%), 2879.5 kbit/s
59.0s - received: 6849/ sent: 6849 pkts (loss 0.000%), 2879.2 kbit/s
60.0s - received: 7210/ sent: 7210 pkts (loss 0.000%), 2882.1 kbit/s
61.0s - received: 7570/ sent: 7570 pkts (loss 0.000%), 2879.7 kbit/s
packet received = 7667 / 7667 sent: 0.000% loss

```

```
host: h1
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ./udpclient 10.10.1.3
10000 3000
1.0s - sent: 376 pkts, 3007.4 kbits/s
2.0s - sent: 752 pkts, 3000.4 kbits/s
3.0s - sent: 1127 pkts, 2999.8 kbits/s
4.0s - sent: 1502 pkts, 2999.9 kbits/s
5.0s - sent: 1878 pkts, 3000.2 kbits/s
6.0s - sent: 2253 pkts, 2997.9 kbits/s
7.0s - sent: 2629 pkts, 3003.6 kbits/s
8.0s - sent: 3004 pkts, 2999.0 kbits/s
9.0s - sent: 3380 pkts, 3003.6 kbits/s
10.0s - sent: 3756 pkts, 3001.5 kbits/s
11.0s - sent: 4132 pkts, 3001.2 kbits/s
12.0s - sent: 4508 pkts, 2999.4 kbits/s
13.0s - sent: 4884 pkts, 3001.1 kbits/s
14.0s - sent: 5260 pkts, 3002.8 kbits/s
15.0s - sent: 5636 pkts, 3001.8 kbits/s
16.0s - sent: 6012 pkts, 3005.0 kbits/s
17.0s - sent: 6388 pkts, 2999.9 kbits/s
18.0s - sent: 6764 pkts, 3002.4 kbits/s
19.0s - sent: 7140 pkts, 3001.1 kbits/s
20.0s - sent: 7515 pkts, 2999.8 kbits/s
^Cpackets sent = 7667, avg rate=2879.1kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp#
```

مقدار loss صفر و مقدار goodput حدودا 3000 kbps است.

- 10 Mbps

```

X host: h3 - + ×
12.0s - received: 2163/ sent: 4033 pkts (loss 46.367%), 2880.0 kbit/s
13.0s - received: 2523/ sent: 5284 pkts (loss 52.252%), 2879.1 kbit/s
14.0s - received: 2883/ sent: 6535 pkts (loss 55.884%), 2877.1 kbit/s
15.0s - received: 3244/ sent: 7792 pkts (loss 58.368%), 2881.2 kbit/s
16.0s - received: 3604/ sent: 9046 pkts (loss 60.159%), 2879.6 kbit/s
17.0s - received: 3965/ sent: 11040 pkts (loss 64.085%), 2880.1 kbit/s
18.0s - received: 4325/ sent: 11551 pkts (loss 62.557%), 2878.8 kbit/s
19.0s - received: 4686/ sent: 12802 pkts (loss 63.396%), 2881.2 kbit/s
20.0s - received: 5046/ sent: 14049 pkts (loss 64.083%), 2879.2 kbit/s
21.0s - received: 5407/ sent: 15303 pkts (loss 64.667%), 2880.9 kbit/s
22.0s - received: 5767/ sent: 16553 pkts (loss 65.160%), 2879.7 kbit/s
23.0s - received: 6127/ sent: 17804 pkts (loss 65.586%), 2879.6 kbit/s
24.0s - received: 6487/ sent: 19054 pkts (loss 65.955%), 2874.3 kbit/s
25.0s - received: 6848/ sent: 20309 pkts (loss 66.281%), 2879.2 kbit/s
26.0s - received: 7209/ sent: 21563 pkts (loss 66.568%), 2881.2 kbit/s
27.1s - received: 7569/ sent: 22815 pkts (loss 66.824%), 2879.3 kbit/s
28.1s - received: 7930/ sent: 24069 pkts (loss 67.053%), 2881.6 kbit/s
29.1s - received: 8290/ sent: 25320 pkts (loss 67.259%), 2879.7 kbit/s
30.1s - received: 8651/ sent: 26574 pkts (loss 67.446%), 2881.1 kbit/s
31.1s - received: 9011/ sent: 27825 pkts (loss 67.615%), 2880.0 kbit/s
32.1s - received: 9372/ sent: 29079 pkts (loss 67.771%), 2880.2 kbit/s
33.1s - received: 9732/ sent: 30329 pkts (loss 67.912%), 2878.8 kbit/s
packet received = 9737 / 30347 sent: 67.914% loss
X host: h1 - + ×
3.0s - sent: 3754 pkts, 10001.4 kbits/s
4.0s - sent: 5005 pkts, 10001.4 kbits/s
5.0s - sent: 6256 pkts, 10001.5 kbits/s
6.0s - sent: 7507 pkts, 9998.5 kbits/s
7.0s - sent: 8758 pkts, 10004.2 kbits/s
8.0s - sent: 10009 pkts, 10000.8 kbits/s
9.0s - sent: 11260 pkts, 10001.3 kbits/s
10.0s - sent: 12512 pkts, 10015.2 kbits/s
11.0s - sent: 13766 pkts, 10025.1 kbits/s
12.0s - sent: 15017 pkts, 10007.9 kbits/s
13.0s - sent: 16271 pkts, 10030.4 kbits/s
14.0s - sent: 17522 pkts, 9993.2 kbits/s
15.0s - sent: 18777 pkts, 10033.8 kbits/s
16.0s - sent: 20028 pkts, 10005.2 kbits/s
17.0s - sent: 21279 pkts, 10004.4 kbits/s
18.0s - sent: 22530 pkts, 9990.6 kbits/s
19.0s - sent: 23784 pkts, 10026.5 kbits/s
20.0s - sent: 25038 pkts, 10027.1 kbits/s
21.0s - sent: 26287 pkts, 9980.8 kbits/s
22.0s - sent: 27543 pkts, 10028.6 kbits/s
23.0s - sent: 28797 pkts, 10024.7 kbits/s
24.0s - sent: 30048 pkts, 10003.0 kbits/s
^Cpackets sent = 30348, avg rate=8973.1kbytes/s
root@TCP-VM:/home/mininet/Desktop/shared/Session6/udp#
```

مقدار loss تقریبا برابر با 68 درصد است که به دلیل بیشتر بودن نرخ ارسال از ظرفیت شبکه این اتفاق رخ داده است. همچنین مقدار goodput نیز تقریبا برابر با 2875 تا 2885 می باشد که بسیار نزدیک به مقدار تئوری می باشد.

## ب - 2 (2-2) تست TCP

سؤال 8: انتظار داریم اندازه (برحسب بایت) فریم های اترنتی که برای ارسال داده ها توسط ارتباط می شوند برابر با 1514 بایت باشد چراکه: با توجه به MTU ، داده های برنامه کاربردی 1448 بایت، هدیر اترنت 20 بایت، هدیر IP 14 بایت و هدیر TCP برابر با 32 بایت است. این فرضیه را از طریق گوش دادن به بسته ها در سمت سرور بررسی کنید.

ابتدا با استفاده از دستور زیر ethtool را نصب می کنیم:

```
sudo apt-get install -y ethtool
```

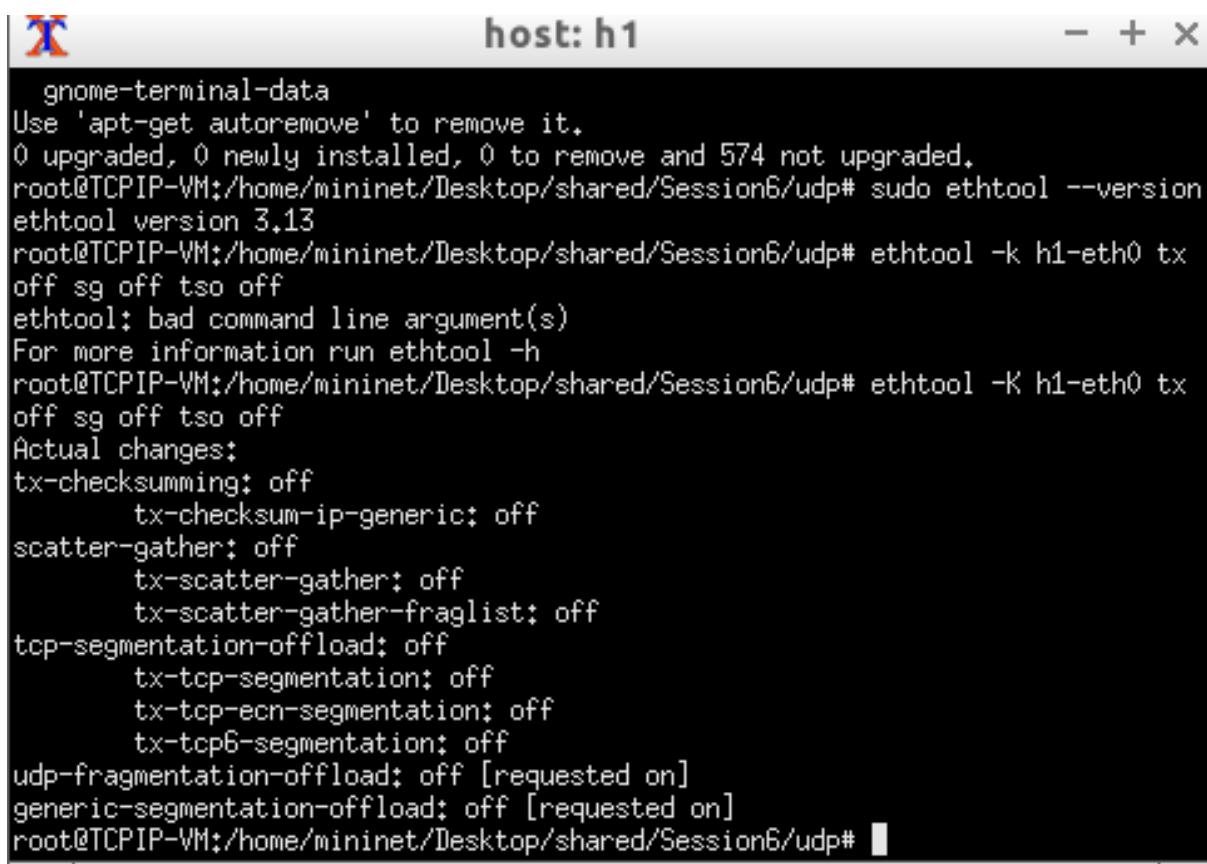
با چک کردن ورژن از نصب شدن آن مطمئن می شویم:

```
sudo ethtool --version
```

دستور موجود در دستورکار (دستور زیر) را اجرا می کنیم:

```
ethtool -K h1-eth0 tx off sg off tso off
```

نتایج به صورت زیر خواهد بود:

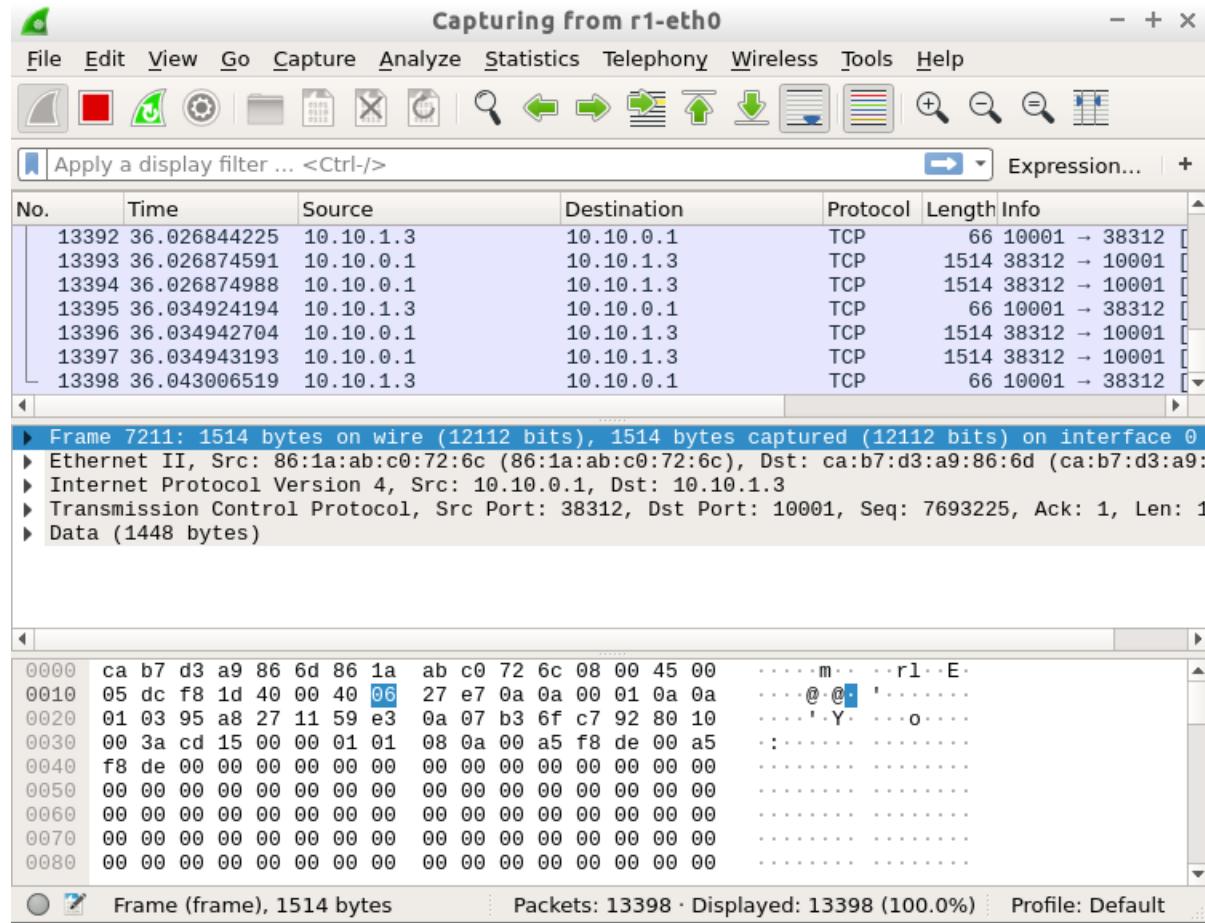


```
host: h1
gnome-terminal-data
Use 'apt-get autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 574 not upgraded.
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# sudo ethtool --version
ethtool version 3.13
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ethtool -K h1-eth0 tx
off sg off tso off
ethtool: bad command line argument(s)
For more information run ethtool -h
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp# ethtool -K h1-eth0 tx
off sg off tso off
Actual changes:
tx-checksumming: off
    tx-checksum-ip-generic: off
scatter-gather: off
    tx-scatter-gather: off
    tx-scatter-gather-fraglist: off
tcp-segmentation-offload: off
    tx-tcp-segmentation: off
    tx-tcp-ecn-segmentation: off
    tx-tcp6-segmentation: off
udp-fragmentation-offload: off [requested on]
generic-segmentation-offload: off [requested on]
root@TCPIP-VM:/home/mininet/Desktop/shared/Session6/udp#
```

در ادامه یک ارتباط TCP بین h3 و h1 ایجاد می کنیم و ترافیک روی r1-eth0 را بررسی می کنیم:

```
host: h3
18.0s - received: 4325/ sent: 11551 pkts (loss 62.557%), 2878.8 kbit/s
19.0s - received: 4686/ sent: 12802 pkts (loss 63.396%), 2881.2 kbit/s
20.0s - received: 5046/ sent: 14049 pkts (loss 64.083%), 2879.2 kbit/s
21.0s - received: 5407/ sent: 15303 pkts (loss 64.667%), 2880.9 kbit/s
22.0s - received: 5767/ sent: 16553 pkts (loss 65.160%), 2879.7 kbit/s
23.0s - received: 6127/ sent: 17804 pkts (loss 65.586%), 2879.6 kbit/s
24.0s - received: 6487/ sent: 19054 pkts (loss 65.955%), 2874.3 kbit/s
25.0s - received: 6848/ sent: 20309 pkts (loss 66.281%), 2879.2 kbit/s
26.0s - received: 7209/ sent: 21563 pkts (loss 66.568%), 2881.2 kbit/s
27.1s - received: 7569/ sent: 22815 pkts (loss 66.824%), 2879.3 kbit/s
28.1s - received: 7930/ sent: 24069 pkts (loss 67.053%), 2881.6 kbit/s
29.1s - received: 8290/ sent: 25320 pkts (loss 67.259%), 2879.7 kbit/s
30.1s - received: 8651/ sent: 26574 pkts (loss 67.446%), 2881.1 kbit/s
31.1s - received: 9011/ sent: 27825 pkts (loss 67.615%), 2880.0 kbit/s
32.1s - received: 9372/ sent: 29079 pkts (loss 67.771%), 2880.2 kbit/s
33.1s - received: 9732/ sent: 30329 pkts (loss 67.912%), 2878.8 kbit/s
packet received = 9737 / 30347 sent: 67.914% loss
^C
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/udp# cd ..
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6# cd tcp
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/tcp# ./tcpserver 10001
Handling client 10.10.0.1
with child process: 23363
[]
```

```
host: h1
13.6: 2288.0kbps avg ( 5595.0[inst], 2618.7[mov,avg]) cwnd 1000 rtt2773.5ms
14.6: 2515.6kbps avg ( 5611.0[inst], 2918.0[mov,avg]) cwnd 1000 rtt3274.0ms
15.6: 2711.8kbps avg ( 5576.0[inst], 3183.8[mov,avg]) cwnd 1000 rtt3769.5ms
16.6: 2717.4kbps avg ( 2805.5[inst], 3145.9[mov,avg]) cwnd 1000 rtt4036.0ms
17.6: 2563.0kbps avg ( 0.0[inst], 2831.3[mov,avg]) cwnd 1000 rtt4035.5ms
19.6: 2301.5kbps avg ( -0.0[inst], 2548.2[mov,avg]) cwnd 1000 rtt4036.0ms
21.1: 2802.7kbps avg ( 9217.0[inst], 3215.1[mov,avg]) cwnd 1001 rtt4043.5ms
22.1: 2802.8kbps avg ( 2805.5[inst], 3174.1[mov,avg]) cwnd 1001 rtt4043.5ms
23.1: 2802.0kbps avg ( 2782.9[inst], 3135.0[mov,avg]) cwnd 1001 rtt4044.0ms
24.1: 2802.1kbps avg ( 2805.5[inst], 3102.0[mov,avg]) cwnd 1001 rtt4043.5ms
25.1: 2802.1kbps avg ( 2805.5[inst], 3102.0[mov,avg]) cwnd 883 rtt4036.0ms
25.1: 2696.0kbps avg ( 135.7[inst], 2805.4[mov,avg]) cwnd 883 rtt4036.0ms
26.1: 2696.0kbps avg ( 135.7[inst], 2805.4[mov,avg]) cwnd 759 rtt4040.0ms
27.1: 2497.3kbps avg ( -0.0[inst], 2524.9[mov,avg]) cwnd 635 rtt4036.0ms
28.1: 2497.3kbps avg ( -0.0[inst], 2524.9[mov,avg]) cwnd 511 rtt4036.0ms
28.2: 2802.7kbps avg (10388.2[inst], 3311.2[mov,avg]) cwnd 500 rtt4038.0ms
29.2: 2802.4kbps avg ( 2794.0[inst], 3259.5[mov,avg]) cwnd 501 rtt3107.5ms
30.2: 2802.5kbps avg ( 2805.5[inst], 3214.1[mov,avg]) cwnd 501 rtt2106.5ms
31.2: 2801.9kbps avg ( 2782.9[inst], 3171.0[mov,avg]) cwnd 501 rtt2027.5ms
32.2: 2802.0kbps avg ( 2805.5[inst], 3134.4[mov,avg]) cwnd 501 rtt2028.0ms
33.2: 2802.0kbps avg ( 2805.5[inst], 3134.4[mov,avg]) cwnd 502 rtt2028.0ms
33.2: 2802.1kbps avg ( 2805.5[inst], 3101.5[mov,avg]) cwnd 502 rtt2028.0ms
34.2: 2802.2kbps avg ( 2805.5[inst], 3071.9[mov,avg]) cwnd 502 rtt2028.0ms
```



همان طور که مشاهده می شود، فریم های اترنتی که ارسال داده ها توسط TCP استفاده می شوند برابر با 1514 بایت است زیرا با توجه به MTU، داده های برنامه های کاربردی 1448 بایت، هدر آی پی 20 بایت، هدر اترنت 14 بایت و هدر TCP برابر با 32 بایت می باشد.

سؤال 9: به لحاظ تئوری، حداکثر مقدار مورد انتظار برای goodput داده های کاربردی چقدر است؟ نحوه محاسبه خود را تشریح کنید.

$$3 * (1448 / 1514) = 2.869 \text{ Mbps}$$

سؤال 10: یک کلاینت TCP روی ماشین h1 راه اندازی کنید که داده هایی را برای سرور واقع در h3 می فرستد. Goodput چقدر است؟ آن را با مقدار تئوری مورد سؤال 9 مقایسه نمایید.

host: h3

```

18.0s - received: 4325/ sent: 11551 pkts (loss 62.557%), 2878.8 kbit/s
19.0s - received: 4686/ sent: 12802 pkts (loss 63.396%), 2881.2 kbit/s
20.0s - received: 5046/ sent: 14049 pkts (loss 64.083%), 2879.2 kbit/s
21.0s - received: 5407/ sent: 15303 pkts (loss 64.667%), 2880.9 kbit/s
22.0s - received: 5767/ sent: 16553 pkts (loss 65.160%), 2879.7 kbit/s
23.0s - received: 6127/ sent: 17804 pkts (loss 65.586%), 2879.6 kbit/s
24.0s - received: 6487/ sent: 19054 pkts (loss 65.955%), 2874.3 kbit/s
25.0s - received: 6848/ sent: 20309 pkts (loss 66.281%), 2879.2 kbit/s
26.0s - received: 7209/ sent: 21563 pkts (loss 66.568%), 2881.2 kbit/s
27.1s - received: 7569/ sent: 22815 pkts (loss 66.824%), 2879.3 kbit/s
28.1s - received: 7930/ sent: 24069 pkts (loss 67.053%), 2881.6 kbit/s
29.1s - received: 8290/ sent: 25320 pkts (loss 67.259%), 2879.7 kbit/s
30.1s - received: 8651/ sent: 26574 pkts (loss 67.446%), 2881.1 kbit/s
31.1s - received: 9011/ sent: 27825 pkts (loss 67.615%), 2880.0 kbit/s
32.1s - received: 9372/ sent: 29079 pkts (loss 67.771%), 2880.2 kbit/s
3.1s - received: 9732/ sent: 30329 pkts (loss 67.912%), 2878.8 kbit/s
socket received = 9737 / 30347 sent: 67.914% loss

```

```

8 ot@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/udp# cd ..
8 ot@TCPPIP-VM:/home/mininet/Desktop/shared/Session6# cd tcp
8 ot@TCPPIP-VM:/home/mininet/Desktop/shared/Session6/tcp# ./tcpserver 10001
  handling client 10.10.0.1
  third child process: 23363

```

host: h1

```

145.2: 2797.3kbps avg ( 2782.9[inst], 2793.9[mov,avg]) cwnd 529 rtt2135.0ms
145.2: 2797.3kbps avg ( 2805.5[inst], 2795.0[mov,avg]) cwnd 529 rtt2135.0ms
146.2: 2797.4kbps avg ( 2805.5[inst], 2796.1[mov,avg]) cwnd 529 rtt2132.0ms
147.2: 2797.4kbps avg ( 2804.9[inst], 2797.0[mov,avg]) cwnd 529 rtt2140.5ms
148.2: 2797.4kbps avg ( 2796.3[inst], 2796.9[mov,avg]) cwnd 529 rtt2140.5ms
149.2: 2797.5kbps avg ( 2805.5[inst], 2797.7[mov,avg]) cwnd 529 rtt2140.0ms
150.2: 2797.5kbps avg ( 2805.5[inst], 2797.7[mov,avg]) cwnd 530 rtt2141.5ms
150.2: 2797.5kbps avg ( 2805.2[inst], 2798.5[mov,avg]) cwnd 530 rtt2141.5ms
151.2: 2797.6kbps avg ( 2803.5[inst], 2799.0[mov,avg]) cwnd 530 rtt2140.0ms
153.2: 2797.7kbps avg ( 2806.8[inst], 2799.8[mov,avg]) cwnd 530 rtt2140.5ms
154.2: 2797.6kbps avg ( 2782.3[inst], 2798.0[mov,avg]) cwnd 531 rtt2140.0ms
155.2: 2797.6kbps avg ( 2805.5[inst], 2798.8[mov,avg]) cwnd 531 rtt2140.0ms
156.2: 2797.7kbps avg ( 2805.4[inst], 2799.4[mov,avg]) cwnd 531 rtt2148.0ms
157.2: 2797.7kbps avg ( 2805.5[inst], 2800.0[mov,avg]) cwnd 531 rtt2148.5ms
158.2: 2797.8kbps avg ( 2805.2[inst], 2800.6[mov,avg]) cwnd 532 rtt2149.5ms
159.2: 2797.8kbps avg ( 2805.5[inst], 2801.0[mov,avg]) cwnd 532 rtt2150.0ms
160.2: 2797.7kbps avg ( 2781.7[inst], 2799.1[mov,avg]) cwnd 532 rtt2148.0ms
161.2: 2797.9kbps avg ( 2818.7[inst], 2801.1[mov,avg]) cwnd 532 rtt2148.0ms
162.2: 2797.9kbps avg ( 2803.6[inst], 2801.3[mov,avg]) cwnd 533 rtt2148.5ms
163.2: 2797.8kbps avg ( 2782.8[inst], 2799.5[mov,avg]) cwnd 533 rtt2148.0ms
164.2: 2797.9kbps avg ( 2818.9[inst], 2801.4[mov,avg]) cwnd 533 rtt2150.0ms
165.2: 2798.0kbps avg ( 2805.5[inst], 2801.8[mov,avg]) cwnd 533 rtt2158.5ms
166.2: 2797.9kbps avg ( 2782.4[inst], 2799.9[mov,avg]) cwnd 533 rtt2159.0ms

```

مقدار به دست آمده برای goodput 2797 kbps بوده که نزدیک به مقدار محاسبه شده در تئوری است.



## دانشکده مهندسی کامپیوتر

### آزمایشگاه شبکه های کامپیوتری

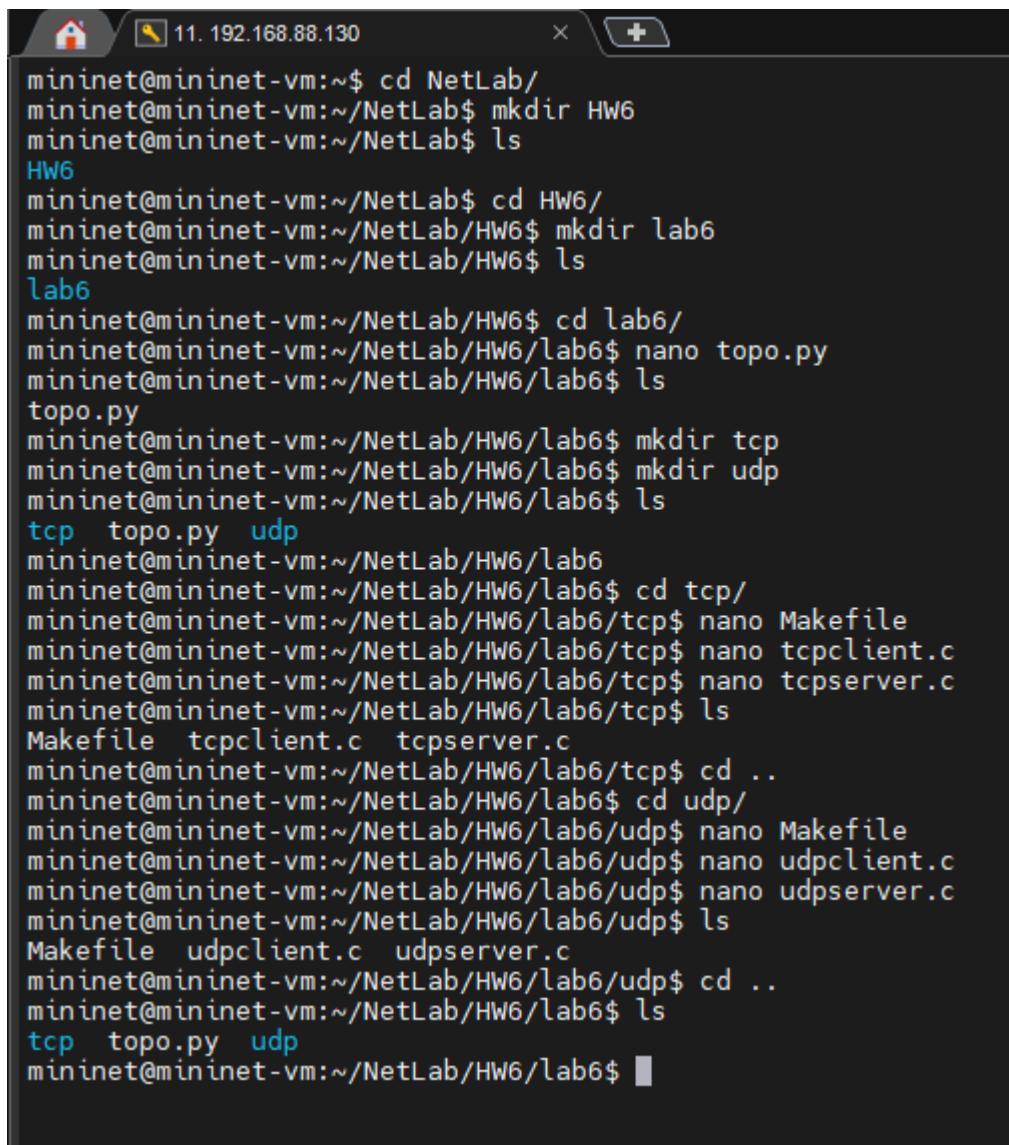
استاد : شکوفه نوروزی

پوریا رحیمی (99521289)

بکتاش انصاری (99521082)

پاییز 1402

در ابتدا برای انجام این آزمایش فolder مناسب را ایجاد کرده و فایل ها و توپولوژی مربوط به این آزمایش را در فolder های مناسب ایجاد می کنیم :



```
mininet@mininet-vm:~$ cd NetLab/
mininet@mininet-vm:~/NetLab$ mkdir HW6
mininet@mininet-vm:~/NetLab$ ls
HW6
mininet@mininet-vm:~/NetLab$ cd HW6/
mininet@mininet-vm:~/NetLab/HW6$ mkdir lab6
mininet@mininet-vm:~/NetLab/HW6$ ls
lab6
mininet@mininet-vm:~/NetLab/HW6$ cd lab6/
mininet@mininet-vm:~/NetLab/HW6/lab6$ nano topo.py
mininet@mininet-vm:~/NetLab/HW6/lab6$ ls
topo.py
mininet@mininet-vm:~/NetLab/HW6/lab6$ mkdir tcp
mininet@mininet-vm:~/NetLab/HW6/lab6$ mkdir udp
mininet@mininet-vm:~/NetLab/HW6/lab6$ ls
tcp topo.py udp
mininet@mininet-vm:~/NetLab/HW6/lab6
mininet@mininet-vm:~/NetLab/HW6/lab6$ cd tcp/
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ nano Makefile
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ nano tcpclient.c
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ nano tcpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ ls
Makefile tcpclient.c tcpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ cd ..
mininet@mininet-vm:~/NetLab/HW6/lab6$ cd udp/
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ nano Makefile
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ nano udpclient.c
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ nano udpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ ls
Makefile udpclient.c udpserver.c
mininet@mininet-vm:~/NetLab/HW6/udp$ cd ..
mininet@mininet-vm:~/NetLab/HW6$ ls
tcp topo.py udp
mininet@mininet-vm:~/NetLab/HW6$
```

سپس بعد از ایجاد فایل ها و فolder های مناسب فایل `tcpclient` , `tcpserver` , `udpclient` را با توجه به دستور زیر کامپایل می کنیم :

```
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ cd tcp
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ ls
Makefile tcpclient.c tcpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ ls
Makefile tcpclient.c tcpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ gcc tcpclient.c -o tcpclient
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ ls
Makefile tcpclient tcpclient.c tcpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ gcc tcpserver.c -o tcpserver
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$ ls
Makefile tcpclient tcpclient.c tcpserver tcpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/tcp$
```

mininet-vm 0% 0.26 GB / 0.96 GB 0.01 Mb/s 0.00 Mb/s 52 min mininet (x2) ← → ×

```
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ ls
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ ls
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ ls
Makefile udpclient.c udpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ gcc udpclient.c -o udpclient
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ ls
Makefile udpclient udpclient.c udpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ gcc udpserver.c -o udpserver
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ ls
Makefile udpclient udpclient.c udpserver udpserver.c
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$
```

mininet-vm 0% 0.26 GB / 0.96 GB 0.01 Mb/s 0.00 Mb/s 53 min mininet (x2) ← → ×

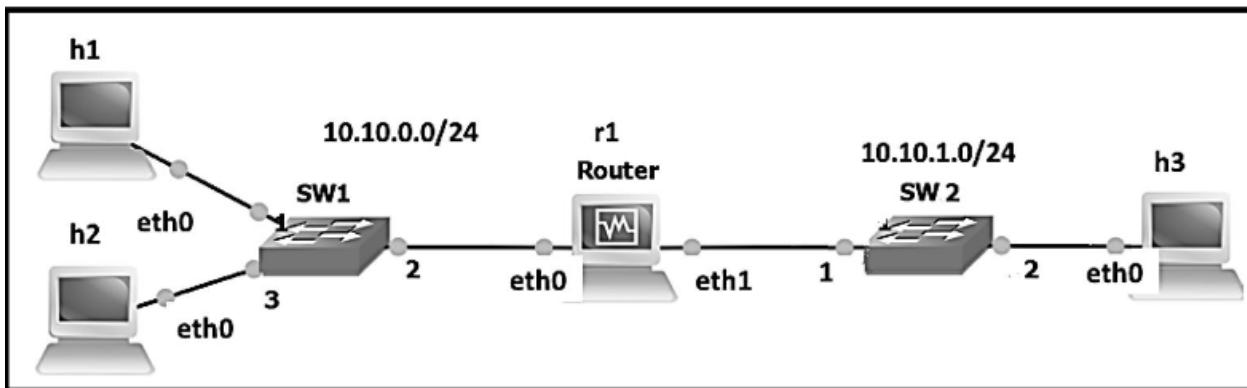
حال برنامه های مناسب را با توجه به دستورات به حالت اجرایی در می آوریم  
سپس مکانیزم کنترل ازدحام را بررسی می کنیم.

سپس طبق چیزی که گفته شده است ما باید از مکانیزم **reno** استفاده کنیم به همین  
منظور طبق دستور موجود مکانیزم را به **reno** تغییر می دهیم :

```
mininet@mininet-vm:~/NetLab/HW6/lab6/udp$ cd ..
mininet@mininet-vm:~/NetLab/HW6/lab6$ ls
tcp topo.py udp
mininet@mininet-vm:~/NetLab/HW6/lab6$ chmod +x tcp/tcpclient tcp/tcpserver udp/udpclient udp/ud
pserver
mininet@mininet-vm:~/NetLab/HW6/lab6$ cat /proc/sys/net/ipv4/tcp_congestion_control
cubic
mininet@mininet-vm:~/NetLab/HW6/lab6$ sudo bash -c 'echo reno >/proc/sys/net/ipv4/tcp_congestion_control'
mininet@mininet-vm:~/NetLab/HW6/lab6$ cat /proc/sys/net/ipv4/tcp_congestion_control
reno
mininet@mininet-vm:~/NetLab/HW6/lab6$
```

mininet-vm 1% 0.26 GB / 0.96 GB 0.01 Mb/s 0.00 Mb/s 56 min mininet (x2) /: 46% /boot/efi: 1%

حال فایل topo.py را به گونه ای کامل می کنیم که توپولوژی زیر را ایجاد کند :

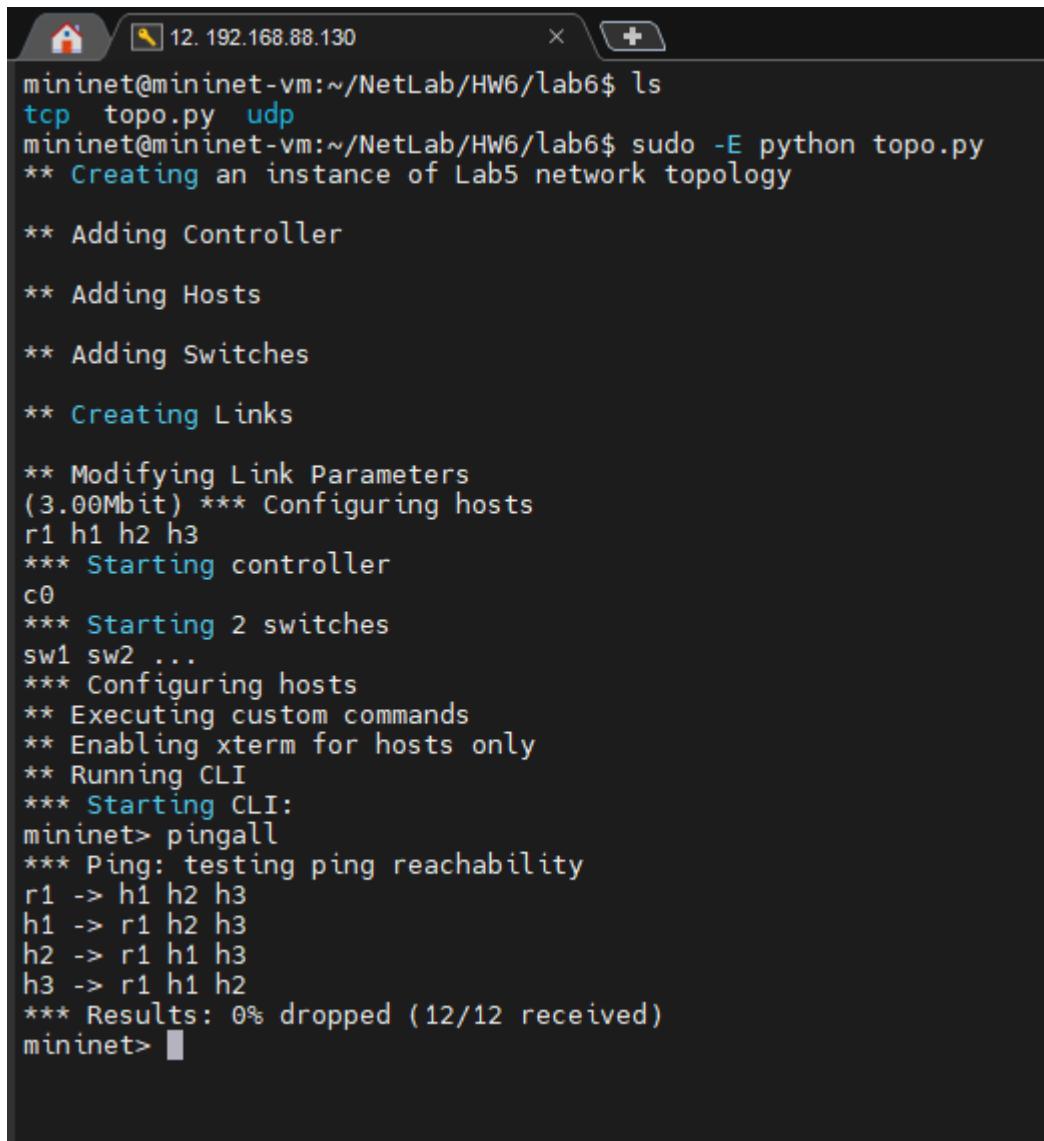


```
h1 = net.addHost('h1', ip='10.10.0.1/24')
h2 = net.addHost('h2', ip='10.10.0.2/24')
h3 = net.addHost('h3', ip='10.10.1.3/24')
r1 = net.addHost('r1', ip='10.10.0.10/24')
```

```
r1.cmd('ifconfig r1-eth1 10.10.1.10 netmask 255.255.255.0')
r1.cmd('echo 1 >/proc/sys/net/ipv4/ip_forward')
h1.cmd('ip route add default via 10.10.0.10')
h2.cmd('ip route add default via 10.10.0.10')
h3.cmd('ip route add default via 10.10.1.10')
```

حال بعد از کامل کردن توپولوژی با استفاده از دستور pingall تست می کنیم ببینیم که آیا توپولوژی به درستی کار می کند یا خیر :

سپس در شکل زیر مشاهده می کنیم که هیچ بسته ای drop نشده است در نتیجه توپولوژی ما به درستی کار می کند.



```
mininet@mininet-vm:~/NetLab/HW6/lab6$ ls
tcp topo.py udp
mininet@mininet-vm:~/NetLab/HW6/lab6$ sudo -E python topo.py
** Creating an instance of Lab5 network topology

** Adding Controller

** Adding Hosts

** Adding Switches

** Creating Links

** Modifying Link Parameters
(3.00Mbit) *** Configuring hosts
r1 h1 h2 h3
*** 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
r1 -> h1 h2 h3
h1 -> r1 h2 h3
h2 -> r1 h1 h3
h3 -> r1 h1 h2
*** Results: 0% dropped (12/12 received)
mininet> █
```

ب(1)

سوال 1) یک سرور UDP در h3 راه اندازی می کنیم که روی پورت 10000 گوش می کند سپس بر روی h1 یک کلاینت ایجاد می کنیم که برای این پورت از h3 داده می فرستد. این آزمایش را دوبار انجام می دهیم تا میزان اطمینان ما بیشتر شود :

X "host: h3"@mininet-vm

```
root@mininet-vm:~/NetLab/HW6/lab6# cd udp/
root@mininet-vm:~/NetLab/HW6/lab6/udp# ./udpserver 10000
```

X "host: h1"@mininet-vm

```
root@mininet-vm:~/NetLab/HW6/lab6/udp# ./udpcclient 10.10.1.3 10000 100
```

X "host: h1"@mininet-vm

```
142.5s - sent: 1782 pkts, 100.0 kbits/s
143.5s - sent: 1795 pkts, 100.1 kbits/s
144.6s - sent: 1808 pkts, 100.0 kbits/s
145.6s - sent: 1821 pkts, 100.1 kbits/s
146.6s - sent: 1834 pkts, 100.0 kbits/s
147.7s - sent: 1847 pkts, 100.0 kbits/s
148.7s - sent: 1860 pkts, 99.9 kbits/s
149.8s - sent: 1873 pkts, 100.1 kbits/s
150.8s - sent: 1886 pkts, 100.1 kbits/s
151.8s - sent: 1899 pkts, 100.1 kbits/s
152.9s - sent: 1912 pkts, 99.9 kbits/s
153.9s - sent: 1925 pkts, 100.0 kbits/s
155.0s - sent: 1938 pkts, 100.0 kbits/s
156.0s - sent: 1951 pkts, 100.0 kbits/s
157.0s - sent: 1964 pkts, 100.0 kbits/s
158.1s - sent: 1977 pkts, 100.1 kbits/s
159.1s - sent: 1990 pkts, 100.0 kbits/s
160.2s - sent: 2003 pkts, 100.0 kbits/s
161.2s - sent: 2016 pkts, 100.0 kbits/s
162.2s - sent: 2029 pkts, 100.0 kbits/s
163.3s - sent: 2042 pkts, 100.0 kbits/s
164.3s - sent: 2055 pkts, 100.0 kbits/s
^Cpackets sent = 2066, avg rate= 100.0kbps
root@mininet-vm:~/NetLab/HW6/lab6/udp#
```

X "host: h3"@mininet-vm

```
254.7s - received: 1782/ sent: 1782 pkts (loss 0.000%), 100.0 kbit/s
255.7s - received: 1795/ sent: 1795 pkts (loss 0.000%), 100.0 kbit/s
256.8s - received: 1808/ sent: 1808 pkts (loss 0.000%), 100.0 kbit/s
257.8s - received: 1821/ sent: 1821 pkts (loss 0.000%), 100.0 kbit/s
258.8s - received: 1834/ sent: 1834 pkts (loss 0.000%), 100.0 kbit/s
259.9s - received: 1847/ sent: 1847 pkts (loss 0.000%), 100.0 kbit/s
260.9s - received: 1860/ sent: 1860 pkts (loss 0.000%), 99.9 kbit/s
262.0s - received: 1873/ sent: 1873 pkts (loss 0.000%), 100.0 kbit/s
263.0s - received: 1886/ sent: 1886 pkts (loss 0.000%), 100.1 kbit/s
264.0s - received: 1899/ sent: 1899 pkts (loss 0.000%), 100.0 kbit/s
265.1s - received: 1912/ sent: 1912 pkts (loss 0.000%), 99.9 kbit/s
266.1s - received: 1925/ sent: 1925 pkts (loss 0.000%), 100.0 kbit/s
267.2s - received: 1938/ sent: 1938 pkts (loss 0.000%), 100.0 kbit/s
268.2s - received: 1951/ sent: 1951 pkts (loss 0.000%), 100.0 kbit/s
269.2s - received: 1964/ sent: 1964 pkts (loss 0.000%), 100.0 kbit/s
270.3s - received: 1977/ sent: 1977 pkts (loss 0.000%), 100.1 kbit/s
271.3s - received: 1990/ sent: 1990 pkts (loss 0.000%), 100.0 kbit/s
272.4s - received: 2003/ sent: 2003 pkts (loss 0.000%), 99.9 kbit/s
273.4s - received: 2016/ sent: 2016 pkts (loss 0.000%), 100.0 kbit/s
274.4s - received: 2029/ sent: 2029 pkts (loss 0.000%), 100.0 kbit/s
275.5s - received: 2042/ sent: 2042 pkts (loss 0.000%), 100.0 kbit/s
276.5s - received: 2055/ sent: 2055 pkts (loss 0.000%), 100.0 kbit/s
packet received = 2066 / 2066 sent: 0.000% loss
```

X "host: h1"@mininet-vm

```
29.1s - sent: 365 pkts, 100.0 kbits/s
30.2s - sent: 378 pkts, 100.0 kbits/s
31.2s - sent: 391 pkts, 100.0 kbits/s
32.2s - sent: 404 pkts, 100.0 kbits/s
33.3s - sent: 417 pkts, 100.0 kbits/s
34.3s - sent: 430 pkts, 100.1 kbits/s
35.4s - sent: 443 pkts, 100.1 kbits/s
36.4s - sent: 456 pkts, 100.0 kbits/s
37.4s - sent: 469 pkts, 100.0 kbits/s
38.5s - sent: 482 pkts, 100.1 kbits/s
39.5s - sent: 495 pkts, 100.0 kbits/s
40.6s - sent: 508 pkts, 100.0 kbits/s
41.6s - sent: 521 pkts, 100.0 kbits/s
42.6s - sent: 534 pkts, 100.0 kbits/s
43.7s - sent: 547 pkts, 100.0 kbits/s
44.7s - sent: 560 pkts, 99.9 kbits/s
45.8s - sent: 573 pkts, 100.0 kbits/s
46.8s - sent: 586 pkts, 100.0 kbits/s
47.8s - sent: 599 pkts, 100.0 kbits/s
48.9s - sent: 612 pkts, 100.0 kbits/s
49.9s - sent: 625 pkts, 100.0 kbits/s
51.0s - sent: 638 pkts, 100.0 kbits/s
^D packets sent = 643, avg rate= 100.0 kbps
```

```

X "host: h3"@mininet-vm
-
□
X

34.4s - received: 365/ sent: 365 pkts (loss 0.000%), 100.0 kbit/s
35.5s - received: 378/ sent: 378 pkts (loss 0.000%), 100.0 kbit/s
36.5s - received: 391/ sent: 391 pkts (loss 0.000%), 100.0 kbit/s
37.6s - received: 404/ sent: 404 pkts (loss 0.000%), 100.0 kbit/s
38.6s - received: 417/ sent: 417 pkts (loss 0.000%), 100.0 kbit/s
39.6s - received: 430/ sent: 430 pkts (loss 0.000%), 100.0 kbit/s
40.7s - received: 443/ sent: 443 pkts (loss 0.000%), 100.0 kbit/s
41.7s - received: 456/ sent: 456 pkts (loss 0.000%), 100.0 kbit/s
42.8s - received: 469/ sent: 469 pkts (loss 0.000%), 100.0 kbit/s
43.8s - received: 482/ sent: 482 pkts (loss 0.000%), 100.0 kbit/s
44.8s - received: 495/ sent: 495 pkts (loss 0.000%), 100.0 kbit/s
45.9s - received: 508/ sent: 508 pkts (loss 0.000%), 100.0 kbit/s
46.9s - received: 521/ sent: 521 pkts (loss 0.000%), 100.0 kbit/s
48.0s - received: 534/ sent: 534 pkts (loss 0.000%), 100.0 kbit/s
49.0s - received: 547/ sent: 547 pkts (loss 0.000%), 100.0 kbit/s
50.0s - received: 560/ sent: 560 pkts (loss 0.000%), 99.9 kbit/s
51.1s - received: 573/ sent: 573 pkts (loss 0.000%), 100.0 kbit/s
52.1s - received: 586/ sent: 586 pkts (loss 0.000%), 100.0 kbit/s
53.2s - received: 599/ sent: 599 pkts (loss 0.000%), 100.1 kbit/s
54.2s - received: 612/ sent: 612 pkts (loss 0.000%), 100.0 kbit/s
55.2s - received: 625/ sent: 625 pkts (loss 0.000%), 100.0 kbit/s
56.3s - received: 638/ sent: 638 pkts (loss 0.000%), 100.0 kbit/s
packet received = 643 / 643 sent: 0.000% loss

```

همانطور که مشاهده می شود مقدار loss در h3 برابر با 0 بوده و علاوه بر آن، مقدار goodput شبکه نیز در بسیاری از موارد، همان 100kbps می باشد که مشخص کردیم.

نتایج این آزمایش برای بار دوم نیز مشابه بار اول می باشد یعنی مقدار loss برابر با 0 بوده و مقدار goodput در اکثر موارد برابر با 100kbps می باشد.

(سوال 2)

مقدار loss و همچنین مقدار goodput مربوط به هر نرخ در تصاویر مربوط به هر نرخ مشخص می باشد :

نرخ : 1Mbps

```
X "host: h1"@mininet-vm
-
48.3s - sent: 6035 pkts, 1000.8 kbits/s
49.3s - sent: 6161 pkts, 1000.4 kbits/s
50.3s - sent: 6287 pkts, 1000.6 kbits/s
51.3s - sent: 6413 pkts, 1000.7 kbits/s
52.3s - sent: 6539 pkts, 1001.0 kbits/s
53.3s - sent: 6664 pkts, 999.8 kbits/s
54.3s - sent: 6790 pkts, 1000.7 kbits/s
55.3s - sent: 6916 pkts, 1000.3 kbits/s
56.3s - sent: 7042 pkts, 1001.4 kbits/s
57.3s - sent: 7168 pkts, 1000.6 kbits/s
58.3s - sent: 7294 pkts, 1000.2 kbits/s
59.4s - sent: 7420 pkts, 1004.0 kbits/s
60.4s - sent: 7546 pkts, 1001.0 kbits/s
61.4s - sent: 7672 pkts, 1000.3 kbits/s
62.4s - sent: 7798 pkts, 1000.3 kbits/s
63.4s - sent: 7924 pkts, 1000.7 kbits/s
64.4s - sent: 8049 pkts, 998.3 kbits/s
65.4s - sent: 8175 pkts, 1002.7 kbits/s
66.4s - sent: 8301 pkts, 1001.0 kbits/s
67.4s - sent: 8427 pkts, 1000.3 kbits/s
68.4s - sent: 8553 pkts, 1000.7 kbits/s
69.4s - sent: 8679 pkts, 999.8 kbits/s
^Cpackets sent = 8743, avg rate=1000.0kbps
root@mininet-vm:~/NetLab/HW6/lab6/udp#
```

```
X "host: h3"@mininet-vm
-
90.8s - received: 6035/ sent: 6035 pkts (loss 0.000%), 1000.4 kbit/s
91.8s - received: 6160/ sent: 6160 pkts (loss 0.000%), 999.6 kbit/s
92.8s - received: 6286/ sent: 6286 pkts (loss 0.000%), 1000.4 kbit/s
93.8s - received: 6411/ sent: 6411 pkts (loss 0.000%), 999.1 kbit/s
94.8s - received: 6537/ sent: 6537 pkts (loss 0.000%), 1001.3 kbit/s
95.8s - received: 6662/ sent: 6662 pkts (loss 0.000%), 998.9 kbit/s
96.8s - received: 6788/ sent: 6788 pkts (loss 0.000%), 1000.5 kbit/s
97.8s - received: 6914/ sent: 6914 pkts (loss 0.000%), 1000.8 kbit/s
98.8s - received: 7039/ sent: 7039 pkts (loss 0.000%), 999.1 kbit/s
99.8s - received: 7165/ sent: 7165 pkts (loss 0.000%), 999.8 kbit/s
100.8s - received: 7291/ sent: 7291 pkts (loss 0.000%), 1001.0 kbit/s
101.8s - received: 7416/ sent: 7416 pkts (loss 0.000%), 999.0 kbit/s
102.8s - received: 7542/ sent: 7542 pkts (loss 0.000%), 1000.4 kbit/s
103.8s - received: 7667/ sent: 7667 pkts (loss 0.000%), 999.8 kbit/s
104.8s - received: 7793/ sent: 7793 pkts (loss 0.000%), 999.7 kbit/s
105.8s - received: 7919/ sent: 7919 pkts (loss 0.000%), 1000.7 kbit/s
106.9s - received: 8044/ sent: 8044 pkts (loss 0.000%), 999.7 kbit/s
107.9s - received: 8170/ sent: 8170 pkts (loss 0.000%), 1000.4 kbit/s
108.9s - received: 8295/ sent: 8295 pkts (loss 0.000%), 999.8 kbit/s
109.9s - received: 8421/ sent: 8421 pkts (loss 0.000%), 1000.2 kbit/s
110.9s - received: 8547/ sent: 8547 pkts (loss 0.000%), 999.9 kbit/s
111.9s - received: 8673/ sent: 8673 pkts (loss 0.000%), 1000.6 kbit/s
packet received = 8743 / 8743 sent: 0.000% loss
```

نرخ : 10Mbps

```
X "host: h1"@mininet-vm
8.0s - sent: 10014 pkts, 10004.7 kbytes/s
9.0s - sent: 11266 pkts, 10008.3 kbytes/s
10.0s - sent: 12518 pkts, 10009.5 kbytes/s
11.0s - sent: 13770 pkts, 10008.2 kbytes/s
12.0s - sent: 15021 pkts, 10005.0 kbytes/s
13.0s - sent: 16273 pkts, 10008.7 kbytes/s
14.0s - sent: 17524 pkts, 10003.4 kbytes/s
15.0s - sent: 18776 pkts, 10010.6 kbytes/s
16.0s - sent: 20027 pkts, 10005.8 kbytes/s
17.0s - sent: 21278 pkts, 10005.8 kbytes/s
18.0s - sent: 22530 pkts, 10007.5 kbytes/s
19.0s - sent: 23781 pkts, 10007.2 kbytes/s
20.0s - sent: 25032 pkts, 10003.0 kbytes/s
21.0s - sent: 26284 pkts, 10009.1 kbytes/s
22.0s - sent: 27535 pkts, 10001.1 kbytes/s
23.0s - sent: 28787 pkts, 10011.5 kbytes/s
24.0s - sent: 30045 pkts, 10056.1 kbytes/s
25.0s - sent: 31297 pkts, 10007.8 kbytes/s
26.0s - sent: 32548 pkts, 10003.2 kbytes/s
27.0s - sent: 33800 pkts, 10008.0 kbytes/s
28.0s - sent: 35052 pkts, 10004.2 kbytes/s
29.0s - sent: 36303 pkts, 10005.3 kbytes/s
^Cpackets sent = 37058, avg rate=10000.4kbps
root@mininet-vm:~/NetLab/HW6/lab6/udp#
```

```
X "host: h3"@mininet-vm
74.7s - received: 10183/ sent: 10217 pkts (loss 0.333%), 9996.4 kbit/s
75.7s - received: 11435/ sent: 11469 pkts (loss 0.296%), 10007.5 kbit/s
76.7s - received: 12686/ sent: 12720 pkts (loss 0.267%), 10000.5 kbit/s
77.7s - received: 13937/ sent: 13971 pkts (loss 0.243%), 10001.6 kbit/s
78.7s - received: 15188/ sent: 15222 pkts (loss 0.223%), 9999.8 kbit/s
79.7s - received: 16439/ sent: 16473 pkts (loss 0.206%), 10004.2 kbit/s
80.7s - received: 17690/ sent: 17724 pkts (loss 0.192%), 9999.6 kbit/s
81.7s - received: 18941/ sent: 18975 pkts (loss 0.179%), 10000.7 kbit/s
82.7s - received: 20191/ sent: 20225 pkts (loss 0.168%), 9997.7 kbit/s
83.7s - received: 21443/ sent: 21477 pkts (loss 0.158%), 10008.9 kbit/s
84.7s - received: 22693/ sent: 22727 pkts (loss 0.150%), 9996.6 kbit/s
85.7s - received: 23945/ sent: 23979 pkts (loss 0.142%), 10006.3 kbit/s
86.7s - received: 25195/ sent: 25229 pkts (loss 0.135%), 9999.0 kbit/s
87.7s - received: 26446/ sent: 26480 pkts (loss 0.128%), 10003.8 kbit/s
88.7s - received: 27696/ sent: 27730 pkts (loss 0.123%), 9995.9 kbit/s
89.7s - received: 28948/ sent: 28982 pkts (loss 0.117%), 10009.8 kbit/s
90.7s - received: 30198/ sent: 30232 pkts (loss 0.112%), 9995.6 kbit/s
91.7s - received: 31450/ sent: 31484 pkts (loss 0.108%), 10005.5 kbit/s
92.7s - received: 32700/ sent: 32734 pkts (loss 0.104%), 9998.6 kbit/s
93.7s - received: 33951/ sent: 33985 pkts (loss 0.100%), 10003.5 kbit/s
94.7s - received: 35202/ sent: 35236 pkts (loss 0.096%), 9998.8 kbit/s
95.7s - received: 36454/ sent: 36488 pkts (loss 0.093%), 10004.0 kbit/s
packet received = 37024 / 37058 sent: 0.092% loss
```

نرخ : 100Mbps

X "host: h1"@mininet-vm

— □ ×

```
23.0s - sent:287551 pkts, 99895.3 kbytes/s
24.0s - sent:300112 pkts, 100458.2 kbytes/s
25.0s - sent:312603 pkts, 99898.9 kbytes/s
26.0s - sent:325106 pkts, 99918.4 kbytes/s
27.0s - sent:337610 pkts, 99974.9 kbytes/s
28.1s - sent:350177 pkts, 100459.5 kbytes/s
29.1s - sent:362677 pkts, 99916.5 kbytes/s
30.1s - sent:375249 pkts, 100542.7 kbytes/s
31.1s - sent:387741 pkts, 99732.6 kbytes/s
32.1s - sent:400303 pkts, 100495.0 kbytes/s
33.1s - sent:412790 pkts, 99825.6 kbytes/s
34.1s - sent:425303 pkts, 100024.6 kbytes/s
35.1s - sent:437853 pkts, 100379.9 kbytes/s
36.1s - sent:450414 pkts, 100458.8 kbytes/s
37.1s - sent:462925 pkts, 100062.0 kbytes/s
38.1s - sent:475427 pkts, 99954.6 kbytes/s
39.1s - sent:487994 pkts, 100469.0 kbytes/s
40.1s - sent:500457 pkts, 99607.0 kbytes/s
41.1s - sent:513018 pkts, 100422.0 kbytes/s
42.1s - sent:525517 pkts, 99924.4 kbytes/s
43.1s - sent:538013 pkts, 99952.6 kbytes/s
44.1s - sent:550509 pkts, 99546.5 kbytes/s
^Cpackets sent = 557540, avg rate=3484.9kbps
root@mininet-vm:~/NetLab/HW6/lab6/udp# █
```

X "host: h3"@mininet-vm

— □ ×

```
31.3s - received: 9380/ sent:292484 pkts (loss 96.793%), 2879.1 kbit/s
32.3s - received: 9740/ sent:304974 pkts (loss 96.806%), 2879.9 kbit/s
33.3s - received: 10100/ sent:317479 pkts (loss 96.819%), 2878.2 kbit/s
34.3s - received: 10461/ sent:330000 pkts (loss 96.830%), 2880.4 kbit/s
35.3s - received: 10821/ sent:342481 pkts (loss 96.840%), 2871.4 kbit/s
36.3s - received: 11182/ sent:355010 pkts (loss 96.850%), 2880.0 kbit/s
37.3s - received: 11542/ sent:367487 pkts (loss 96.859%), 2879.0 kbit/s
38.3s - received: 11902/ sent:380012 pkts (loss 96.868%), 2878.0 kbit/s
39.3s - received: 12262/ sent:392500 pkts (loss 96.876%), 2879.7 kbit/s
40.3s - received: 12622/ sent:404982 pkts (loss 96.883%), 2877.5 kbit/s
41.3s - received: 12983/ sent:417510 pkts (loss 96.890%), 2879.3 kbit/s
42.3s - received: 13343/ sent:429996 pkts (loss 96.897%), 2876.3 kbit/s
43.3s - received: 13702/ sent:442455 pkts (loss 96.903%), 2871.4 kbit/s
44.3s - received: 14062/ sent:454942 pkts (loss 96.909%), 2877.8 kbit/s
45.3s - received: 14422/ sent:467437 pkts (loss 96.915%), 2879.9 kbit/s
46.3s - received: 14782/ sent:479961 pkts (loss 96.920%), 2877.6 kbit/s
47.3s - received: 15142/ sent:492458 pkts (loss 96.925%), 2879.7 kbit/s
48.3s - received: 15502/ sent:504940 pkts (loss 96.930%), 2877.8 kbit/s
49.3s - received: 15863/ sent:517465 pkts (loss 96.934%), 2878.6 kbit/s
50.3s - received: 16223/ sent:529960 pkts (loss 96.939%), 2878.7 kbit/s
51.3s - received: 16583/ sent:542443 pkts (loss 96.943%), 2876.9 kbit/s
52.3s - received: 16944/ sent:554971 pkts (loss 96.947%), 2879.9 kbit/s
packet received = 17018 / 557530 sent: 96.948% loss █
```

نرخ : 1Gbps

X "host: h1"@mininet-vm

```
40.1s - sent:4941769 pkts, 987327.1 kbytes/s
41.1s - sent:5065350 pkts, 988648.0 kbytes/s
42.1s - sent:5188818 pkts, 987741.2 kbytes/s
43.1s - sent:5312237 pkts, 987331.3 kbytes/s
44.1s - sent:5435846 pkts, 988407.5 kbytes/s
45.1s - sent:5559278 pkts, 987347.5 kbytes/s
46.1s - sent:5682794 pkts, 988112.2 kbytes/s
47.1s - sent:5806418 pkts, 988388.0 kbytes/s
48.1s - sent:5929854 pkts, 987485.9 kbytes/s
49.1s - sent:6053363 pkts, 988071.1 kbytes/s
50.1s - sent:6176785 pkts, 987375.1 kbytes/s
51.1s - sent:6299981 pkts, 984428.0 kbytes/s
52.1s - sent:6423902 pkts, 991079.5 kbytes/s
53.1s - sent:6547479 pkts, 988616.0 kbytes/s
54.1s - sent:6668316 pkts, 965523.8 kbytes/s
55.1s - sent:6794462 pkts, 1008978.4 kbytes/s
56.1s - sent:6917961 pkts, 987957.4 kbytes/s
57.1s - sent:7041115 pkts, 985228.9 kbytes/s
58.1s - sent:7165055 pkts, 991052.2 kbytes/s
59.1s - sent:7288608 pkts, 988354.0 kbytes/s
60.1s - sent:7412021 pkts, 987304.0 kbytes/s
61.1s - sent:7535404 pkts, 987063.1 kbytes/s
^Cpackets sent = 7588361, avg rate=8978.9kbps
root@mininet-vm:~/NetLab/HW6/lab6/udp#
```

X "host: h3"@mininet-vm

```
43.9s - received: 15188/ sent:4871266 pkts (loss 99.688%), 2879.1 kbit/s
44.9s - received: 15548/ sent:4995086 pkts (loss 99.689%), 2879.0 kbit/s
45.9s - received: 15908/ sent:5118511 pkts (loss 99.689%), 2876.0 kbit/s
46.9s - received: 16268/ sent:5315848 pkts (loss 99.694%), 2879.8 kbit/s
47.9s - received: 16628/ sent:5365318 pkts (loss 99.690%), 2878.8 kbit/s
48.9s - received: 16988/ sent:5488801 pkts (loss 99.690%), 2879.2 kbit/s
49.9s - received: 17348/ sent:5611893 pkts (loss 99.691%), 2878.3 kbit/s
50.9s - received: 17709/ sent:5735642 pkts (loss 99.691%), 2880.2 kbit/s
51.9s - received: 18069/ sent:5858970 pkts (loss 99.692%), 2879.0 kbit/s
52.9s - received: 18429/ sent:5982413 pkts (loss 99.692%), 2873.2 kbit/s
53.9s - received: 18790/ sent:6106131 pkts (loss 99.692%), 2885.6 kbit/s
54.9s - received: 19150/ sent:6229508 pkts (loss 99.693%), 2874.9 kbit/s
55.9s - received: 19511/ sent:6353321 pkts (loss 99.693%), 2881.4 kbit/s
56.9s - received: 19872/ sent:6477060 pkts (loss 99.693%), 2881.2 kbit/s
57.9s - received: 20233/ sent:6600817 pkts (loss 99.693%), 2880.5 kbit/s
58.9s - received: 20593/ sent:6724198 pkts (loss 99.694%), 2879.4 kbit/s
59.9s - received: 20953/ sent:6847555 pkts (loss 99.694%), 2877.9 kbit/s
60.9s - received: 21314/ sent:6971363 pkts (loss 99.694%), 2880.0 kbit/s
61.9s - received: 21674/ sent:7094778 pkts (loss 99.695%), 2879.4 kbit/s
62.9s - received: 22034/ sent:7218133 pkts (loss 99.695%), 2879.3 kbit/s
63.9s - received: 22394/ sent:7341512 pkts (loss 99.695%), 2877.5 kbit/s
64.9s - received: 22754/ sent:7464949 pkts (loss 99.695%), 2872.8 kbit/s
packet received = 23114 / 7588361 sent: 99.695% loss
]
```

همانطور که از نتایج مشخص می باشد مشاهده می کنیم که در این پروتکل به ازای نرخ 10Mbps به بالاتر مقدار loss جهش زیادی پیدا می کند و طبق فایل topo.py به نظر می رسد که این افزایش مقدار loss به دلیل اشباع لینک بین r1 و sw2 است. با افزایش نرخ ارسال، میزان ترافیک در این لینک نیز افزایش می یابد و در نهایت از ظرفیت آن فراتر می رود. این امر منجر به اتلاف بسته می شود، زیرا روتر قادر به پردازش تمام بسته های ورودی نیست.

برای کاهش این مشکل، می توان به عنوان مثال، پهنهای باند لینک r1 و sw2 را افزایش داد که به آن اجازه می دهد تا ترافیک بیشتری را مدیریت کند و اتلاف بسته را کاهش دهد. برای این کار می توان در فایل topo.py یک متغیر به اسم res ایجاد کرد و برای آن مقدار bw را set کرد و مقدار آن را بیشتر کنیم تا این مشکل کاهش پیدا کند :

```
res = net.addLink( sw2, r1, intfName2='r1-eth1' )
```

```
info( '*** Starting network\n' )
res.intf1.config(bw=30)
```

(سوال 3)

پس از ایجاد سرور و کلاینت بر روی هاست های h1 , h3 برای آزمایش مرتبه اول نتایج مطابق تصویر زیر می باشد :

X "host: h3"@mininet-vm

```
root@mininet-vm:~/NetLab/HW6/lab6/tcp# ./tcpserver 10001
Handling client 10.10.0.1
with child process: 94829
^C
root@mininet-vm:~/NetLab/HW6/lab6/tcp# █
```

X "host: h1"@mininet-vm

```
65.6: 2802.0kbps avg ( 2804.7[inst], 2810.7[mov,avg]) cwnd 991 rtt3996.4ms
66.6: 2802.1kbps avg ( 2806.7[inst], 2810.3[mov,avg]) cwnd 991 rtt3997.3ms
67.7: 2802.0kbps avg ( 2794.5[inst], 2808.8[mov,avg]) cwnd 992 rtt3451.9ms
68.7: 2802.1kbps avg ( 2807.5[inst], 2808.6[mov,avg]) cwnd 992 rtt3999.3ms
69.7: 2802.1kbps avg ( 2803.1[inst], 2808.1[mov,avg]) cwnd 992 rtt4000.8ms
70.7: 2802.0kbps avg ( 2797.2[inst], 2807.0[mov,avg]) cwnd 992 rtt3998.1ms
71.8: 2802.1kbps avg ( 2805.4[inst], 2806.8[mov,avg]) cwnd 993 rtt4006.0ms
72.8: 2802.0kbps avg ( 2793.8[inst], 2805.5[mov,avg]) cwnd 993 rtt4005.2ms
73.8: 2802.0kbps avg ( 2807.3[inst], 2805.7[mov,avg]) cwnd 993 rtt4004.9ms
74.8: 2802.0kbps avg ( 2796.5[inst], 2804.8[mov,avg]) cwnd 993 rtt4005.3ms
75.9: 2802.2kbps avg ( 2816.7[inst], 2806.0[mov,avg]) cwnd 994 rtt4008.5ms
76.9: 2802.0kbps avg ( 2792.7[inst], 2804.6[mov,avg]) cwnd 994 rtt4006.6ms
77.9: 2802.1kbps avg ( 2806.9[inst], 2804.9[mov,avg]) cwnd 994 rtt4008.2ms
78.9: 2802.0kbps avg ( 2794.5[inst], 2803.8[mov,avg]) cwnd 994 rtt4005.3ms
80.0: 2802.1kbps avg ( 2807.2[inst], 2804.2[mov,avg]) cwnd 995 rtt4012.5ms
81.0: 2802.0kbps avg ( 2794.9[inst], 2803.2[mov,avg]) cwnd 995 rtt4013.2ms
82.0: 2802.0kbps avg ( 2805.0[inst], 2803.4[mov,avg]) cwnd 995 rtt4012.5ms
83.0: 2802.1kbps avg ( 2806.6[inst], 2803.7[mov,avg]) cwnd 995 rtt4014.0ms
84.1: 2802.1kbps avg ( 2806.2[inst], 2804.0[mov,avg]) cwnd 996 rtt4016.6ms
85.1: 2802.0kbps avg ( 2795.6[inst], 2803.1[mov,avg]) cwnd 996 rtt4014.3ms
86.1: 2802.1kbps avg ( 2806.9[inst], 2803.5[mov,avg]) cwnd 996 rtt4016.3ms
87.1: 2802.1kbps avg ( 2805.3[inst], 2803.7[mov,avg]) cwnd 996 rtt4016.7ms
88.1: 2802.1kbps avg ( 2805.3[inst], 2803.7[mov,avg]) cwnd 997 rtt4019.8ms
root@mininet-vm:~/NetLab/HW6/lab6/tcp# █
```

X "host: h3"@mininet-vm

```
root@mininet-vm:~/NetLab/HW6/lab6/tcp# ./tcpserver 10001
Handling client 10.10.0.1
with child process: 93614
```

X "host: h1"@mininet-vm

```
105.4: 2801.5kbps avg ( 2807.6[inst], 2822.1[mov,avg]) cwnd 990 rtt3990.6ms
106.4: 2801.5kbps avg ( 2794.5[inst], 2819.3[mov,avg]) cwnd 991 rtt3996.9ms
107.4: 2801.5kbps avg ( 2806.3[inst], 2818.0[mov,avg]) cwnd 991 rtt3781.2ms
108.4: 2801.5kbps avg ( 2804.4[inst], 2816.7[mov,avg]) cwnd 991 rtt3997.3ms
109.5: 2801.5kbps avg ( 2794.5[inst], 2814.4[mov,avg]) cwnd 991 rtt3996.9ms
110.5: 2801.5kbps avg ( 2805.6[inst], 2813.6[mov,avg]) cwnd 992 rtt3998.0ms
111.5: 2801.6kbps avg ( 2807.9[inst], 2813.0[mov,avg]) cwnd 992 rtt4000.4ms
112.5: 2801.6kbps avg ( 2804.5[inst], 2812.1[mov,avg]) cwnd 992 rtt4000.6ms
113.6: 2801.5kbps avg ( 2795.6[inst], 2810.5[mov,avg]) cwnd 992 rtt3998.0ms
114.6: 2801.5kbps avg ( 2794.3[inst], 2808.9[mov,avg]) cwnd 993 rtt4004.8ms
115.6: 2801.5kbps avg ( 2806.6[inst], 2808.6[mov,avg]) cwnd 993 rtt4004.5ms
116.6: 2801.6kbps avg ( 2807.8[inst], 2808.6[mov,avg]) cwnd 993 rtt4004.5ms
117.7: 2801.5kbps avg ( 2793.4[inst], 2807.0[mov,avg]) cwnd 993 rtt4005.1ms
118.7: 2801.5kbps avg ( 2806.1[inst], 2806.9[mov,avg]) cwnd 994 rtt4005.5ms
119.7: 2801.6kbps avg ( 2807.4[inst], 2807.0[mov,avg]) cwnd 994 rtt4007.8ms
120.7: 2801.6kbps avg ( 2805.7[inst], 2806.9[mov,avg]) cwnd 994 rtt4009.4ms
121.8: 2801.6kbps avg ( 2795.0[inst], 2805.7[mov,avg]) cwnd 994 rtt4006.4ms
122.8: 2801.5kbps avg ( 2794.8[inst], 2804.6[mov,avg]) cwnd 995 rtt4014.6ms
123.8: 2801.5kbps avg ( 2805.7[inst], 2804.7[mov,avg]) cwnd 995 rtt4014.2ms
124.8: 2801.6kbps avg ( 2806.0[inst], 2804.8[mov,avg]) cwnd 995 rtt4013.5ms
125.9: 2801.5kbps avg ( 2795.7[inst], 2803.9[mov,avg]) cwnd 995 rtt4013.9ms
126.9: 2801.6kbps avg ( 2805.6[inst], 2804.1[mov,avg]) cwnd 996 rtt4013.8ms
^C
```

```
root@mininet-vm:~/NetLab/HW6/lab6/tcp#
```

همانطور که در تصویر مشخص می باشد در آزمایش مرتبه اول بعد از گذشت تقریبا 88 ثانیه مقدار goodput برای TCP 2802 kbps می باشد و این آزمایش را برای بار دوم نیز تکرار می کنیم که بعد از گذشت تقریبا 127 ثانیه مقدار 2801 kbps برای TCP برابر است با

می توان از میانگین گرفتن این دو عدد مقدار goodput را تقریبا بدست آورد.

#### (4) سوال

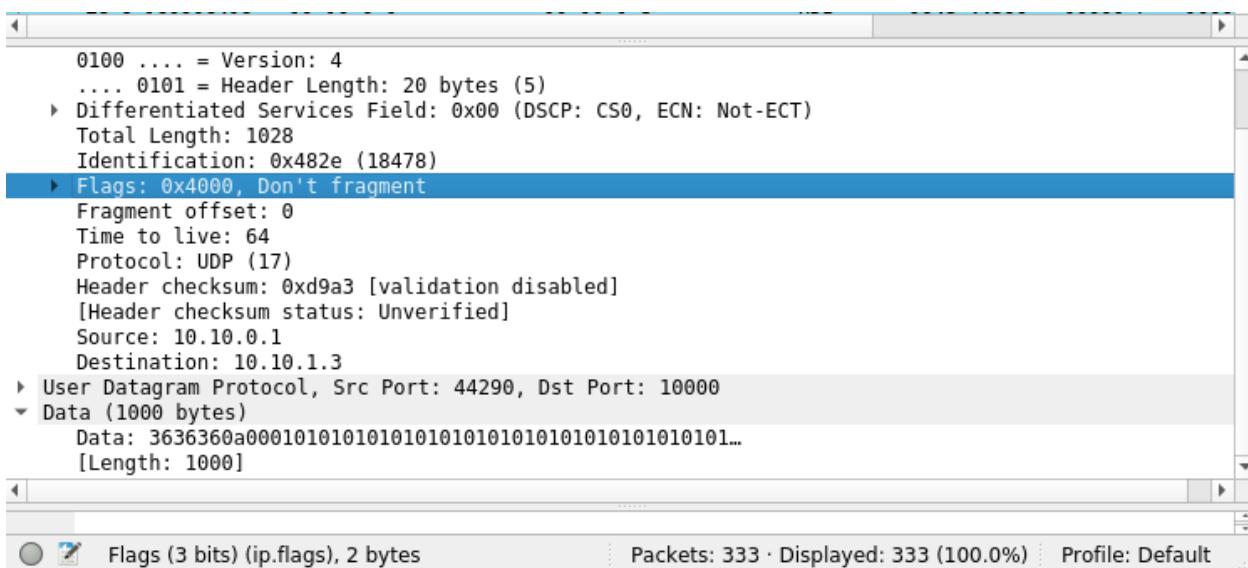
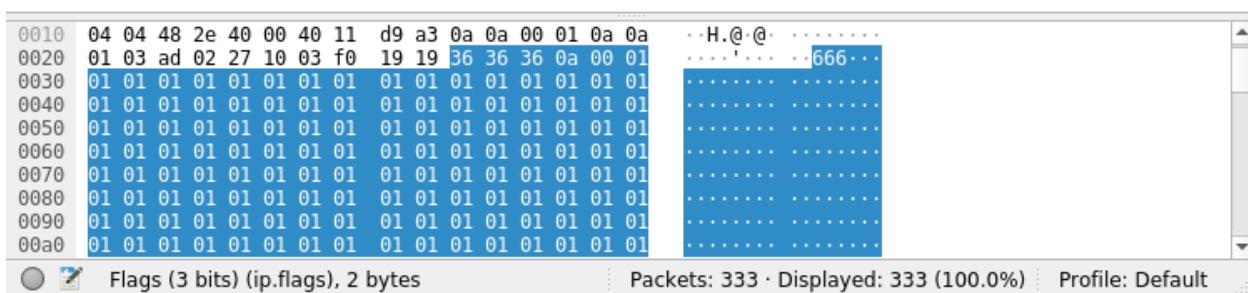
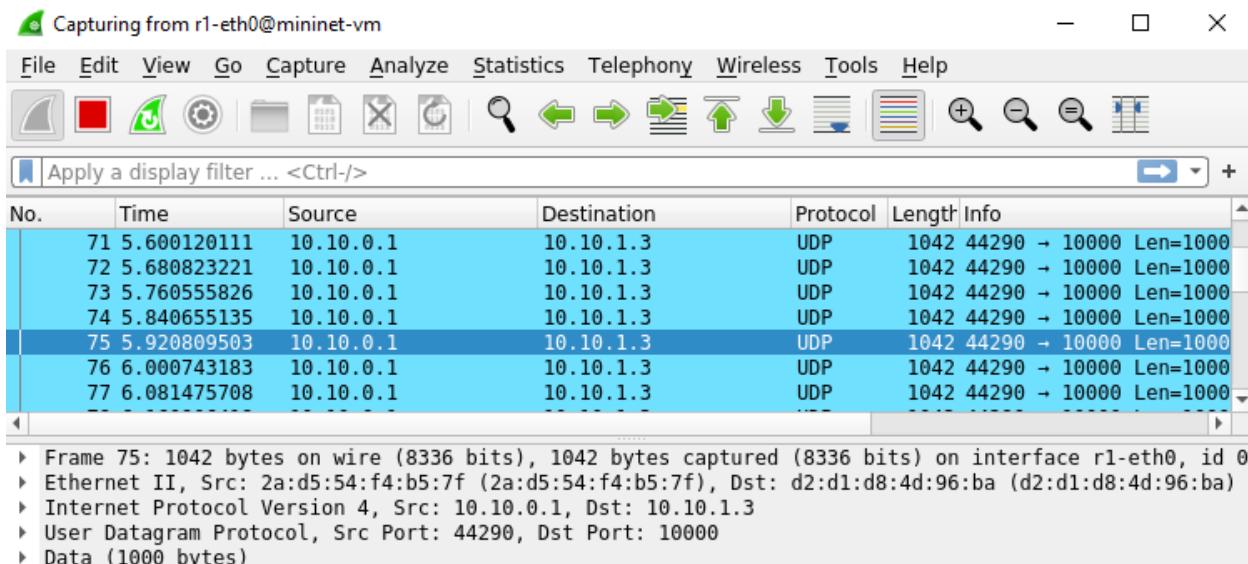
در این بخش، ابتدا در فایل پایتون topo.py ، پهنای باند اینترفیس eth1 از روتر را به 3 Mbps محدود می کنیم.

```
info( '*** Starting network\n')
res.intf1.config(bw=3)
```

سپس توپولوژی را clean کرده و دوباره اجرا می کنیم.

حال برای این سوال ابتدا udpserver را بر روی h3 و udpclient را بر روی h1 و بر روی پورت 10000، همانند حالت قبل ایجاد می کنیم.

در ادامه wireshark را برای r1-eth0 باز می کنیم تا مقادیر خواسته شده را بررسی کنیم و درستی آن ها را مشخص کنیم :



```

Frame 75: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits) on interface r1-eth0, id 1
  ▶ Interface id: 0 (r1-eth0)
  ▶ Encapsulation type: Ethernet (1)
  ▶ Arrival Time: Jan 4, 2024 16:33:57.056165776 PST
  ▶ [Time shift for this packet: 0.000000000 seconds]
  ▶ Epoch Time: 1704414837.056165776 seconds
  ▶ [Time delta from previous captured frame: 0.080154368 seconds]
  ▶ [Time delta from previous displayed frame: 0.080154368 seconds]
  ▶ [Time since reference or first frame: 5.920809503 seconds]
  ▶ Frame Number: 75
  ▶ Frame Length: 1042 bytes (8336 bits)
  ▶ Capture Length: 1042 bytes (8336 bits)
  ▶ [Frame is marked: False]
  ▶ [Frame is ignored: False]
  ▶ [Protocols in frame: eth:ethertype:ip:udp:data]
  ▶ [Coloring Rule Name: UDP]
  ▶ [Coloring Rule String: udp]

Destination (ip.dst), 4 bytes
  Packets: 333 · Displayed: 333 (100.0%) · Profile: Default

```

همانطور که در عکس ها مشاهده می شود هدر UDP به میزان 8 بایت، هدر IP به مقدار 20 بایت و هدر Ethernet به میزان 14 بایت و اندازه داده های Application برابر با 1000 بایت می باشد که در مجموع برابر است با 1042 بایت.

(5) سوال

به لحاظ تئوری، حداقل مقدار قابل دستیابی برای گزینه های کاربردی یا همان صورت زیر می باشد :

$$2 * (1000 / 1042) = 2.879 \text{ Mbps}$$

## سوال 6

مقدار loss برابر است با 0 و مقدار goodput طی آخرین ثانیه نیز برابر با 100 Kbit/s می باشد.

```
X "host: h1"@mininet-vm
-
41.6s - sent: 521 pkts, 100.1 kbits/s
42.6s - sent: 534 pkts, 100.0 kbits/s
43.7s - sent: 547 pkts, 100.1 kbits/s
44.7s - sent: 560 pkts, 99.9 kbits/s
45.8s - sent: 573 pkts, 100.1 kbits/s
46.8s - sent: 586 pkts, 100.0 kbits/s
47.8s - sent: 599 pkts, 100.1 kbits/s
48.9s - sent: 612 pkts, 99.9 kbits/s
49.9s - sent: 625 pkts, 100.1 kbits/s
51.0s - sent: 638 pkts, 100.0 kbits/s
52.0s - sent: 651 pkts, 99.9 kbits/s
53.0s - sent: 664 pkts, 100.2 kbits/s
54.1s - sent: 677 pkts, 100.0 kbits/s
55.1s - sent: 690 pkts, 100.0 kbits/s
56.2s - sent: 703 pkts, 100.0 kbits/s
57.2s - sent: 716 pkts, 100.0 kbits/s
58.2s - sent: 729 pkts, 100.0 kbits/s
59.3s - sent: 742 pkts, 100.0 kbits/s
60.3s - sent: 755 pkts, 100.0 kbits/s
61.4s - sent: 768 pkts, 100.0 kbits/s
62.4s - sent: 781 pkts, 100.1 kbits/s
63.4s - sent: 794 pkts, 100.0 kbits/s
^Cpackets sent = 802, avg rate= 100.1kbps
root@mininet-vm:~/NetLab/HW6/lab6/udp#
```

```
X "host: h3"@mininet-vm
-
44.3s - received: 534/ sent: 534 pkts (loss 0.000%), 100.0 kbit/s
45.4s - received: 547/ sent: 547 pkts (loss 0.000%), 100.1 kbit/s
46.4s - received: 560/ sent: 560 pkts (loss 0.000%), 99.9 kbit/s
47.4s - received: 573/ sent: 573 pkts (loss 0.000%), 100.0 kbit/s
48.5s - received: 586/ sent: 586 pkts (loss 0.000%), 100.0 kbit/s
49.5s - received: 599/ sent: 599 pkts (loss 0.000%), 100.0 kbit/s
50.6s - received: 612/ sent: 612 pkts (loss 0.000%), 99.9 kbit/s
51.6s - received: 625/ sent: 625 pkts (loss 0.000%), 100.1 kbit/s
52.6s - received: 638/ sent: 638 pkts (loss 0.000%), 100.0 kbit/s
53.7s - received: 651/ sent: 651 pkts (loss 0.000%), 99.9 kbit/s
54.7s - received: 664/ sent: 664 pkts (loss 0.000%), 100.1 kbit/s
55.8s - received: 677/ sent: 677 pkts (loss 0.000%), 100.0 kbit/s
56.8s - received: 690/ sent: 690 pkts (loss 0.000%), 100.0 kbit/s
57.8s - received: 703/ sent: 703 pkts (loss 0.000%), 100.0 kbit/s
58.9s - received: 716/ sent: 716 pkts (loss 0.000%), 100.1 kbit/s
59.9s - received: 729/ sent: 729 pkts (loss 0.000%), 100.0 kbit/s
61.0s - received: 742/ sent: 742 pkts (loss 0.000%), 100.0 kbit/s
62.0s - received: 755/ sent: 755 pkts (loss 0.000%), 100.1 kbit/s
63.0s - received: 768/ sent: 768 pkts (loss 0.000%), 99.9 kbit/s
64.1s - received: 781/ sent: 781 pkts (loss 0.000%), 100.1 kbit/s
65.1s - received: 794/ sent: 794 pkts (loss 0.000%), 100.0 kbit/s
packet received = 802 / 802 sent: 0.000% loss
^C
root@mininet-vm:~/NetLab/HW6/lab6/udp#
```

سؤال 7

نرخ 3Mbps

```
X "host: h1"@mininet-vm
-
107.3s - sent: 40226 pkts, 3004.0 kbytes/s
108.3s - sent: 40602 pkts, 3002.0 kbytes/s
109.3s - sent: 40977 pkts, 2999.5 kbytes/s
110.3s - sent: 41353 pkts, 3001.3 kbytes/s
111.3s - sent: 41729 pkts, 3001.5 kbytes/s
112.3s - sent: 42105 pkts, 3002.4 kbytes/s
113.3s - sent: 42481 pkts, 3002.7 kbytes/s
114.3s - sent: 42857 pkts, 3003.7 kbytes/s
115.3s - sent: 43232 pkts, 2999.6 kbytes/s
116.3s - sent: 43608 pkts, 3000.6 kbytes/s
117.3s - sent: 43984 pkts, 3003.7 kbytes/s
118.3s - sent: 44360 pkts, 3001.9 kbytes/s
119.3s - sent: 44736 pkts, 3002.9 kbytes/s
120.3s - sent: 45112 pkts, 3000.5 kbytes/s
121.3s - sent: 45488 pkts, 3003.5 kbytes/s
122.3s - sent: 45864 pkts, 2999.7 kbytes/s
123.3s - sent: 46240 pkts, 3004.4 kbytes/s
124.3s - sent: 46615 pkts, 2999.8 kbytes/s
125.3s - sent: 46991 pkts, 3002.2 kbytes/s
126.3s - sent: 47367 pkts, 3001.7 kbytes/s
127.3s - sent: 47743 pkts, 3000.6 kbytes/s
128.3s - sent: 48119 pkts, 3001.7 kbytes/s
^Cpackets sent = 48238, avg rate=2936.5kbps
root@mininet-vm:~/NetLab/HW6/lab6/udp#
```

```
X "host: h3"@mininet-vm
-
122.4s - received: 39621/ sent: 40255 pkts (loss 1.575%), 2879.3 kbit/s
123.4s - received: 39982/ sent: 40631 pkts (loss 1.597%), 2879.9 kbit/s
124.4s - received: 40342/ sent: 41006 pkts (loss 1.619%), 2879.9 kbit/s
125.4s - received: 40702/ sent: 41380 pkts (loss 1.638%), 2878.1 kbit/s
126.4s - received: 41062/ sent: 41755 pkts (loss 1.660%), 2873.2 kbit/s
127.4s - received: 41422/ sent: 42130 pkts (loss 1.681%), 2873.8 kbit/s
128.4s - received: 41783/ sent: 42506 pkts (loss 1.701%), 2880.7 kbit/s
129.4s - received: 42144/ sent: 42883 pkts (loss 1.723%), 2879.2 kbit/s
130.5s - received: 42504/ sent: 43258 pkts (loss 1.743%), 2878.9 kbit/s
131.5s - received: 42864/ sent: 43634 pkts (loss 1.765%), 2879.1 kbit/s
132.5s - received: 43224/ sent: 44009 pkts (loss 1.784%), 2877.8 kbit/s
133.5s - received: 43585/ sent: 44385 pkts (loss 1.802%), 2880.8 kbit/s
134.5s - received: 43945/ sent: 44760 pkts (loss 1.821%), 2876.0 kbit/s
135.5s - received: 44305/ sent: 45135 pkts (loss 1.839%), 2879.4 kbit/s
136.5s - received: 44665/ sent: 45511 pkts (loss 1.859%), 2878.8 kbit/s
137.5s - received: 45026/ sent: 45887 pkts (loss 1.876%), 2880.0 kbit/s
138.5s - received: 45386/ sent: 46262 pkts (loss 1.894%), 2875.8 kbit/s
139.5s - received: 45746/ sent: 46637 pkts (loss 1.911%), 2879.9 kbit/s
140.5s - received: 46106/ sent: 47012 pkts (loss 1.927%), 2878.3 kbit/s
141.5s - received: 46466/ sent: 47388 pkts (loss 1.946%), 2879.8 kbit/s
142.5s - received: 46827/ sent: 47764 pkts (loss 1.962%), 2879.9 kbit/s
143.5s - received: 47187/ sent: 48139 pkts (loss 1.978%), 2880.0 kbit/s
packet received = 47282 / 48238 sent: 1.982% loss
```

مقدار loss برابر با 1.9 درصد و مقدار goodput برابر با 2880 کیلو بایت بر ثانیه است که بسیار نزدیک به مقدار 2879 است که به صورت تئوری محاسبه کردیم.

: 10 Mbps نرخ

```
X "host: h1"@mininet-vm
-
□ ×

80.1s - sent:100124 pkts, 10014.5 kbytes/s
81.1s - sent:101374 pkts, 9999.6 kbytes/s
82.1s - sent:102626 pkts, 10006.1 kbytes/s
83.1s - sent:103877 pkts, 10006.1 kbytes/s
84.1s - sent:105128 pkts, 10007.1 kbytes/s
85.1s - sent:106379 pkts, 9999.9 kbytes/s
86.1s - sent:107631 pkts, 10009.4 kbytes/s
87.1s - sent:108883 pkts, 10007.9 kbytes/s
88.1s - sent:110134 pkts, 10007.7 kbytes/s
89.1s - sent:111385 pkts, 10006.6 kbytes/s
90.1s - sent:112637 pkts, 9999.1 kbytes/s
91.1s - sent:113889 pkts, 10015.3 kbytes/s
92.1s - sent:115141 pkts, 10007.7 kbytes/s
93.1s - sent:116392 pkts, 10004.6 kbytes/s
94.1s - sent:117643 pkts, 10005.9 kbytes/s
95.1s - sent:118895 pkts, 10006.2 kbytes/s
96.1s - sent:120147 pkts, 10008.4 kbytes/s
97.1s - sent:121398 pkts, 10005.5 kbytes/s
98.1s - sent:122650 pkts, 10010.5 kbytes/s
99.1s - sent:123901 pkts, 10005.0 kbytes/s
100.1s - sent:125152 pkts, 10005.1 kbytes/s
101.1s - sent:126402 pkts, 9942.6 kbytes/s
^Cpackets sent = 127157, avg rate=9960.1kbps
root@mininet-vm:~/NetLab/HW6/lab6/udp# █
```

```
X "host: h3"@mininet-vm
-
□ ×

90.0s - received: 28820/ sent: 96679 pkts (loss 70.190%), 2879.1 kbit/s
91.0s - received: 29180/ sent: 97930 pkts (loss 70.203%), 2876.4 kbit/s
92.0s - received: 29540/ sent: 99181 pkts (loss 70.216%), 2878.4 kbit/s
93.0s - received: 29900/ sent:100432 pkts (loss 70.229%), 2879.8 kbit/s
94.0s - received: 30260/ sent:101684 pkts (loss 70.241%), 2878.6 kbit/s
95.0s - received: 30620/ sent:102936 pkts (loss 70.253%), 2877.6 kbit/s
96.0s - received: 30981/ sent:104189 pkts (loss 70.265%), 2879.5 kbit/s
97.0s - received: 31341/ sent:105439 pkts (loss 70.276%), 2879.8 kbit/s
98.0s - received: 31701/ sent:106690 pkts (loss 70.287%), 2877.7 kbit/s
99.0s - received: 32062/ sent:107944 pkts (loss 70.298%), 2881.7 kbit/s
100.0s - received: 32422/ sent:109194 pkts (loss 70.308%), 2878.7 kbit/s
101.0s - received: 32782/ sent:110445 pkts (loss 70.318%), 2878.4 kbit/s
102.0s - received: 33142/ sent:111695 pkts (loss 70.328%), 2875.3 kbit/s
103.0s - received: 33502/ sent:112947 pkts (loss 70.338%), 2876.5 kbit/s
104.0s - received: 33862/ sent:114197 pkts (loss 70.348%), 2877.2 kbit/s
105.0s - received: 34222/ sent:115449 pkts (loss 70.357%), 2879.3 kbit/s
106.0s - received: 34583/ sent:116705 pkts (loss 70.367%), 2881.2 kbit/s
107.0s - received: 34943/ sent:117955 pkts (loss 70.376%), 2877.8 kbit/s
108.0s - received: 35303/ sent:119206 pkts (loss 70.385%), 2878.5 kbit/s
109.0s - received: 35663/ sent:120456 pkts (loss 70.393%), 2876.6 kbit/s
110.0s - received: 36023/ sent:121707 pkts (loss 70.402%), 2879.4 kbit/s
111.0s - received: 36378/ sent:122941 pkts (loss 70.410%), 2834.4 kbit/s
packet received = 36736 / 127157 sent: 71.110% loss
█
```

مقدار loss تقریبا برابر با 71 درصد است که به دلیل بیشتر بودن نرخ ارسال از ظرفیت شبکه است. همچنین مقدار goodput نیز تقریبا برابر با 2834 تا 2876 می باشد که بسیار نزدیک به مقدار تئوری می باشد.

## (سوال 8)

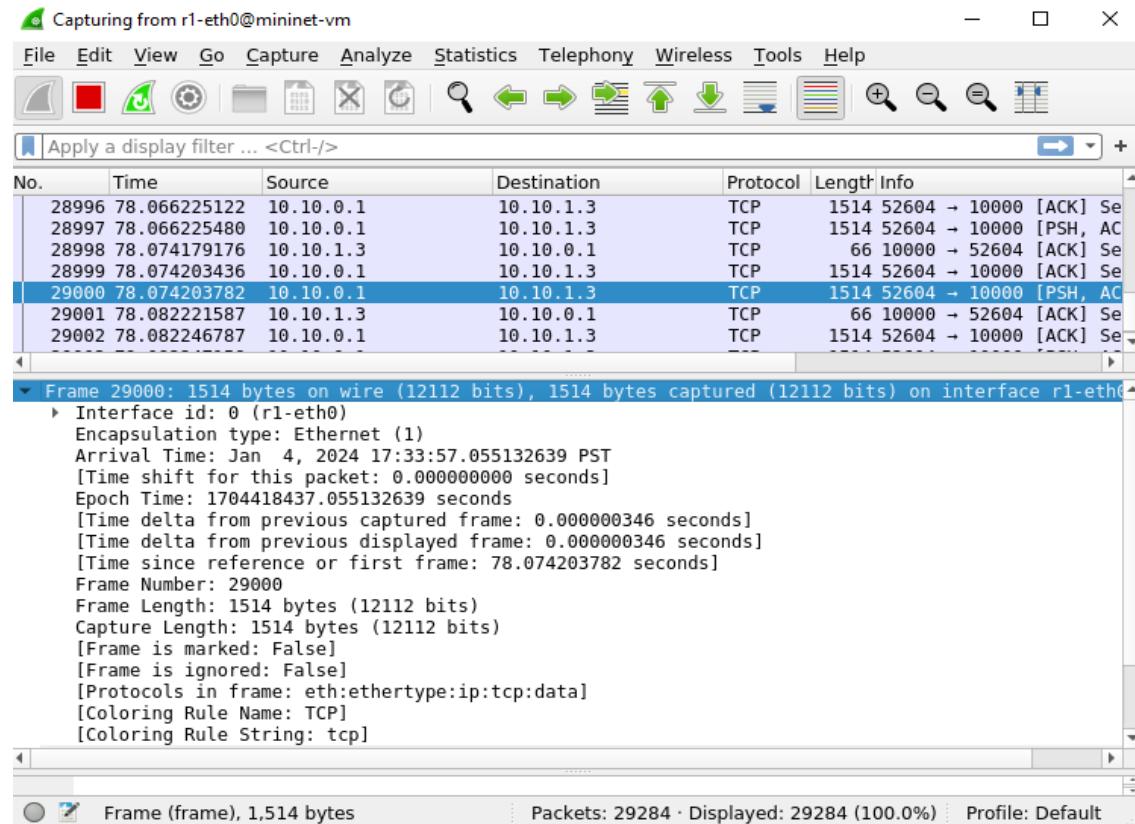
در ابتدا برای این سوال دستور گفته شده در صورت سوال را در h1 اجرا می کنیم.

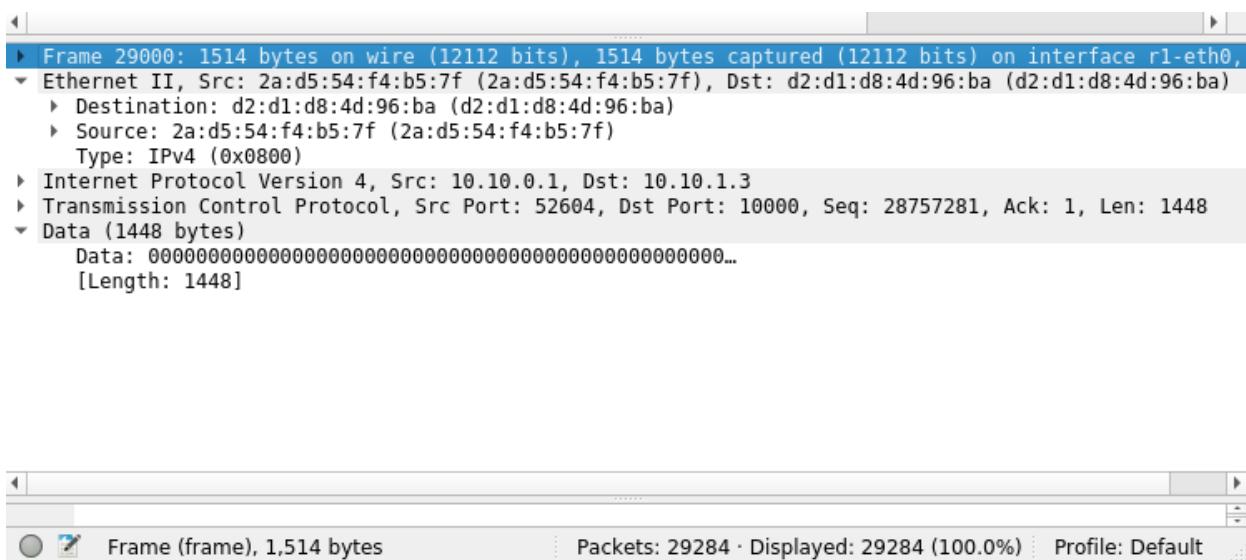
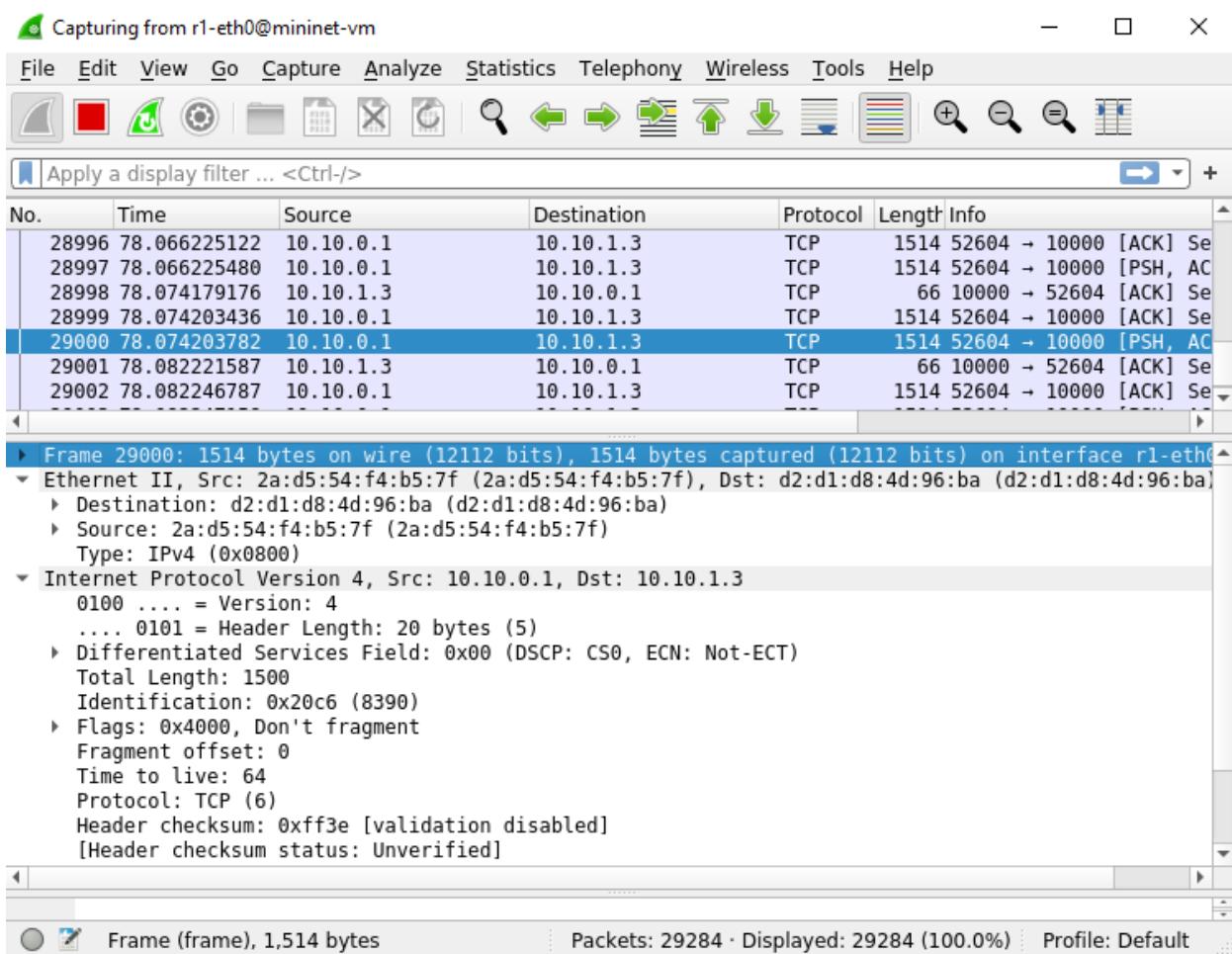
در ادامه یک ارتباط TCP بین h1 , h3 ایجاد می کنیم و ترافیک روی r1-eth0 را بررسی می کنیم که به صورت زیر می باشد :

```
X "host: h1"@mininet-vm
root@mininet-vm:~/NetLab/HW6/lab6# sudo apt-get install -y ethtool
Reading package lists... Done
Building dependency tree
Reading state information... Done
ethtool is already the newest version (1:5.4-1).
0 upgraded, 0 newly installed, 0 to remove and 84 not upgraded.
root@mininet-vm:~/NetLab/HW6/lab6# sudo ethtool --version
ethtool version 5.4
root@mininet-vm:~/NetLab/HW6/lab6# ethtool -k h1-eth0 tx off sg off tso off
ethtool: bad command line argument(s)
For more information run ethtool -h
root@mininet-vm:~/NetLab/HW6/lab6# ethtool -K h1-eth0 tx off sg off tso off
root@mininet-vm:~/NetLab/HW6/lab6# █
```

```

T "host: h1"@mininet-vm
40.5: 2586.4kbps avg ( 2783.8[inst], 2814.3[mov,avg]) cwnd 508 rtt2047.2ms
41.5: 2591.8kbps avg ( 2807.6[inst], 2813.6[mov,avg]) cwnd 508 rtt2044.3ms
42.5: 2596.7kbps avg ( 2794.4[inst], 2811.7[mov,avg]) cwnd 509 rtt2054.0ms
43.5: 2601.6kbps avg ( 2806.2[inst], 2811.1[mov,avg]) cwnd 509 rtt2051.9ms
44.5: 2606.6kbps avg ( 2815.7[inst], 2811.6[mov,avg]) cwnd 510 rtt2052.9ms
45.6: 2610.6kbps avg ( 2784.7[inst], 2808.9[mov,avg]) cwnd 510 rtt2054.3ms
46.6: 2614.8kbps avg ( 2805.3[inst], 2808.5[mov,avg]) cwnd 511 rtt2059.0ms
47.6: 2618.9kbps avg ( 2805.1[inst], 2808.2[mov,avg]) cwnd 511 rtt2058.1ms
48.6: 2622.9kbps avg ( 2807.4[inst], 2808.1[mov,avg]) cwnd 512 rtt2060.9ms
49.7: 2626.4kbps avg ( 2794.2[inst], 2806.7[mov,avg]) cwnd 512 rtt2065.8ms
50.7: 2630.1kbps avg ( 2806.8[inst], 2806.7[mov,avg]) cwnd 513 rtt2064.2ms
51.7: 2633.3kbps avg ( 2795.0[inst], 2805.6[mov,avg]) cwnd 513 rtt2067.9ms
52.7: 2636.7kbps avg ( 2804.4[inst], 2805.4[mov,avg]) cwnd 514 rtt2073.8ms
53.8: 2639.9kbps avg ( 2808.2[inst], 2805.7[mov,avg]) cwnd 514 rtt2067.6ms
54.8: 2642.8kbps avg ( 2795.0[inst], 2804.7[mov,avg]) cwnd 515 rtt2074.5ms
55.8: 2645.6kbps avg ( 2794.9[inst], 2803.7[mov,avg]) cwnd 515 rtt2075.9ms
56.8: 2648.5kbps avg ( 2805.5[inst], 2803.9[mov,avg]) cwnd 516 rtt2077.0ms
57.9: 2651.1kbps avg ( 2795.7[inst], 2803.0[mov,avg]) cwnd 516 rtt2079.3ms
58.9: 2653.8kbps avg ( 2804.3[inst], 2803.2[mov,avg]) cwnd 517 rtt2079.9ms
59.9: 2656.5kbps avg ( 2811.6[inst], 2804.0[mov,avg]) cwnd 517 rtt2080.9ms
60.9: 2658.7kbps avg ( 2790.4[inst], 2802.6[mov,avg]) cwnd 518 rtt2089.0ms
62.0: 2661.3kbps avg ( 2817.0[inst], 2804.1[mov,avg]) cwnd 518 rtt2087.4ms
63.0: 2663.2kbps avg ( 2773.9[inst], 2801.1[mov,avg]) cwnd 519 rtt2091.3ms
64.0: 2665.4kbps avg ( 2805.4[inst], 2801.5[mov,avg]) cwnd 519 rtt2087.8ms
65.0: 2667.7kbps avg ( 2806.7[inst], 2802.0[mov,avg]) cwnd 520 rtt2092.9ms
66.1: 2670.0kbps avg ( 2816.3[inst], 2803.4[mov,avg]) cwnd 520 rtt2095.4ms
67.1: 2671.7kbps avg ( 2783.1[inst], 2801.4[mov,avg]) cwnd 521 rtt2096.2ms
68.1: 2674.1kbps avg ( 2830.2[inst], 2804.3[mov,avg]) cwnd 521 rtt2099.9ms
^C
root@mininet-vm:~/NetLab/HW6/lab6/tcp#
```





همانطور که از عکس ها مشاهده می کنید اندازه بر حسب بایت ، فریم های اترنتی که برای ارسال داده ها توسط TCP استفاده می شوند برابر با 1514 بایت است چرا که با توجه به MTU ، داده های برنامه های کاربردی 1448 بایت، هدر آی پی 20 بایت، هدر اترنت 14 بایت و هدر TCP برابر با 32 بایت می باشد.

(سوال 9)

$$3 * (1448 / 1514) = 2.869 \text{ Mbps}$$

(سوال 10)

```

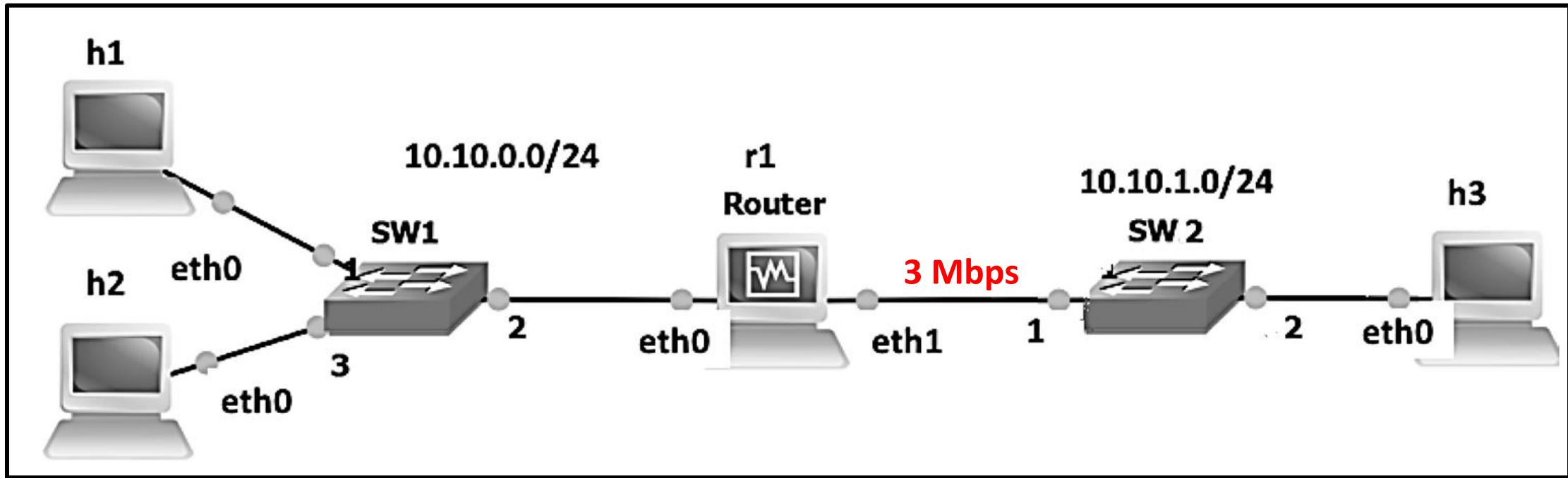
254.4: 2801.2kbps avg ( 2786.5[inst], 2800.5[mov,avg]) cwnd 612 rtt2464.4ms
255.4: 2801.3kbps avg ( 2816.3[inst], 2802.1[mov,avg]) cwnd 612 rtt2468.4ms
256.5: 2801.3kbps avg ( 2803.6[inst], 2802.2[mov,avg]) cwnd 613 rtt2467.6ms
257.5: 2801.2kbps avg ( 2785.3[inst], 2800.5[mov,avg]) cwnd 613 rtt2468.8ms
258.5: 2801.3kbps avg ( 2815.1[inst], 2802.0[mov,avg]) cwnd 614 rtt2475.7ms
259.5: 2801.2kbps avg ( 2796.5[inst], 2801.4[mov,avg]) cwnd 614 rtt2478.3ms
260.6: 2801.3kbps avg ( 2805.5[inst], 2801.8[mov,avg]) cwnd 614 rtt2474.9ms
261.6: 2801.3kbps avg ( 2808.5[inst], 2802.5[mov,avg]) cwnd 615 rtt2476.8ms
262.6: 2801.2kbps avg ( 2781.7[inst], 2800.4[mov,avg]) cwnd 615 rtt2478.7ms
263.6: 2801.2kbps avg ( 2805.8[inst], 2801.0[mov,avg]) cwnd 616 rtt2482.8ms
264.7: 2801.2kbps avg ( 2794.6[inst], 2800.3[mov,avg]) cwnd 616 rtt2483.5ms
265.7: 2801.2kbps avg ( 2806.5[inst], 2801.0[mov,avg]) cwnd 617 rtt2483.8ms
266.7: 2801.2kbps avg ( 2806.4[inst], 2801.5[mov,avg]) cwnd 617 rtt2488.7ms
267.7: 2801.3kbps avg ( 2805.8[inst], 2801.9[mov,avg]) cwnd 617 rtt2487.3ms
268.7: 2801.2kbps avg ( 2794.3[inst], 2801.2[mov,avg]) cwnd 618 rtt2491.2ms
269.8: 2801.3kbps avg ( 2808.8[inst], 2801.9[mov,avg]) cwnd 618 rtt2491.2ms
270.8: 2801.3kbps avg ( 2805.8[inst], 2802.3[mov,avg]) cwnd 619 rtt2498.3ms
271.8: 2801.2kbps avg ( 2784.0[inst], 2800.5[mov,avg]) cwnd 619 rtt2495.7ms
272.8: 2801.2kbps avg ( 2803.8[inst], 2800.8[mov,avg]) cwnd 619 rtt2495.6ms
273.9: 2801.3kbps avg ( 2807.6[inst], 2801.5[mov,avg]) cwnd 620 rtt2496.7ms
274.9: 2801.2kbps avg ( 2794.8[inst], 2800.8[mov,avg]) cwnd 620 rtt2500.1ms
275.9: 2801.2kbps avg ( 2803.6[inst], 2801.1[mov,avg]) cwnd 621 rtt2500.6ms
276.9: 2801.3kbps avg ( 2808.8[inst], 2801.9[mov,avg]) cwnd 621 rtt2501.8ms
278.0: 2801.3kbps avg ( 2804.3[inst], 2802.1[mov,avg]) cwnd 621 rtt2503.1ms
279.0: 2801.3kbps avg ( 2797.8[inst], 2801.7[mov,avg]) cwnd 622 rtt2507.4ms
280.0: 2801.3kbps avg ( 2805.1[inst], 2802.0[mov,avg]) cwnd 622 rtt2507.5ms
281.0: 2801.3kbps avg ( 2806.1[inst], 2802.4[mov,avg]) cwnd 623 rtt2513.5ms
282.1: 2801.3kbps avg ( 2795.0[inst], 2801.7[mov,avg]) cwnd 623 rtt2508.2ms
^C
-
```

همانطور که مشاهده می شود مقدار بدست آمده برای goodput برابر با 2801 بوده که نزدیک به مقدار تئوری محاسبه شده  $= 2869 \text{ Kbps}$  شده است.

# TCP & UDP

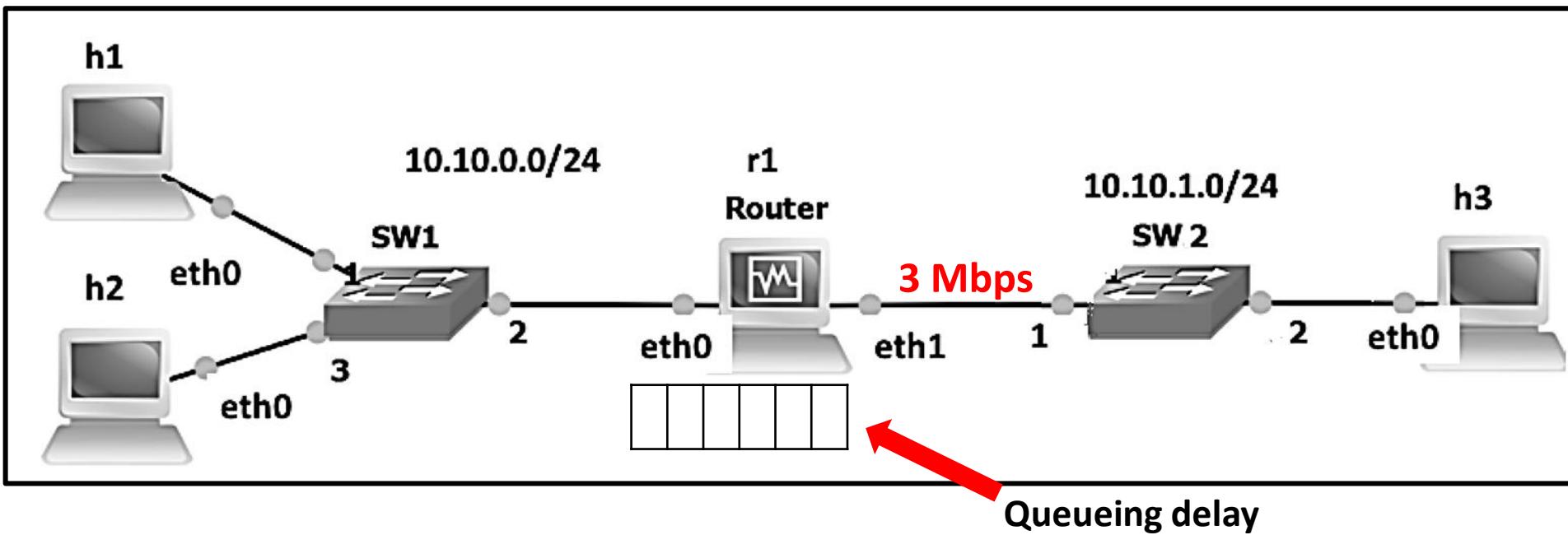
# Previous scenario

- `link_r1sw2.intf1.config( bw=3 )`

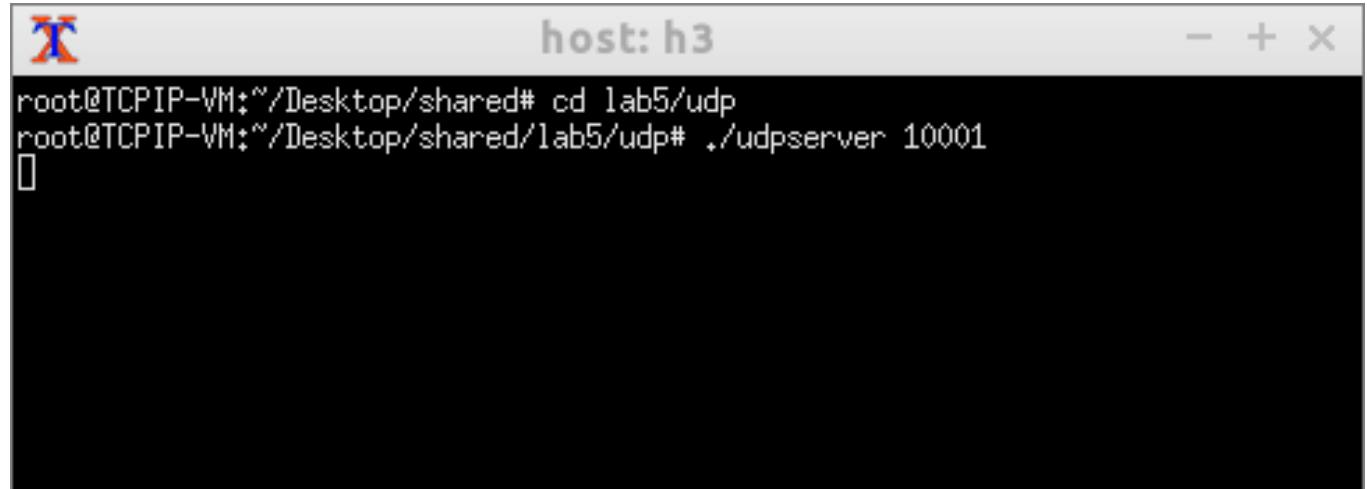


# Competing UDP Flows

Scenario	h1 (UDP)	h2 (UDP)
1	1 Mbps	1 Mbps
2	1 Mbps	2 Mbps
3	1 Mbps	4.5 Mbps



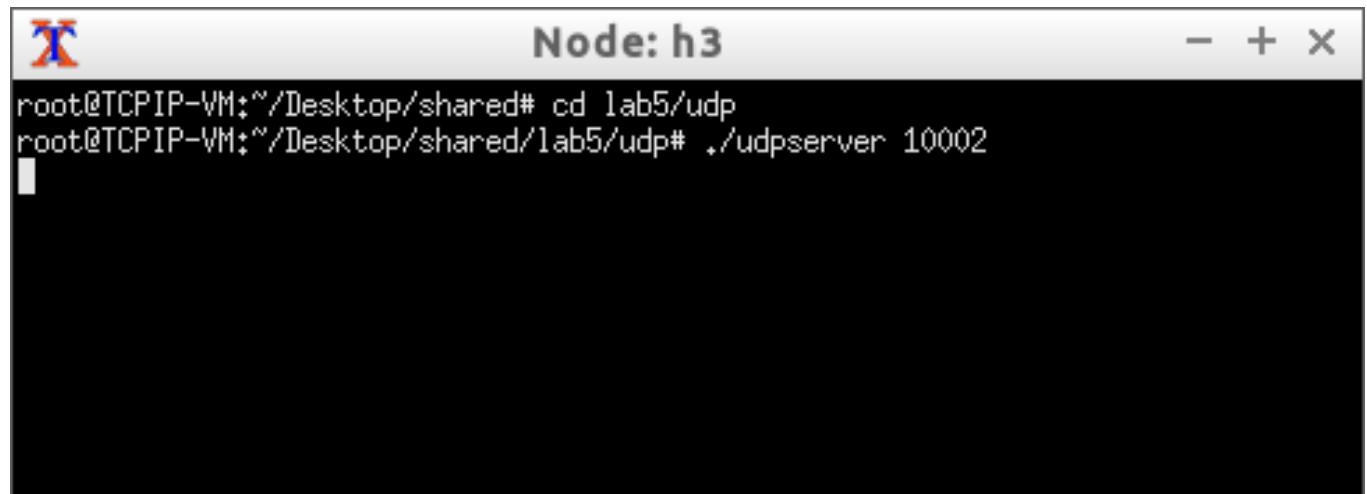
# Open a new terminal for h3



host: h3

```
root@TCPPIP-VM:~/Desktop/shared# cd lab5/udp
root@TCPPIP-VM:~/Desktop/shared/lab5/udp# ./udpserver 10001
[]
```

- mininet> xterm h3



Node: h3

```
root@TCPPIP-VM:~/Desktop/shared# cd lab5/udp
root@TCPPIP-VM:~/Desktop/shared/lab5/udp# ./udpserver 10002
[]
```

# Competing UDP Flows

Scenario	h1 (UDP)	h2 (UDP)
1	X = 1 Mbps	Y = 1 Mbps
2	X = 1 Mbps	Y = 2 Mbps
3	X = 1 Mbps	Y = 4.5 Mbps

$$goodput_{h1} = \min \left( \left( \frac{X}{X + Y} \right) \times 3 \times \frac{1000}{1042}, X \right) Mbps$$

$$goodput_{h2} = \min \left( \left( \frac{Y}{X + Y} \right) \times 3 \times \frac{1000}{1042}, Y \right) Mbps$$

# TCP flows Competing with UDP Flows

Scenario	h1 (UDP)	h2 (UDP)	h2 (TCP)
1	$X = 1 \text{ Mbps}$	$Y = 1 \text{ Mbps}$	$Z$
2	$X = 1 \text{ Mbps}$	$Y = 2 \text{ Mbps}$	$Z$
3	$X = 1 \text{ Mbps}$	$Y = 4.5 \text{ Mbps}$	$Z$

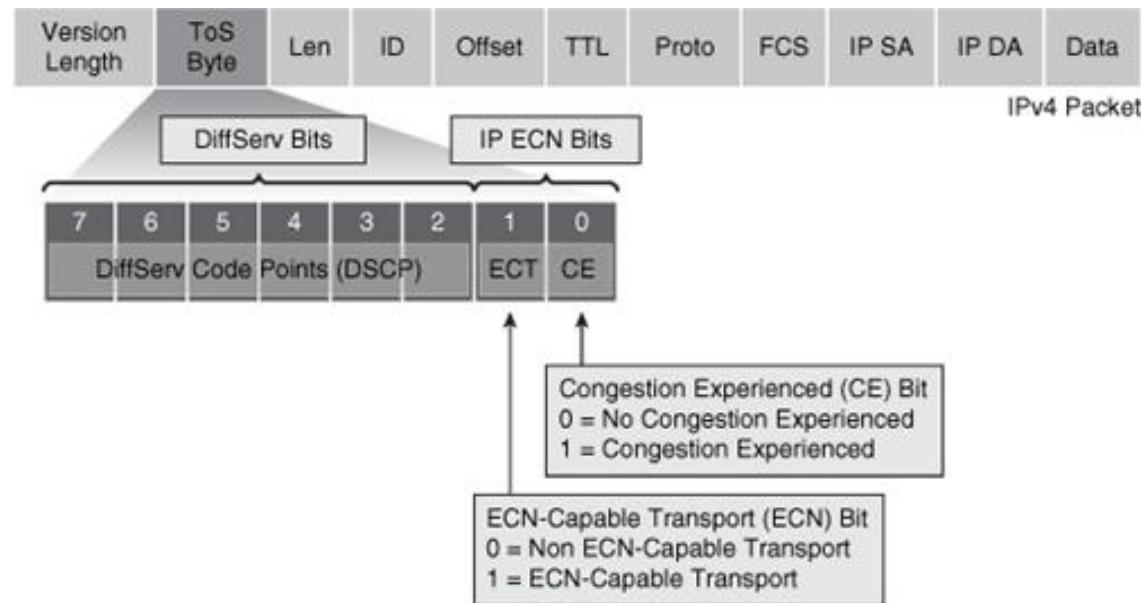
$$goodput_{h1} = \min \left( \left( \frac{X}{X+Y} \right) \times 3 \times \frac{1000}{1042}, X \right) \text{Mbps}$$

$$goodput_{h2,UDP} = \min \left( \left( \frac{Y}{X+Y} \right) \times 3 \times \frac{1000}{1042}, Y \right) \text{Mbps}$$

$$goodput_{h2,TCP} = \begin{cases} 0 \text{ Mbps}, & X + Y \geq 3 \times \frac{1000}{1042} \\ \left( 3 - \left( (X+Y) \times \frac{1042}{1000} \right) \right) \times \frac{1448}{1514} \text{ Mbps}, & X + Y < 3 \times \frac{1000}{1042} \end{cases}$$

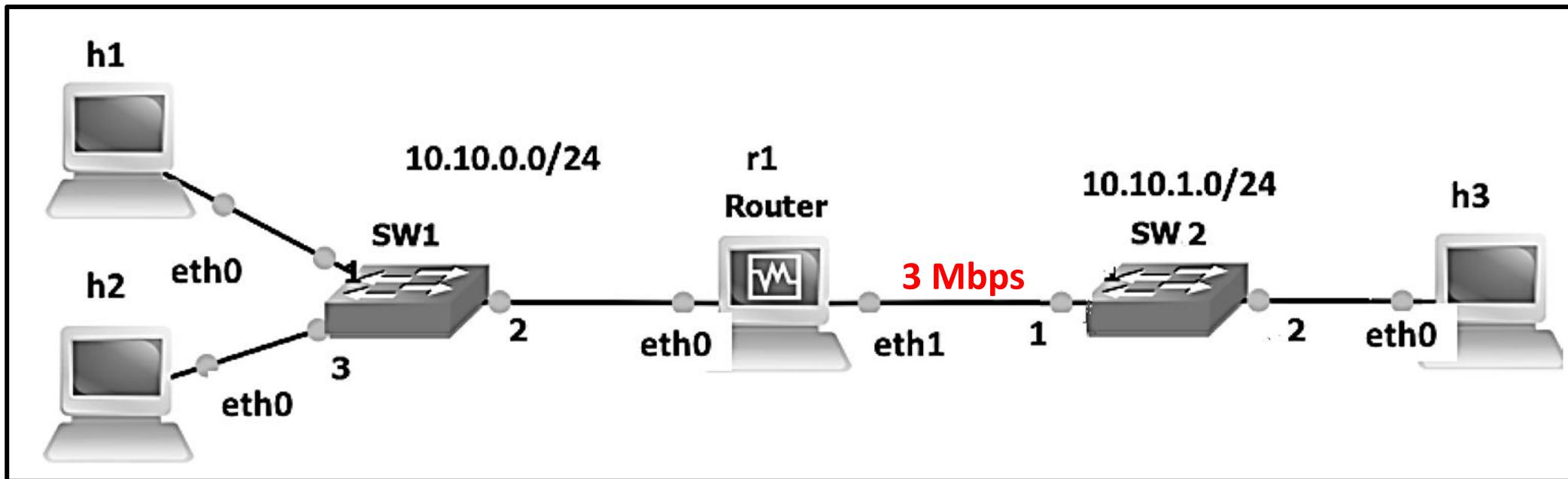
# Explicit Congestion Notification (ECN)

- An extension to the Internet Protocol (the network layer protocol)



1. `link_r1sw2.intf1.config( bw=5, max_queue_size=1000, enable_ecn=False )`
2. `link_r1sw2.intf1.config( bw=5, max_queue_size=1000, enable_ecn=True )`

# Add delay to all packets going out of an interface



- (h3)# tc qdisc add dev h3-eth0 root netem delay 300ms

# Fairness Between TCP Connections and Delay

Scenario	h1 (TCP)	h2 (TCP)	h2 (TCP)	h2 (TCP)
1	X	X	X	X

$$goodput_{h1} = \left( \frac{X}{4X} \right) \times 3 \times \frac{1448}{1514} \text{ Mbps}$$

$$goodput_{h2} = \left( \frac{3X}{4X} \right) \times 3 \times \frac{1448}{1514} \text{ Mbps}$$

## آزمایشگاه شبکه

### آزمایش ۷: بررسی رفتار جریان‌های TCP و UDP

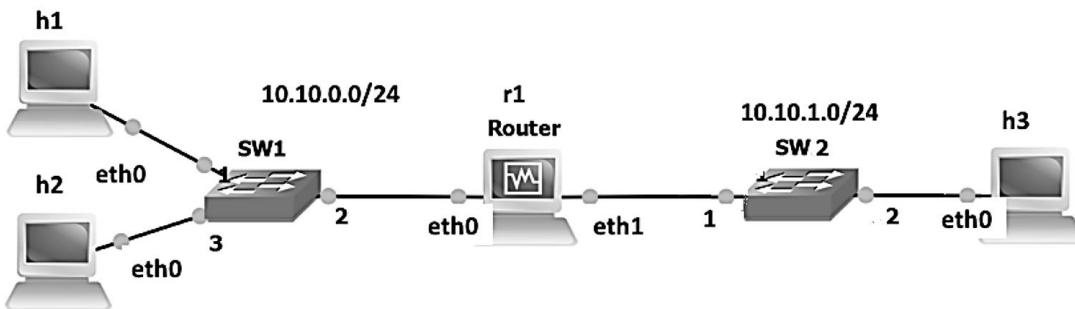
در این آزمایش، به بررسی رفتار جریان‌های ترافیک TCP و UDP در سناریوهای مختلف می‌پردازیم.

#### الف) رقابت جریان‌های UDP با یکدیگر

بار دیگر، توپولوژی آزمایش قبل را تجسم کنید (شکل ۱). پیش از انجام آزمایش، قصد داریم که در سه سناریوی مختلف، مقدار داده‌های تبادل شده را پیش‌بینی کنیم.

جدول ۱- سه سناریوی ترافیکی مختلف مبتنی بر UDP

سناریو	نرخ h1 (UDP)	نرخ h2 (UDP)
۱	1 Mbps	1 Mbps
۲	1 Mbps	2 Mbps
۳	1 Mbps	4.5 Mbps



شکل ۱- توپولوژی مت Shank از سه PC و یک روتر

در هر سناریو از جدول ۱، بر اساس تحلیل نظری، مقادیر goodput و احتمالات loss را می‌توان به این شرح محاسبه نمود:  $X$  را برابر با نرخ h1 و  $Y$  را نرخ h2 در نظر بگیرید. داریم:

$$goodput_{h1} = \min \left( \left( \frac{X}{X+Y} \right) \times \frac{1000}{1042} \times 3, X \right) Mbps$$

$$goodput_{h2} = \min \left( \left( \frac{Y}{X+Y} \right) \times \frac{1000}{1042} \times 3, Y \right) Mbps$$

بر اساس این محاسبات، جدول ۲، goodput و احتمال loss را برای سه سناریوی مندرج در جدول ۱ نشان می‌دهد.

جدول ۲- مقادیر نظری goodput و احتمال loss برای سه سناریوی ترافیکی مختلف مبتنی بر UDP

سناریو	مقدار goodput برای h1	احتمال loss برای h1	مقدار goodput برای h2	احتمال loss برای h2	احتمال
(۱)	1 Mbps	0%	1 Mbps	0%	0%
(۲)	0.9596 Mbps	4%	1.919 Mbps	4%	4%
(۳)	0.523 Mbps	48%	2.36 Mbps	48%	48%

حال، می خواهیم که نتایج تحلیلی فوق را با انجام آزمایش، مورد بررسی قرار دهیم. مشابه قبل، پهنانی باند اینترفیس eth1 از روتر به مقدار 3Mbps محدود شده است.

- برای هر سناریو، دو سرور UDP روی پورت های 10001 و 10002 گوش می دهند. از دستور h3 xterm در پنجره ترمینال mininet استفاده کنید تا یک ترمینال جدید برای h3 باز شود. در انجام آزمایش، دقت کنید که این دو پنجره را به اشتباه نگیرید. سپس، یک کلاینت UDP روی h1 باز کنید که با نرخ 1 Mbps داده ارسال می کند و یک کلاینت UDP هم روی h2 باز کنید که (بسته به سناریو) با نرخ های 1 Mbps و 2 Mbps 4.5 Mbps برای سرور h3 داده می فرستد.

**سؤال ۱: مقادیر goodput و احتمالات loss مورد مشاهده در سناریوهای (۱)، (۲) و (۳) چقدر است؟**

**سؤال ۲: آیا تفاوتی میان این مقادیر تجربی با مقادیر تحلیلی مشاهده می کنید؟ اگر بلی، فکر می کنید این تفاوتها ناشی از چیست؟**

### ب) رقابت جریان TCP با جریان های UDP

در این بخش، سناریوی مشابه بخش (الف) را مد نظر قرار می دهیم. جدول ۳ را ملاحظه نمایید. ماشین h1 جریان داده های UDP را با نرخ 1 Mbps به سوی h3 ارسال می کند و ماشین h2 نیز جریان UDP دیگری (مثلاً ویدئو) با نرخ های 1، 2 و 4 Mbps برای h3 می فرستد. علاوه بر اینها، ماشین h2 یک ارتباط TCP با h3 باز کرده تا مثلاً فایلی را روی این سرور بارگذاری نماید.

جدول ۳- سه سناریویی ترافیکی مختلف با ترکیب جریان‌های TCP و UDP

سناریو	نرخ h1 (UDP)	نرخ h2 (UDP)
(۱)	1 Mbps	1 Mbps
(۲)	1 Mbps	2 Mbps
(۳)	1 Mbps	4.5 Mbps

در هر سناریو از جدول ۳، بر اساس تحلیل نظری، مقادیر goodput و احتمالات loss را می‌توان به این شرح محاسبه نمود:  $X$  را برابر با نرخ جریان UDP در  $h1$  و  $Y$  را نرخ جریان UDP در  $h2$  و  $Z$  را هم نرخ جریان TCP در  $h2$  در نظر بگیرید. داریم:

$$Z = 3 - X - Y \text{ Mbps}$$

$$goodput_{h1,UDP} = \min \left( \left( \frac{X}{X+Y} \right) \times \frac{1000}{1042} \times 3, X \right) \text{ Mbps}$$

$$goodput_{h2,UDP} = \min \left( \left( \frac{Y}{X+Y} \right) \times \frac{1000}{1042} \times 3, Y \right) \text{ Mbps}$$

$$goodput_{h2,TCP} = Z \times \frac{1448}{1514} \text{ Mbps}$$

\* در واقع، به طور دقیق‌تر، در مواردی که  $X + Y < 3$  خواهیم داشت که:

$$Z = 3 - \frac{1042 \times (X + Y)}{1000},$$

$$goodput_{h1,UDP} = X, \quad goodput_{h2,UDP} = Y, \quad goodput_{h2,TCP} = Z \times \frac{1448}{1514}$$

بر اساس محاسبات فوق، جدول ۴، مقادیر goodput را برای سه سناریویی مندرج در جدول ۳ نشان می‌دهد.

جدول ۴- مقادیر نظری goodput برای سه سناریویی ترافیکی مختلف مبتنی بر UDP و TCP

سناریو	مقدار goodput برای جریان h2 در TCP	مقدار goodput برای جریان h2 در UDP	مقدار goodput برای جریان h1 در UDP
(۱)	0.876 Mbps	1 Mbps	1 Mbps
(۲)	0 Mbps	1.919 Mbps	0.959 Mbps
(۳)	0 Mbps	2.36 Mbps	0.523 Mbps

**سؤال ۳: سناریوهای جدول ۳ را مورد آزمایش تجربی قرار دهید. آیا تفاوتی میان این مقادیر تجربی با مقادیر تحلیلی مشاهده می‌کنید؟ اگر بلی، فکر می‌کنید این تفاوت‌ها ناشی از چیست؟**

ج) بررسی تأثیر مکانیزم «اخطر صریح ازدحام<sup>۱</sup>» بر RTT و «پنجره ازدحام<sup>۲</sup>»

قابلیت ECN این امکان را فراهم می‌کند که بدون drop شدن بسته‌ها، فرستنده‌ها را از رویداد ازدحام قریب الوقوع مطلع ساخت. به طور سنتی، شبکه‌های TCP/IP با drop کردن بسته‌ها وقوع ازدحام را به طور ضمنی اطلاع می‌دهند. اماً وقتی قابلیت ECN فعال شود، یک روتِر مجهز به این قابلیت، می‌تواند به جای drop کردن بسته، یک نشانه در هدیر IP بگذارد تا از احتمال وقوع ازدحام خبر دهد. گیرنده بسته هم، اخطر ازدحام را به اطلاع فرستنده نظیرش می‌رساند تا در نهایت، با کاهش نرخ ارسال واکنش نشان دهد.

در این بخش، تأثیر فعال‌سازی قابلیت ECN روی RTT و «پنجره ازدحام» را بررسی خواهیم نمود. برای این منظور، از همان توپولوژی شکل ۱ استفاده می‌کنیم.

- پس از خروج از Mininet و پاک کردن توپولوژی پیشین، اسکریپت lab6.py را طوری تغییر دهید که طول صفت روتِر ۱ (با تنظیم پارامتر `max_queue_size`) به مقدار ۱۰۰۰ بسته کاهش یابد. بعلاوه، پنهانی `False` باند اینترفیس `eth1` از این روتِر را روی ۵ Mbps تنظیم کرده و ویژگی `enable_ecn` را نیز برابر `True` قرار دهید. یک سرور TCP روی ماشین `h3` اجرا نموده و یک کلاینت TCP هم روی ماشین `h1` بالا بیاورید. منتظر بمانید تا نرخ‌ها پایدار شوند.

**سؤال ۴: مقدار نرخ ماشین منبع (یعنی `h1`) چقدر است؟ حدود مقادیر RTT و نیز محدوده مقادیر «پنجره ازدحام» را مشخص نمایید.**

- حال، با دستکاری پیکربندی `eth1` از روتِر ۱، قابلیت ECN در آن را فعال نمایید، به صورت زیر:

```
link_r1sw2.intf1.config(bw=5, max_queue_size=1000, enable_ecn=True)
```

ویژگی `enable_ecn` به صورت پیش‌فرض با مقدار `False` تنظیم شده است که اگر این مقدار را به `True` تغییر دهید، عملًا ECN فعال می‌شود. حال، از Mininet خارج شده و پس از پاک کردن توپولوژی قبلی، اسکریپت اصلاح شده `lab6.py` را اجرا نمایید. یک سرور TCP روی ماشین `h3` بالا آورده و یک کلاینت TCP هم روی `h1` اجرا کنید. منتظر بمانید تا نرخ‌ها پایدار شوند.

<sup>1</sup> Explicit Congestion Notification (ECN)

<sup>2</sup> Congestion Window (cwnd)

**سؤال ۵: مقدار نرخ منبع (یعنی  $h_1$ ) چقدر است؟ محدوده مقادیر RTT و حدود مقادیر «پنجره ازدحام» را بیان کنید.**

**سؤال ۶: با مقایسه مقادیر مشاهده شده در سؤال ۵ با مقادیری که نظیر حالت ECN غیرفعال هستند (سؤال ۴)، چه نتیجه‌ای می‌توان گرفت؟**

#### د) عدالت در TCP و تأثیر RTT

الگوریتم کنترل ازدحام TCP تضمین می‌کند که منابع شبکه به طور عادلانه میان ارتباطات مختلف به اشتراک گذاشته شود. در این بخش، بررسی می‌کنیم که در شرایطی که چندین گلوبگاه (bottleneck) در شبکه وجود دارد، الگوریتم RENO چگونه پهنانی باند را به اشتراک می‌گذارد. مشخصاً، وجود دو ویژگی را در RENO بررسی می‌کنیم: اینکه برای هر جریان (عادلانه) fair است و اینکه نسبت به تأخیر، حساس است.

- برای بخش‌های (۵-۱) و (۵-۲)، همان تپیلوژی شکل ۱ را مبنا قرار می‌دهیم. همچنین، پهنانی باند روتر را به 3 Mbps محدود می‌کنیم و طول صفحه را به ۱۰۰۰ بسته.
- برای آزمایش‌های این بخش، اهمیت ویژه‌ای دارد که صبر کنید تا goodput های حاصل، پایدار شوند (حدود ۵ دقیقه). همچنین، وقتی پهنانی باند روتر را محدود می‌نماییم، برای اینکه همگرایی بهتری عاید شود و کمتر تحت تأثیر تأخیر صفحه باشیم (که روی RTT اثر می‌گذارد)، باید مشابه بخش قبل، قابلیت ECN را فعال کنیم.

#### ۵-۱) افزودن تأخیر به یک اینترفیس

به منظور اینکه شرایط آزمایش را واقع‌بینانه‌تر کنیم، قدری تأخیر به شبکه می‌افزاییم. برای این منظور، از ماجول netem از نرم‌افزار کنترل ترافیکی که در لینوکس موجود است، استفاده می‌کنیم. می‌توان از دستور tc برای تعریف یک قانون جهت افزودن تأخیر به یک اینترفیس بهره گرفت (جهت آشنایی بیشتر با سازوکار دستورات tc و ماجول netem به ویسایت <http://www.linuxfoundation.org/collaborate/workgroups/networking/netem> رجوع نمایید).

برای مثال، دستور زیر به میزان 300ms تأخیر به کلیه بسته‌های خارج‌شونده از اینترفیس eth0 اضافه می‌کند (توجه کنید که افزایش تأخیر فقط در یک جهت انجام می‌شود و نه به بسته‌های وارد شونده):

```
# tc qdisc add dev eth0 root netem delay 300ms
```

برای اعمال تغییر در این قانون، می‌توان دستوری مثل زیر را تایپ نمود:

```
# tc qdisc change dev eth0 root netem delay 400ms
```

همچنین، برای حذف یک قانون، دستور زیر می‌تواند استفاده شود:

```
# tc qdisc del dev eth0 root
```

**سؤال ۷:** به مقدار 300ms تأخیر به اینترفیس eth0 از h3 اضافه نمایید. سپس، از سوی h1 ماشین h3 را پینگ کنید. مقدار RTT مشاهده شده چقدر است؟

\* اگر جدول ARP را flush کرده باشید (مثلاً با استفاده از ifconfig eth0 up و ifconfig eth0 down) و سپس، netem را برای اضافه کردن تأخیر 300ms استفاده کنید، RTT اولین بسته باید بزرگ‌تر از RTT بسته دوم باشد (به خاطر درخواست ARP).

## ۵-۲) عدالت میان جریان‌های TCP و تأثیر تأخیر بر آن

TCP نوعی اشتراک‌گذاری عادلانه میان جریان‌های ترافیک فراهم می‌کند به این معناکه ماشینی که چندین ارتباط TCP باز می‌کند، پهنه‌ای باند بیشتری هم عایدش خواهد شد. برای بررسی این موضوع، سناریویی را در نظر می‌گیریم که در آن،

- یک مقدار تأخیر اضافه 300ms روی اینترفیس eth0 از ماشین h3 وجود دارد ولی هیچ تأخیر اضافه‌ای روی ماشین‌های h1 یا h2 نیست.
- ماشین h1 یک ارتباط TCP به سوی h3 باز می‌کند و ماشین h2 هم سه ارتباط TCP به سوی h3 می‌گشاید. بر مبنای تحلیل نظری، کل goodput ماشین‌های h1 و h2 در سناریوی فوق بدست می‌آورند، به صورت زیر قابل محاسبه است:

فرض کنید که سهم جریان h1 را با  $x$  نشان دهیم. در اینصورت، سهم h2 برابر است با  $3x$ . از طرفی، مجموع این دو جریان یعنی  $4x$  باید حداقل برابر با  $\frac{3}{4}x$  بشود. پس،  $\frac{3}{4}x = 3/4$ . در نهایت، می‌توان سهم goodput ماشین h1 را به صورت  $\frac{3/4 * 1448}{1514}$  محاسبه نمود که برابر می‌شود با:  $7.17 \text{ kbps}$ . در خصوص سهم کلی h2 هم این مقدار معادل  $2152 \text{ kbps}$  خواهد شد.

• سه کلاینت TCP روی ماشین h2 اجرا کنید و تنها یک کلاینت TCP روی h1. صبر کنید نرخ‌ها پایدار شوند.

**سؤال ۸:** مقدار goodput هر جریان از ماشین‌های h1 و h2 چقدر است؟

**سؤال ۹:** آیا مقادیر حاصل از آزمایش با مقادیر نظری همخوانی دارند؟

**سؤال ۱۰:** آیا می‌توانید متوجه شوید که اساساً تأخیر صفر هم داریم یا خیر؟

## گزارشکار آزمایش 7: بررسی رفتار جریان های UDP و TCP

سیده شکیبا انارکی فیروز - ۹۹۴۴۲۰۴۷

بهاره کاوی نژاد - ۹۹۴۳۱۲۱۷

الف) رقابت جریان های UDP با یکدیگر

سوال 1: مقادیر goodput و احتمالات loss مورد مشاهده در سناریوهای (1)، (2) و (3) چقدر است؟

به منظور آماده کردن محیط برنامه، فایل ها و فولدریندی های مناسب (برای TCP و UDP) را طبق آزمایش قبل آماده می کنیم و با اجرای فایل پایتون، وارد محیط mininet می شویم.

```

host: h1
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ls
lab7.py  tcp  udp
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd tcp
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./tcpclient 10002
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./tcpclient 10001
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./tcpserver
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./tcpserver
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ls
Makefile  tcpclient.c  tcpserver.c
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd ..
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd udp
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpserver 10002
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpserver 10001
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpclient 10002
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpclient 10001
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# sudo -E python lab7.py
** Creating an instance of Lab5 network topology

host: h2
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ls
** Adding Controller
** Adding Hosts
** Adding Switches
** Creating Links
** Modifying Link Parameters
(3.00Mbit) *** Configuring hosts
h1 h2 h3 r1
*** Starting controller
*** Starting 2 switches
s1 s2
*** Configuring hosts
** Executing custom commands
** Enabling xterm for hosts only
** Running CLI
*** Starting CLI:
mininet> _

host: h3
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd tcp
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./tcpclient 10002
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./tcpclient 10001
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd ..
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd udp
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpserver 10002
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpserver 10001
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpclient 10002
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpclient 10001
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ls
Makefile  udpclient.c  udpserver.c
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd ..
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ..
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# sudo -E python lab7.py
** Creating an instance of Lab5 network topology

```

سناریو ها را به ترتیب اجرا می کنیم. با استفاده از دستور host h3 یک ترمینال جدید برای host h3 باز می کنیم. ترمینال host h3 همان ترمینالی است که از host h1 به آن درخواست می فرستیم و ترمینال host h3 (که با دستور بالا باز شد) همان ترمینالی است که host h2 با رخ های مختلف به آن درخواست ارسال می کند.

سرورهای UDP را روی host h3 با پورت 10001 و روی Node h3 با پورت 10002 اجرا می کنیم:

```

Node: h3
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd udp
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpserver 10002

```

```

host: h3
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# cd udp
bash: cd: udp: No such file or directory
root@TCP/IP-VM:/home/mininet/Desktop/shared/Session7# ./udpserver 10001

```

در هر سه سناریو، host h1 بسته ها را با نرخ 4.5 Mbps، host h2 با نرخ 2 Mbps و host h3 با نرخ 1 Mbps ارسال می کند. اما ما در این آزمایش نرخ های مختلف را برای host h1 نیز امتحان کردیم.

بخش اول:

1 Mbps را با نرخ 1 Mbps ارسال را انجام می دهیم:

```
host: h1
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# ./udpclient 10.10.1.3 1000
1 1000
bash: ./udpclient: No such file or directory
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# cd udp
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
1000 1000
 1.0s - sent: 126 pkts, 1007.8 kbytes/s
 2.0s - sent: 252 pkts, 1000.7 kbytes/s
 3.0s - sent: 352 pkts, 798.4 kbytes/s
 4.0s - sent: 506 pkts, 1224.5 kbytes/s
 5.1s - sent: 630 pkts, 984.8 kbytes/s
 6.1s - sent: 762 pkts, 1054.6 kbytes/s
 7.1s - sent: 888 pkts, 1002.0 kbytes/s
 8.1s - sent: 1014 pkts, 1000.5 kbytes/s
 9.1s - sent: 1140 pkts, 998.7 kbytes/s
10.1s - sent: 1266 pkts, 1003.0 kbytes/s
11.1s - sent: 1391 pkts, 999.7 kbytes/s
12.1s - sent: 1517 pkts, 1000.1 kbytes/s
13.1s - sent: 1643 pkts, 1000.9 kbytes/s
14.1s - sent: 1768 pkts, 999.7 kbytes/s
15.1s - sent: 1894 pkts, 1000.0 kbytes/s
^Cpackets sent = 1943, avg rate=1000.0kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp#
```

```
host: h3
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# cd upd
bash: cd: upd: No such file or directory
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# cd udp
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp# ./udpserver 10001
 0.0s - received: 1/ sent: 1 pkts (loss 0.000%), 0.0 kbit/s
369.4s - received: 128/ sent: 128 pkts (loss 0.000%), 1008.6 kbit/s
370.4s - received: 254/ sent: 254 pkts (loss 0.000%), 999.9 kbit/s
371.4s - received: 353/ sent: 353 pkts (loss 0.000%), 782.1 kbit/s
372.4s - received: 506/ sent: 506 pkts (loss 0.000%), 1221.9 kbit/s
373.4s - received: 631/ sent: 631 pkts (loss 0.000%), 956.0 kbit/s
374.5s - received: 763/ sent: 763 pkts (loss 0.000%), 1053.1 kbit/s
375.5s - received: 889/ sent: 889 pkts (loss 0.000%), 999.8 kbit/s
376.5s - received: 1015/ sent: 1015 pkts (loss 0.000%), 1000.8 kbit/s
377.5s - received: 1140/ sent: 1140 pkts (loss 0.000%), 999.9 kbit/s
378.5s - received: 1266/ sent: 1266 pkts (loss 0.000%), 1001.5 kbit/s
379.5s - received: 1391/ sent: 1391 pkts (loss 0.000%), 999.9 kbit/s
380.5s - received: 1517/ sent: 1517 pkts (loss 0.000%), 1000.5 kbit/s
381.5s - received: 1642/ sent: 1642 pkts (loss 0.000%), 1000.0 kbit/s
382.5s - received: 1768/ sent: 1768 pkts (loss 0.000%), 1000.4 kbit/s
383.5s - received: 1894/ sent: 1894 pkts (loss 0.000%), 999.9 kbit/s
packet received = 1943 / 1943 sent: 0.000% loss
[]
```

همان طور که مشاهده می شود، مقدار loss برابر با صفر درصد و مقدار goodput در host h3 برابر با 1000 kbps است.

2 Mbps نرخ با host h1 ارسال را انجام می دهیم:

host: h1

```
15.1s - sent: 1894 pkts, 1000.0 kbytes/s
^Cpackets sent = 1943, avg rate=1000.0kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10001 2000
 1.0s - sent: 251 pkts, 2007.8 kbytes/s
 2.0s - sent: 476 pkts, 1797.3 kbytes/s
 3.0s - sent: 754 pkts, 2217.1 kbytes/s
 4.0s - sent: 1005 pkts, 2001.5 kbytes/s
 5.0s - sent: 1255 pkts, 1920.8 kbytes/s
 6.0s - sent: 1519 pkts, 2104.4 kbytes/s
 7.0s - sent: 1770 pkts, 2002.8 kbytes/s
 8.0s - sent: 2021 pkts, 2000.7 kbytes/s
 9.0s - sent: 2272 pkts, 2001.8 kbytes/s
10.0s - sent: 2523 pkts, 2003.0 kbytes/s
11.0s - sent: 2774 pkts, 2003.4 kbytes/s
12.0s - sent: 3025 pkts, 2002.1 kbytes/s
13.0s - sent: 3275 pkts, 1998.9 kbytes/s
14.0s - sent: 3526 pkts, 2003.2 kbytes/s
15.0s - sent: 3777 pkts, 2001.2 kbytes/s
16.0s - sent: 4028 pkts, 2002.7 kbytes/s
17.0s - sent: 4279 pkts, 1998.0 kbytes/s
18.0s - sent: 4530 pkts, 2006.0 kbytes/s
^Cpackets sent = 4750, avg rate=1999.7kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp#
```

host: h3

```
382.5s - received: 1768/ sent: 1768 pkts (loss 0.000%), 1000.4 kbit/s
383.5s - received: 1894/ sent: 1894 pkts (loss 0.000%), 999.9 kbit/s
packet received = 1943 / 1943 sent: 0.000% loss
384.5s - received: 1/ sent: 1 pkts (loss 0.000%), 10.3 kbit/s
423.4s - received: 252/ sent: 252 pkts (loss 0.000%), 2005.7 kbit/s
424.4s - received: 477/ sent: 477 pkts (loss 0.000%), 1791.1 kbit/s
425.4s - received: 755/ sent: 755 pkts (loss 0.000%), 2216.4 kbit/s
426.4s - received: 1006/ sent: 1006 pkts (loss 0.000%), 2001.2 kbit/s
427.4s - received: 1255/ sent: 1255 pkts (loss 0.000%), 1976.8 kbit/s
428.4s - received: 1511/ sent: 1511 pkts (loss 0.000%), 2041.3 kbit/s
429.4s - received: 1762/ sent: 1762 pkts (loss 0.000%), 1999.2 kbit/s
430.4s - received: 2013/ sent: 2013 pkts (loss 0.000%), 2002.2 kbit/s
431.5s - received: 2264/ sent: 2264 pkts (loss 0.000%), 2001.0 kbit/s
432.5s - received: 2515/ sent: 2515 pkts (loss 0.000%), 2000.8 kbit/s
433.5s - received: 2766/ sent: 2766 pkts (loss 0.000%), 2000.7 kbit/s
434.5s - received: 3017/ sent: 3017 pkts (loss 0.000%), 2000.9 kbit/s
435.5s - received: 3268/ sent: 3268 pkts (loss 0.000%), 2000.8 kbit/s
436.5s - received: 3519/ sent: 3519 pkts (loss 0.000%), 2000.0 kbit/s
437.5s - received: 3770/ sent: 3770 pkts (loss 0.000%), 2001.4 kbit/s
438.5s - received: 4021/ sent: 4021 pkts (loss 0.000%), 1999.8 kbit/s
439.5s - received: 4272/ sent: 4272 pkts (loss 0.000%), 2001.8 kbit/s
440.5s - received: 4522/ sent: 4522 pkts (loss 0.000%), 1999.0 kbit/s
packet received = 4750 / 4750 sent: 0.000% loss
```

همان طور که مشاهده می شود، مقدار loss با صفر درصد و مقدار goodput 2000 kbps در host h3 برابر با است.

ارسال را انجام می دهیم:

```
host: h1
^Cpackets sent = 4750, avg rate=1999.7kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10001 4500
 1.0s - sent: 511 pkts, 3994.6 kbytes/s
 2.0s - sent: 1140 pkts, 5025.8 kbytes/s
 3.0s - sent: 1700 pkts, 4390.3 kbytes/s
 4.1s - sent: 2273 pkts, 4538.1 kbytes/s
 5.1s - sent: 2855 pkts, 4648.5 kbytes/s
 6.1s - sent: 3418 pkts, 4500.4 kbytes/s
 7.1s - sent: 3982 pkts, 4507.4 kbytes/s
 8.1s - sent: 4545 pkts, 4504.0 kbytes/s
 9.1s - sent: 5108 pkts, 4502.9 kbytes/s
10.1s - sent: 5672 pkts, 4505.0 kbytes/s
11.1s - sent: 6235 pkts, 4502.4 kbytes/s
12.1s - sent: 6799 pkts, 4508.0 kbytes/s
13.1s - sent: 7363 pkts, 4504.3 kbytes/s
14.1s - sent: 7926 pkts, 4499.9 kbytes/s
15.1s - sent: 8489 pkts, 4500.9 kbytes/s
16.1s - sent: 9053 pkts, 4507.8 kbytes/s
17.1s - sent: 9617 pkts, 4504.2 kbytes/s
18.1s - sent: 10094 pkts, 3813.8 kbytes/s
19.1s - sent: 10748 pkts, 5223.0 kbytes/s
^Cpackets sent = 10865, avg rate=3932.6kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp#
```

```
host: h3
441.5s - received: 1/ sent: 1 pkts (loss 0.000%), 42.6 kbit/s
484.5s - received: 348/ sent: 348 pkts (loss 0.000%), 2705.2 kbit/s
485.5s - received: 725/ sent: 725 pkts (loss 0.000%), 3014.9 kbit/s
486.5s - received: 1081/ sent: 1081 pkts (loss 0.000%), 2842.1 kbit/s
487.5s - received: 1446/ sent: 1446 pkts (loss 0.000%), 2826.7 kbit/s
488.6s - received: 1820/ sent: 1820 pkts (loss 0.000%), 2988.9 kbit/s
489.6s - received: 2180/ sent: 2180 pkts (loss 0.000%), 2879.9 kbit/s
490.6s - received: 2540/ sent: 2540 pkts (loss 0.000%), 2876.0 kbit/s
491.6s - received: 2902/ sent: 2995 pkts (loss 3.105%), 2889.0 kbit/s
492.6s - received: 3263/ sent: 3560 pkts (loss 8.343%), 2880.1 kbit/s
493.6s - received: 3623/ sent: 4122 pkts (loss 12.106%), 2879.8 kbit/s
494.6s - received: 3983/ sent: 6156 pkts (loss 35.299%), 2880.0 kbit/s
495.6s - received: 4342/ sent: 6515 pkts (loss 33.354%), 2870.9 kbit/s
496.6s - received: 4700/ sent: 6873 pkts (loss 31.616%), 2857.7 kbit/s
497.6s - received: 5060/ sent: 7233 pkts (loss 30.043%), 2874.4 kbit/s
498.6s - received: 5421/ sent: 7594 pkts (loss 28.615%), 2882.5 kbit/s
499.6s - received: 5782/ sent: 7955 pkts (loss 27.316%), 2879.4 kbit/s
500.6s - received: 6143/ sent: 8316 pkts (loss 26.130%), 2881.2 kbit/s
501.6s - received: 6478/ sent: 8651 pkts (loss 25.118%), 2875.2 kbit/s
502.6s - received: 6841/ sent: 9159 pkts (loss 25.308%), 2902.3 kbit/s
503.6s - received: 7202/ sent: 9723 pkts (loss 25.928%), 2880.6 kbit/s
504.6s - received: 7563/ sent: 10326 pkts (loss 26.758%), 2881.7 kbit/s
packet received = 7908 / 10865 sent: 27.216% loss
```

همان طور که مشاهده می شود، مقدار loss تقریبا برابر با 27 درصد و مقدار goodput در host h3 تقریبا برابر با 4800 kbps است.

نرخ 1 Mbps را انجام می دهیم:

Node: h3

```
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# cd udp
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp# ./udpserver 10002
  0.0s - received:  1/ sent:    1 pkts (loss  0.000%),  0.0 kbit/s
1002.7s - received: 127/ sent: 127 pkts (loss  0.000%), 1002.0 kbit/s
1003.7s - received: 252/ sent: 252 pkts (loss  0.000%), 999.9 kbit/s
1004.7s - received: 378/ sent: 378 pkts (loss  0.000%), 1000.2 kbit/s
1005.7s - received: 503/ sent: 503 pkts (loss  0.000%), 999.9 kbit/s
1006.7s - received: 628/ sent: 628 pkts (loss  0.000%), 999.9 kbit/s
1007.7s - received: 754/ sent: 754 pkts (loss  0.000%), 1000.4 kbit/s
1008.7s - received: 880/ sent: 880 pkts (loss  0.000%), 1000.5 kbit/s
1009.8s - received: 1006/ sent: 1006 pkts (loss  0.000%), 999.9 kbit/s
1010.8s - received: 1132/ sent: 1132 pkts (loss  0.000%), 999.7 kbit/s
1011.8s - received: 1258/ sent: 1258 pkts (loss  0.000%), 1000.5 kbit/s
1012.8s - received: 1384/ sent: 1384 pkts (loss  0.000%), 1000.2 kbit/s
1013.8s - received: 1509/ sent: 1509 pkts (loss  0.000%), 999.6 kbit/s
1014.8s - received: 1635/ sent: 1635 pkts (loss  0.000%), 1000.7 kbit/s
1015.8s - received: 1761/ sent: 1761 pkts (loss  0.000%), 1000.5 kbit/s
1016.8s - received: 1886/ sent: 1886 pkts (loss  0.000%), 999.7 kbit/s
1017.8s - received: 2012/ sent: 2012 pkts (loss  0.000%), 1000.3 kbit/s
1018.8s - received: 2137/ sent: 2137 pkts (loss  0.000%), 1000.0 kbit/s
1019.8s - received: 2263/ sent: 2263 pkts (loss  0.000%), 1000.8 kbit/s
packet received = 2346 / 2346 sent: 0.000% loss
[]
```

host: h2

```
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# cd udp
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp# ./udpcclient 10.10.1.3
10002 1000
  1.0s - sent: 126 pkts, 1007.9 kbytes/s
  2.0s - sent: 251 pkts, 999.8 kbytes/s
  3.0s - sent: 377 pkts, 1000.4 kbytes/s
  4.0s - sent: 502 pkts, 1000.0 kbytes/s
  5.0s - sent: 627 pkts, 999.9 kbytes/s
  6.0s - sent: 753 pkts, 1000.2 kbytes/s
  7.0s - sent: 879 pkts, 999.8 kbytes/s
  8.0s - sent: 1005 pkts, 1000.6 kbytes/s
  9.0s - sent: 1130 pkts, 999.7 kbytes/s
 10.0s - sent: 1256 pkts, 1000.9 kbytes/s
 11.0s - sent: 1382 pkts, 999.7 kbytes/s
 12.1s - sent: 1508 pkts, 1000.4 kbytes/s
 13.1s - sent: 1634 pkts, 1000.6 kbytes/s
 14.1s - sent: 1760 pkts, 999.9 kbytes/s
 15.1s - sent: 1886 pkts, 1000.0 kbytes/s
 16.1s - sent: 2012 pkts, 1000.3 kbytes/s
 17.1s - sent: 2137 pkts, 1000.0 kbytes/s
 18.1s - sent: 2263 pkts, 1000.7 kbytes/s
^Cpackets sent = 2346, avg rate=1000.2kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp#
```

همان طور که مشاهده می شود، مقدار loss برابر با صفر درصد و مقدار goodput در h3 برابر با 1000 kbps است.

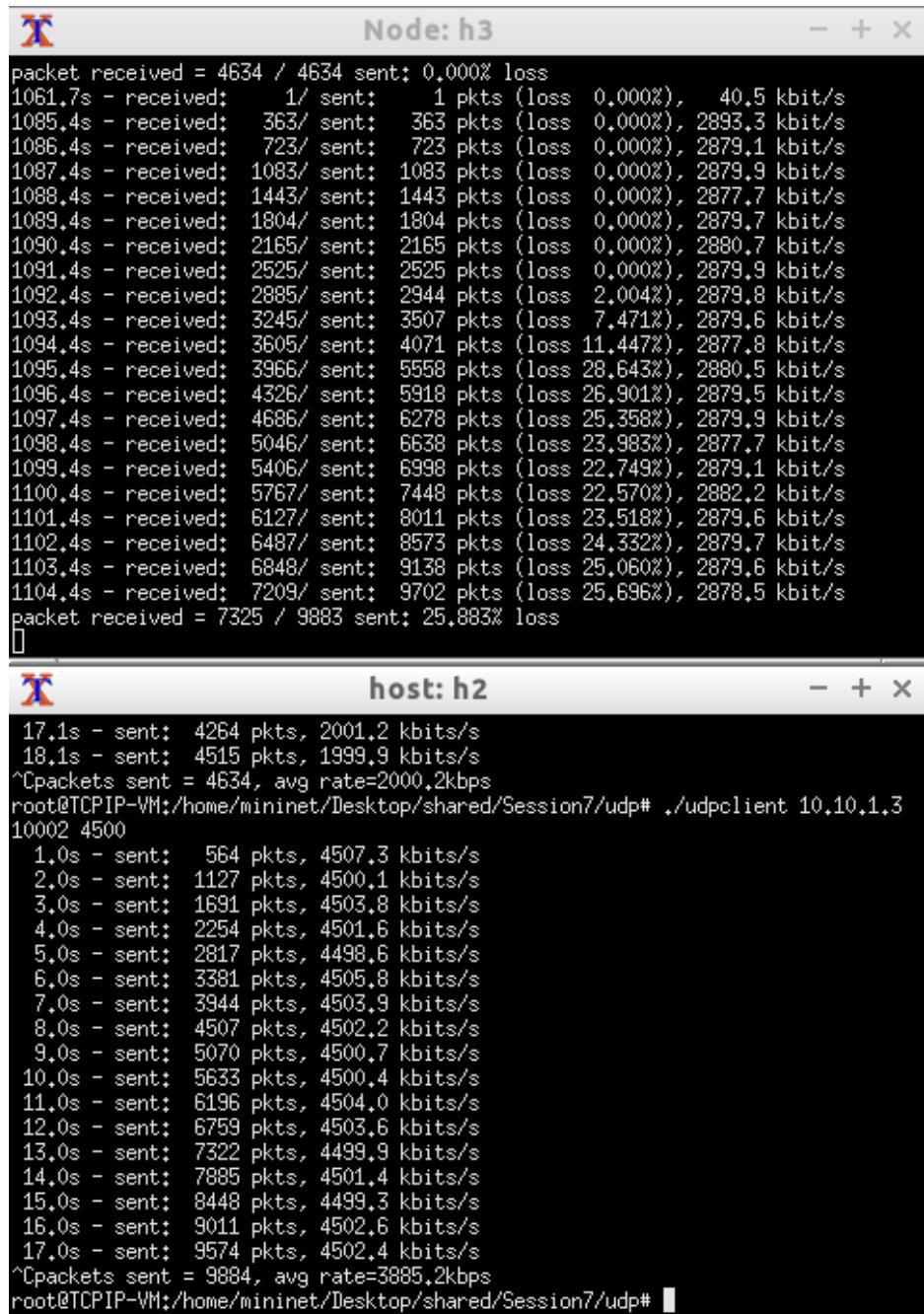
ارسال را انجام می دهیم: 2 Mbps نرخ با host h2

```
Node: h3
1018.8s - received: 2137/ sent: 2137 pkts (loss 0.000%), 1000.0 kbit/s
1019.8s - received: 2263/ sent: 2263 pkts (loss 0.000%), 1000.8 kbit/s
packet received = 2346 / 2346 sent: 0.000% loss
1020.8s - received: 1/ sent: 1 pkts (loss 0.000%), 29.5 kbit/s
1043.6s - received: 252/ sent: 252 pkts (loss 0.000%), 2002.5 kbit/s
1044.6s - received: 503/ sent: 503 pkts (loss 0.000%), 1999.2 kbit/s
1045.6s - received: 754/ sent: 754 pkts (loss 0.000%), 2000.5 kbit/s
1046.6s - received: 1005/ sent: 1005 pkts (loss 0.000%), 2000.7 kbit/s
1047.6s - received: 1255/ sent: 1255 pkts (loss 0.000%), 1999.5 kbit/s
1048.6s - received: 1506/ sent: 1506 pkts (loss 0.000%), 2000.0 kbit/s
1049.6s - received: 1757/ sent: 1757 pkts (loss 0.000%), 2000.9 kbit/s
1050.6s - received: 2008/ sent: 2008 pkts (loss 0.000%), 2000.9 kbit/s
1051.6s - received: 2258/ sent: 2258 pkts (loss 0.000%), 1999.5 kbit/s
1052.6s - received: 2509/ sent: 2509 pkts (loss 0.000%), 2001.7 kbit/s
1053.6s - received: 2760/ sent: 2760 pkts (loss 0.000%), 2000.4 kbit/s
1054.6s - received: 3010/ sent: 3010 pkts (loss 0.000%), 1999.7 kbit/s
1055.6s - received: 3261/ sent: 3261 pkts (loss 0.000%), 2001.2 kbit/s
1056.6s - received: 3512/ sent: 3512 pkts (loss 0.000%), 1999.6 kbit/s
1057.6s - received: 3763/ sent: 3763 pkts (loss 0.000%), 2001.1 kbit/s
1058.6s - received: 4013/ sent: 4013 pkts (loss 0.000%), 1999.8 kbit/s
1059.6s - received: 4264/ sent: 4264 pkts (loss 0.000%), 2000.9 kbit/s
1060.6s - received: 4515/ sent: 4515 pkts (loss 0.000%), 2000.1 kbit/s
packet received = 4634 / 4634 sent: 0.000% loss
[]

host: h2
18.1s - sent: 2263 pkts, 1000.7 kbytes/s
^Cpackets sent = 2346, avg rate=1000.2kbps
root@TCP-VM:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10002 2000
1.0s - sent: 251 pkts, 2006.3 kbytes/s
2.0s - sent: 502 pkts, 2002.5 kbytes/s
3.0s - sent: 752 pkts, 2000.0 kbytes/s
4.0s - sent: 1003 pkts, 2000.1 kbytes/s
5.0s - sent: 1254 pkts, 2002.7 kbytes/s
6.0s - sent: 1505 pkts, 2001.0 kbytes/s
7.0s - sent: 1756 pkts, 2001.0 kbytes/s
8.0s - sent: 2007 pkts, 2000.7 kbytes/s
9.0s - sent: 2258 pkts, 1998.9 kbytes/s
10.0s - sent: 2509 pkts, 2002.0 kbytes/s
11.0s - sent: 2760 pkts, 2000.6 kbytes/s
12.0s - sent: 3011 pkts, 2001.8 kbytes/s
13.0s - sent: 3261 pkts, 1999.9 kbytes/s
14.0s - sent: 3512 pkts, 1999.9 kbytes/s
15.0s - sent: 3763 pkts, 2002.2 kbytes/s
16.0s - sent: 4013 pkts, 1999.6 kbytes/s
17.1s - sent: 4264 pkts, 2001.2 kbytes/s
18.1s - sent: 4515 pkts, 1999.9 kbytes/s
^Cpackets sent = 4634, avg rate=2000.2kbps
root@TCP-VM:/home/mininet/Desktop/shared/Session7/udp#
```

همان طور که مشاهده می شود، مقدار loss برابر با صفر درصد و مقدار goodput در host h3 برابر با 2000 kbps است.

ارسال را نرخ 4.5 Mbps با host h2 انجام می دهیم:



```
Node: h3
packet received = 4634 / 4634 sent: 0.000% loss
1061.7s - received: 1/ sent: 1 pkts (loss 0.000%), 40.5 kbit/s
1085.4s - received: 363/ sent: 363 pkts (loss 0.000%), 2893.3 kbit/s
1086.4s - received: 723/ sent: 723 pkts (loss 0.000%), 2879.1 kbit/s
1087.4s - received: 1083/ sent: 1083 pkts (loss 0.000%), 2879.9 kbit/s
1088.4s - received: 1443/ sent: 1443 pkts (loss 0.000%), 2877.7 kbit/s
1089.4s - received: 1804/ sent: 1804 pkts (loss 0.000%), 2879.7 kbit/s
1090.4s - received: 2165/ sent: 2165 pkts (loss 0.000%), 2880.7 kbit/s
1091.4s - received: 2525/ sent: 2525 pkts (loss 0.000%), 2879.9 kbit/s
1092.4s - received: 2885/ sent: 2944 pkts (loss 2.004%), 2879.8 kbit/s
1093.4s - received: 3245/ sent: 3507 pkts (loss 7.471%), 2879.6 kbit/s
1094.4s - received: 3605/ sent: 4071 pkts (loss 11.447%), 2877.8 kbit/s
1095.4s - received: 3966/ sent: 5558 pkts (loss 28.643%), 2880.5 kbit/s
1096.4s - received: 4326/ sent: 5918 pkts (loss 26.901%), 2879.5 kbit/s
1097.4s - received: 4686/ sent: 6278 pkts (loss 25.358%), 2879.9 kbit/s
1098.4s - received: 5046/ sent: 6638 pkts (loss 23.983%), 2877.7 kbit/s
1099.4s - received: 5406/ sent: 6998 pkts (loss 22.749%), 2879.1 kbit/s
1100.4s - received: 5767/ sent: 7448 pkts (loss 22.570%), 2882.2 kbit/s
1101.4s - received: 6127/ sent: 8011 pkts (loss 23.518%), 2879.6 kbit/s
1102.4s - received: 6487/ sent: 8573 pkts (loss 24.332%), 2879.7 kbit/s
1103.4s - received: 6848/ sent: 9138 pkts (loss 25.060%), 2879.6 kbit/s
1104.4s - received: 7209/ sent: 9702 pkts (loss 25.696%), 2878.5 kbit/s
packet received = 7325 / 9883 sent: 25.883% loss
[]

host: h2
17.1s - sent: 4264 pkts, 2001.2 kbytes/s
18.1s - sent: 4515 pkts, 1999.9 kbytes/s
^Cpackets sent = 4634, avg rate=2000.2kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10002 4500
1.0s - sent: 564 pkts, 4507.3 kbytes/s
2.0s - sent: 1127 pkts, 4500.1 kbytes/s
3.0s - sent: 1691 pkts, 4503.8 kbytes/s
4.0s - sent: 2254 pkts, 4501.6 kbytes/s
5.0s - sent: 2817 pkts, 4498.6 kbytes/s
6.0s - sent: 3381 pkts, 4505.8 kbytes/s
7.0s - sent: 3944 pkts, 4503.9 kbytes/s
8.0s - sent: 4507 pkts, 4502.2 kbytes/s
9.0s - sent: 5070 pkts, 4500.7 kbytes/s
10.0s - sent: 5633 pkts, 4500.4 kbytes/s
11.0s - sent: 6196 pkts, 4504.0 kbytes/s
12.0s - sent: 6759 pkts, 4503.6 kbytes/s
13.0s - sent: 7322 pkts, 4499.9 kbytes/s
14.0s - sent: 7885 pkts, 4501.4 kbytes/s
15.0s - sent: 8448 pkts, 4499.3 kbytes/s
16.0s - sent: 9011 pkts, 4502.6 kbytes/s
17.0s - sent: 9574 pkts, 4502.4 kbytes/s
^Cpackets sent = 9884, avg rate=3885.2kbps
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/udp#
```

همان طور که مشاهده می شود، مقدار loss حدودا برابر با 26 درصد و مقدار goodput در host h3 برابر با 2900 kbps است.

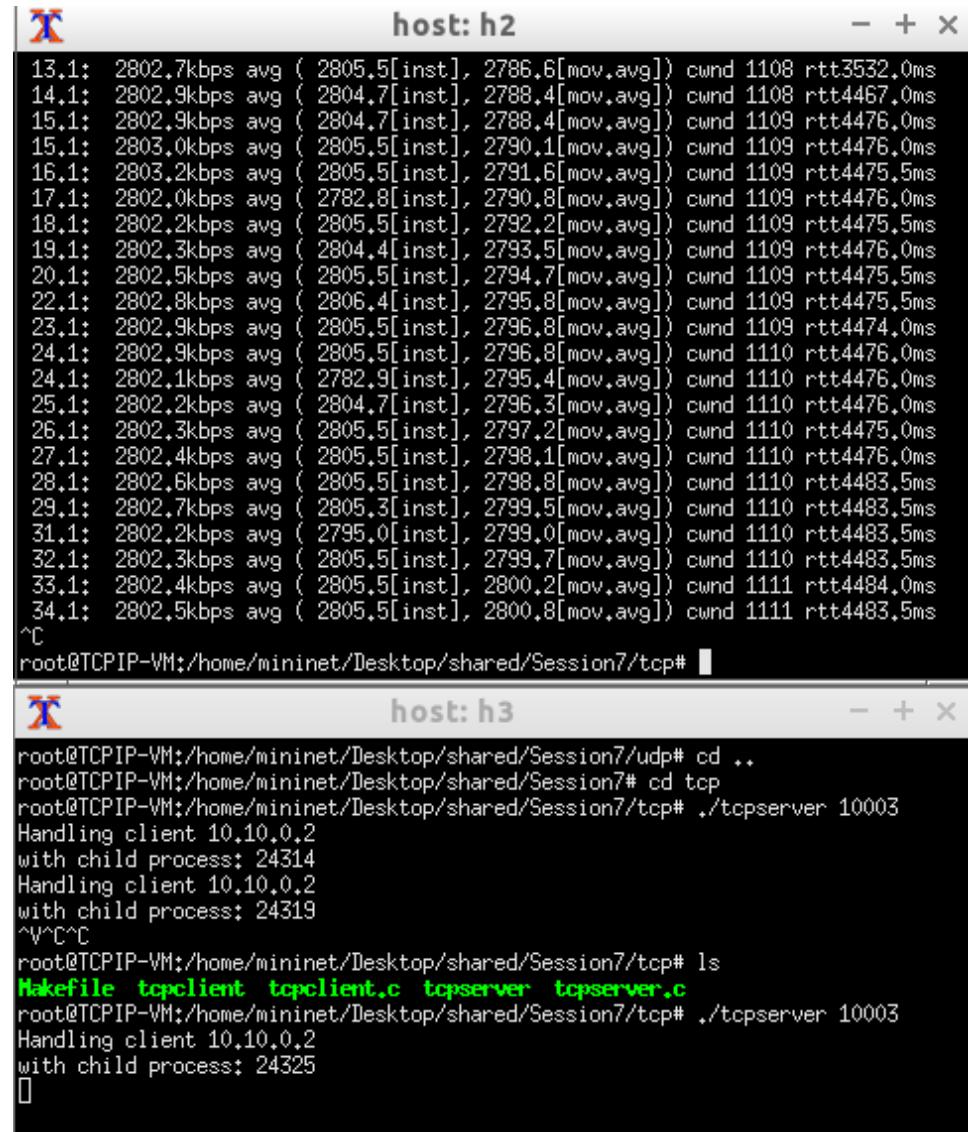
سوال 2: آیا تفاوتی میان این مقادیر تجربی با مقدار تحلیلی مشاهده می کنید؟ اگر بلی، فکر می کنید این تفاوت ها ناشی از چیست؟

تفاوت های کوچکی وجود دارند. بزرگترین این تفاوت ها مقدار loss در بخش آخر است که احتمالا به دلیل وجود باندهای جداگانه است. اما به صورت کلی به مقادیر تئوری و تحلیلی سیار نزدیکند.

ب) رقابت جریان های TCP با جریان های UDP

سوال 3: سناریوهای جدول 3 را مورد آزمایش تجربی قرار دهید. آیا تفاوتی میان این مقادیر تجربی با مقادیر تحلیلی مشاهده می کنید؟ اگر بلی، فکر می کنید این تفاوت ها ناشی از چیست؟

برای این قسمت ابتدا یک سرور TCP روی پورت 10003 در host h3 ایجاد می کنیم. همچنین یک TCP Client روی h2 ایجاد کرده که با آن به TCP Server بسته ارسال می کنیم. سناریوهای موجود در جدول را اجرا کرده تا نتایج را بدست آورده و با مقادیر تئوری مقایسه کنیم.



The image shows two terminal windows. The top window, titled 'host: h2', displays a list of network traffic entries. Each entry shows a timestamp, bandwidth (e.g., 2802.7 kbps), average (avg), and a list of instances (e.g., 2805.5[inst], 2786.6[mov,avg]). The bottom window, titled 'host: h3', shows a server log. It starts with 'root@TCPPIP-VM:~/home/mininet/Desktop/shared/Session7/udp# cd ..' and 'root@TCPPIP-VM:~/home/mininet/Desktop/shared/Session7# cd tcp'. It then runs 'root@TCPPIP-VM:~/home/mininet/Desktop/shared/Session7/tcp# ./tcpserver 10003'. The log then shows 'Handling client 10.10.0.2' and 'with child process: 24314'. It repeats this for another client. The log ends with '^V^C^C' and 'root@TCPPIP-VM:~/home/mininet/Desktop/shared/Session7/tcp# ls'.

```

host: h2
13.1: 2802.7kbps avg ( 2805.5[inst], 2786.6[mov,avg]) cwnd 1108 rtt3832.0ms
14.1: 2802.9kbps avg ( 2804.7[inst], 2788.4[mov,avg]) cwnd 1108 rtt4467.0ms
15.1: 2802.9kbps avg ( 2804.7[inst], 2788.4[mov,avg]) cwnd 1109 rtt4476.0ms
15.1: 2803.0kbps avg ( 2805.5[inst], 2790.1[mov,avg]) cwnd 1109 rtt4476.0ms
16.1: 2803.2kbps avg ( 2805.5[inst], 2791.6[mov,avg]) cwnd 1109 rtt4475.5ms
17.1: 2802.0kbps avg ( 2782.8[inst], 2790.8[mov,avg]) cwnd 1109 rtt4476.0ms
18.1: 2802.2kbps avg ( 2805.5[inst], 2792.2[mov,avg]) cwnd 1109 rtt4475.5ms
19.1: 2802.3kbps avg ( 2804.4[inst], 2793.5[mov,avg]) cwnd 1109 rtt4476.0ms
20.1: 2802.5kbps avg ( 2805.5[inst], 2794.7[mov,avg]) cwnd 1109 rtt4475.5ms
22.1: 2802.8kbps avg ( 2806.4[inst], 2795.8[mov,avg]) cwnd 1109 rtt4475.5ms
23.1: 2802.9kbps avg ( 2805.5[inst], 2796.8[mov,avg]) cwnd 1109 rtt4474.0ms
24.1: 2802.9kbps avg ( 2805.5[inst], 2796.8[mov,avg]) cwnd 1110 rtt4476.0ms
24.1: 2802.1kbps avg ( 2782.9[inst], 2795.4[mov,avg]) cwnd 1110 rtt4476.0ms
25.1: 2802.2kbps avg ( 2804.7[inst], 2796.3[mov,avg]) cwnd 1110 rtt4476.0ms
26.1: 2802.3kbps avg ( 2805.5[inst], 2797.2[mov,avg]) cwnd 1110 rtt4475.0ms
27.1: 2802.4kbps avg ( 2805.5[inst], 2798.1[mov,avg]) cwnd 1110 rtt4476.0ms
28.1: 2802.6kbps avg ( 2805.5[inst], 2798.8[mov,avg]) cwnd 1110 rtt4483.5ms
29.1: 2802.7kbps avg ( 2805.3[inst], 2799.5[mov,avg]) cwnd 1110 rtt4483.5ms
31.1: 2802.2kbps avg ( 2795.0[inst], 2799.0[mov,avg]) cwnd 1110 rtt4483.5ms
32.1: 2802.3kbps avg ( 2805.5[inst], 2799.7[mov,avg]) cwnd 1110 rtt4483.5ms
33.1: 2802.4kbps avg ( 2805.5[inst], 2800.2[mov,avg]) cwnd 1111 rtt4484.0ms
34.1: 2802.5kbps avg ( 2805.5[inst], 2800.8[mov,avg]) cwnd 1111 rtt4483.5ms
^C
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7/tcp#
```

```

host: h3
root@TCPPIP-VM:~/home/mininet/Desktop/shared/Session7/udp# cd ..
root@TCPPIP-VM:~/home/mininet/Desktop/shared/Session7# cd tcp
root@TCPPIP-VM:~/home/mininet/Desktop/shared/Session7/tcp# ./tcpserver 10003
Handling client 10.10.0.2
with child process: 24314
Handling client 10.10.0.2
with child process: 24319
^V^C^C
root@TCPPIP-VM:~/home/mininet/Desktop/shared/Session7/tcp# ls
Makefile tcpclient tcpclient.c tcpserver tcpserver.c
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7/tcp# ./tcpserver 10003
Handling client 10.10.0.2
with child process: 24325
[]
```

آزمایش را با نرخ های host h2 4.5 Mbps و 2 Mbps و 1 Mbps و host h1 1 Mbps و 1 Mbps انجام می دهیم:

آزمایش را با نرخ 1 Mbps ارسال را انجام می دهیم:

**X** Node: h3 - + ×

101.6s - received: 380/	sent: 380	pkts (loss 0.000%), 1016.8 kbit/s
102.6s - received: 498/	sent: 498	pkts (loss 0.000%), 925.1 kbit/s
103.6s - received: 557/	sent: 557	pkts (loss 0.000%), 470.4 kbit/s
104.6s - received: 619/	sent: 619	pkts (loss 0.000%), 493.0 kbit/s
105.6s - received: 682/	sent: 682	pkts (loss 0.000%), 498.2 kbit/s
106.6s - received: 745/	sent: 745	pkts (loss 0.000%), 490.8 kbit/s
107.7s - received: 809/	sent: 809	pkts (loss 0.000%), 503.2 kbit/s
108.7s - received: 872/	sent: 872	pkts (loss 0.000%), 496.6 kbit/s
109.7s - received: 936/	sent: 936	pkts (loss 0.000%), 503.4 kbit/s
110.7s - received: 999/	sent: 999	pkts (loss 0.000%), 497.3 kbit/s
111.7s - received: 1062/	sent: 1062	pkts (loss 0.000%), 497.9 kbit/s
112.7s - received: 1115/	sent: 1154	pkts (loss 3.380%), 420.5 kbit/s
113.8s - received: 1166/	sent: 1266	pkts (loss 7.899%), 396.4 kbit/s
114.8s - received: 1225/	sent: 1381	pkts (loss 11.296%), 466.1 kbit/s
115.8s - received: 1295/	sent: 1507	pkts (loss 14.068%), 557.6 kbit/s
116.8s - received: 1367/	sent: 1633	pkts (loss 16.289%), 571.4 kbit/s
117.8s - received: 1446/	sent: 1799	pkts (loss 19.622%), 625.1 kbit/s
118.8s - received: 1535/	sent: 2001	pkts (loss 23.288%), 707.6 kbit/s
119.8s - received: 1732/	sent: 2216	pkts (loss 21.841%), 1575.1 kbit/s
120.8s - received: 1907/	sent: 2397	pkts (loss 20.442%), 1398.5 kbit/s
121.8s - received: 2002/	sent: 2523	pkts (loss 20.650%), 758.0 kbit/s
122.8s - received: 2163/	sent: 2684	pkts (loss 19.411%), 1285.1 kbit/s
packet received = 2228 / 2749 sent: 18.952% loss		

█

host: h1

```
10001 1000
 1.0s - sent: 126 pkts, 1006.6 kbytes/s
 2.0s - sent: 252 pkts, 1002.3 kbytes/s
 3.0s - sent: 377 pkts, 1000.0 kbytes/s
 4.0s - sent: 503 pkts, 1000.2 kbytes/s
 5.0s - sent: 629 pkts, 1000.4 kbytes/s
 6.0s - sent: 755 pkts, 1000.7 kbytes/s
 7.0s - sent: 881 pkts, 1000.4 kbytes/s
 8.0s - sent: 1007 pkts, 999.8 kbytes/s
 9.1s - sent: 1133 pkts, 1001.4 kbytes/s
10.1s - sent: 1259 pkts, 999.8 kbytes/s
11.1s - sent: 1385 pkts, 1001.3 kbytes/s
12.1s - sent: 1511 pkts, 1000.7 kbytes/s
13.1s - sent: 1637 pkts, 1000.3 kbytes/s
14.1s - sent: 1763 pkts, 1000.5 kbytes/s
15.1s - sent: 1889 pkts, 999.2 kbytes/s
16.1s - sent: 2015 pkts, 1001.9 kbytes/s
17.1s - sent: 2141 pkts, 1001.1 kbytes/s
18.1s - sent: 2267 pkts, 1000.4 kbytes/s
19.1s - sent: 2393 pkts, 1000.3 kbytes/s
20.1s - sent: 2519 pkts, 1000.2 kbytes/s
21.2s - sent: 2645 pkts, 1000.5 kbytes/s
*packets sent = 2749, avg rate= 892.6kbps
root@TCPPIP-WM:/home/mininet/Desktop/shared/Session7/udp#
```

نرخ 1 Mbps برای host h2

```

          Node: h3

101.6s - received: 389/ sent: 389 pkts (loss: 0.000%), 1016.8 kbit/s
102.1s - received: 453/ sent: 453 pkts (loss: 0.000%), 1021.4 kbit/s
102.6s - received: 557/ sent: 557 pkts (loss: 0.000%), 1020.4 kbit/s
104.6s - received: 619/ sent: 619 pkts (loss: 0.000%), 1043.0 kbit/s
105.6s - received: 682/ sent: 682 pkts (loss: 0.000%), 1052.2 kbit/s
106.6s - received: 745/ sent: 745 pkts (loss: 0.000%), 1062.0 kbit/s
107.7s - received: 809/ sent: 809 pkts (loss: 0.000%), 1072.4 kbit/s
108.7s - received: 872/ sent: 872 pkts (loss: 0.000%), 1082.6 kbit/s
109.7s - received: 935/ sent: 935 pkts (loss: 0.000%), 1093.4 kbit/s
110.7s - received: 1000/ sent: 1000 pkts (loss: 0.000%), 1103.2 kbit/s
111.7s - received: 1062/ sent: 1062 pkts (loss: 0.000%), 1073.9 kbit/s
112.7s - received: 1115/ sent: 1154 pkts (loss: 3.300%), 1103.5 kbit/s
113.8s - received: 1165/ sent: 1165 pkts (loss: 0.789%), 1114.6 kbit/s
114.8s - received: 1218/ sent: 1218 pkts (loss: 0.000%), 1125.4 kbit/s
115.8s - received: 1270/ sent: 1270 pkts (loss: 0.000%), 1136.0 kbit/s
116.8s - received: 1367/ sent: 1633 pkts (loss: 16.28%), 1571.4 kbit/s
117.8s - received: 1448/ sent: 1793 pkts (loss: 19.622%), 6251.1 kbit/s
118.8s - received: 1529/ sent: 2041 pkts (loss: 20.441%), 7501.1 kbit/s
119.8s - received: 1720/ sent: 2232 pkts (loss: 21.841%), 7571.8 kbit/s
120.8s - received: 1997/ sent: 2397 pkts (loss: 20.442%), 13981.1 kbit/s
121.8s - received: 2002/ sent: 2523 pkts (loss: 20.650%), 7900.7 kbit/s
122.8s - received: 2163/ sent: 2684 pkts (loss: 19.411%), 12891.1 kbit/s
packets received = 2226 / 2743 sent: 18,952 loss

```

```

X Node: h3 + x
69.5s - received: 33/ sent: 333 pkts (loss: 0.00%), 631.9 kbit/s
70.5s - received: 359/ sent: 339 pkts (loss: 0.00%), 622.0 kbit/s
71.5s - received: 524/ sent: 524 pkts (loss: 0.00%), 594.0 kbit/s
72.5s - received: 524/ sent: 524 pkts (loss: 0.00%), 594.0 kbit/s
73.5s - received: 588/ sent: 588 pkts (loss: 0.00%), 505.1 kbit/s
74.5s - received: 649/ sent: 649 pkts (loss: 0.00%), 497.6 kbit/s
75.5s - received: 776/ sent: 776 pkts (loss: 0.00%), 498.1 kbit/s
76.5s - received: 776/ sent: 776 pkts (loss: 0.00%), 498.1 kbit/s
77.5s - received: 839/ sent: 839 pkts (loss: 0.00%), 497.6 kbit/s
78.5s - received: 859/ sent: 977 pkts (loss: 0.00%), 495.7 kbit/s
79.5s - received: 859/ sent: 977 pkts (loss: 0.00%), 495.7 kbit/s
80.5s - received: 964/ sent: 1130 pkts (loss: 14.53%), 276.0 kbit/s
81.7s - received: 1014/ sent: 1253 pkts (loss: 14.53%), 395.8 kbit/s
82.7s - received: 1071/ sent: 1379 pkts (loss: 22.53%), 451.7 kbit/s
83.7s - received: 1130/ sent: 1496 pkts (loss: 22.53%), 451.7 kbit/s
84.8s - received: 1188/ sent: 1717 pkts (loss: 31.57%), 357.4 kbit/s
85.8s - received: 1239/ sent: 1924 pkts (loss: 34.93%), 725.5 kbit/s
86.8s - received: 1409/ sent: 2144 pkts (loss: 34.93%), 1184.9 kbit/s
87.8s - received: 1409/ sent: 2144 pkts (loss: 34.93%), 1184.9 kbit/s
88.8s - received: 1590/ sent: 2413 pkts (loss: 34.10%), 798.6 kbit/s
89.8s - received: 1677/ sent: 2950 pkts (loss: 34.77%), 1415.9 kbit/s
90.8s - received: 1881/ sent: 2684 pkts (loss: 30.66%), 750.1 kbit/s
packet received = 1936 / 2019 sent, 29.1951 loss

```

```
host: h3
root@TCPPIP-Wi:~$ /home/mininet/Desktop/shared/Session7/tcpserver 10003
Handling client 10.0.0.2
with child process: 24392
```

```

X host: h2
0.0: 0.0kbps avg ( 0.0[inst], 0.0[avg,avg]) cwnd 16 rtt 9.0ms
0.2: 0.0kbps avg ( 0.0[inst], 0.0[avg,avg]) cwnd 49 rtt 85.0ms
0.4: 0.0kbps avg ( 0.0[inst], 0.0[avg,avg]) cwnd 102 rtt 144.0ms
0.5: 0.0kbps avg ( 0.0[inst], 0.0[avg,avg]) cwnd 162 rtt 1444.0ms
1.3: 195.3kbps avg ( 195.3[inst], 195.3[avg,avg]) cwnd 239 rtt 95.0ms
2.5: 188.1kbps avg ( 188.1[inst], 188.1[avg,avg]) cwnd 388 rtt 114.0ms
3.5: 188.1kbps avg ( 188.1[inst], 188.1[avg,avg]) cwnd 537 rtt 114.0ms
5.3: 188.1kbps avg ( 188.1[inst], 188.1[avg,avg]) cwnd 592 rtt 185.0ms
4.3: 188.1kbps avg ( 188.1[inst], 188.1[avg,avg]) cwnd 737 rtt 215.0ms
5.3: 188.1kbps avg ( 188.1[inst], 188.1[avg,avg]) cwnd 879 rtt 356.0ms
5.5: 188.1kbps avg ( 188.1[inst], 188.1[avg,avg]) cwnd 927 rtt 356.0ms
5.9: 188.1kbps avg ( 188.1[inst], 188.1[avg,avg]) cwnd 974 rtt 356.0ms
6.5: 188.1kbps avg ( 188.1[inst], 188.1[avg,avg]) cwnd 1108 rtt 345.0ms
7.9: 184.0kbps avg ( 184.0[inst], 184.0[avg,avg]) cwnd 1108 rtt 390.0ms
8.5: 184.0kbps avg ( 184.0[inst], 184.0[avg,avg]) cwnd 1108 rtt 390.0ms
9.9: 178.8kbps avg ( 178.8[inst], 178.8[avg,avg]) cwnd 1108 rtt 390.0ms
10.9: 165.3kbps avg ( 0.0[inst], 165.3[avg,avg]) cwnd 885 rtt 504.0ms
11.9: 146.2kbps avg ( 0.0[inst], 146.2[avg,avg]) cwnd 698 rtt 575.0ms
12.9: 146.2kbps avg ( 0.0[inst], 146.2[avg,avg]) cwnd 692 rtt 575.0ms
13.9: 146.2kbps avg ( 0.0[inst], 146.2[avg,avg]) cwnd 682 rtt 575.0ms

```

## نرخ 2 Mbps برای h2

```

host: h3
Node: h3
90.8s - received: 2163 sent: 2684 pkts (loss 19.411%), 1285.1 kbit/s
packet received = 2228 / 2743 sent: 18.592k loss
123.8s - received: 1274 sent: 1511 pkts (loss 0.000%), 5.9 kbit/s
239.9s - received: 1274 sent: 1511 pkts (loss 0.000%), 1001.5 kbit/s
260.2s - received: 240* sent: 240 pkts (loss 0.000%), 694.7 kbit/s
261.2s - received: 347* sent: 347 pkts (loss 0.000%), 695.4 kbit/s
262.2s - received: 347* sent: 347 pkts (loss 0.000%), 695.4 kbit/s
263.2s - received: 654* sent: 954 pkts (loss 0.000%), 702.2 kbit/s
264.2s - received: 639* sent: 639 pkts (loss 0.000%), 645.8 kbit/s
265.2s - received: 714* sent: 714 pkts (loss 0.000%), 625.7 kbit/s
266.2s - received: 731* sent: 731 pkts (loss 0.000%), 625.7 kbit/s
267.2s - received: 667* sent: 657 pkts (loss 0.000%), 520.0 kbit/s
268.2s - received: 920* sent: 920 pkts (loss 0.000%), 905.2 kbit/s
269.2s - received: 100* sent: 100 pkts (loss 0.000%), 794.0 kbit/s
270.2s - received: 100* sent: 100 pkts (loss 0.000%), 794.0 kbit/s
271.2s - received: 117* sent: 117 pkts (loss 1.68%), 606.2 kbit/s
272.2s - received: 123* sent: 123 pkts (loss 1.68%), 603.5 kbit/s
273.2s - received: 123* sent: 123 pkts (loss 1.68%), 603.5 kbit/s
274.2s - received: 143* sent: 1582 pkts (loss 9.45%), 681.0 kbit/s
275.2s - received: 152* sent: 1723 pkts (loss 11.79%), 700.5 kbit/s
276.2s - received: 163* sent: 1873 pkts (loss 12.54%), 902.0 kbit/s
277.2s - received: 163* sent: 1873 pkts (loss 14.11%), 762.5 kbit/s
packet received = 1819 / 2142 sent: 15.07% loss
host: h1
Node: h1
20.1s - sent: 2619 pkts, 1000.2 kbit/s
21.0s - sent: 2645 pkts, 1000.5 kbit/s
^Packets sent = 2749, avg rate= 892.6kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10001 2000
1.0s - sent: 126 pkts, 1000.0 kbit/s
2.0s - sent: 254 pkts, 998.5 kbit/s
3.0s - sent: 251 pkts, 1000.0 kbit/s
4.0s - sent: 503 pkts, 1000.0 kbit/s
5.0s - sent: 623 pkts, 1001.2 kbit/s
6.0s - sent: 795 pkts, 998.8 kbit/s
7.0s - sent: 1008 pkts, 1000.0 kbit/s
8.0s - sent: 1008 pkts, 1000.4 kbit/s
9.0s - sent: 1134 pkts, 1000.4 kbit/s
10.0s - sent: 1398 pkts, 1000.5 kbit/s
11.0s - sent: 1511 pkts, 998.5 kbit/s
12.0s - sent: 1637 pkts, 1001.2 kbit/s
13.0s - sent: 1883 pkts, 1000.4 kbit/s
14.0s - sent: 1883 pkts, 1000.4 kbit/s
15.0s - sent: 2015 pkts, 998.5 kbit/s
16.0s - sent: 2015 pkts, 1000.2 kbit/s
17.0s - sent: 2142 pkts, 1000.4 kbit/s
^Packets sent = 2142, avg rate= 1000.4kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# []
host: h3
Node: h3
241s - sent: 2544 pkts, 1000.0 kbit/s
22.2s - sent: 2770 pkts, 1000.0 kbit/s
^Packets sent = 2819, avg rate= 901.1kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10001 2000
1.0s - sent: 251 pkts, 2001.5 kbit/s
2.0s - sent: 509 pkts, 1963.2 kbit/s
3.0s - sent: 797 pkts, 2052.3 kbit/s
4.0s - sent: 1239 pkts, 2002.6 kbit/s
5.0s - sent: 1239 pkts, 2002.6 kbit/s
6.0s - sent: 1510 pkts, 2002.4 kbit/s
7.0s - sent: 1510 pkts, 2002.4 kbit/s
8.0s - sent: 2018 pkts, 2000.2 kbit/s
9.0s - sent: 2253 pkts, 2003.3 kbit/s
10.0s - sent: 2514 pkts, 2000.0 kbit/s
11.0s - sent: 2514 pkts, 2000.0 kbit/s
12.0s - sent: 3015 pkts, 1998.4 kbit/s
13.0s - sent: 3267 pkts, 2010.0 kbit/s
14.0s - sent: 3538 pkts, 2002.0 kbit/s
15.0s - sent: 3538 pkts, 2002.0 kbit/s
16.0s - sent: 4020 pkts, 2001.8 kbit/s
17.0s - sent: 4271 pkts, 2001.8 kbit/s
^Packets sent = 4337, avg rate=1729.7kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# []
host: h2
Node: h2
241s - sent: 2544 pkts, 1000.0 kbit/s
22.2s - sent: 2770 pkts, 1000.0 kbit/s
^Packets sent = 2819, avg rate= 901.1kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10001 2000
0.0s - 0.0kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 10 rtt3294.0ms
0.6s - 0.0kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 14 rtt3295.0ms
1.0s - 352.0kbit/s avg ( 352.0[minew,avg]) cmd 32 rtt3243.0ms
2.0s - 392.0kbit/s avg ( 392.0[minew,avg]) cmd 33 rtt3244.0ms
3.0s - 392.0kbit/s avg ( 392.0[minew,avg]) cmd 34 rtt3253.0ms
4.0s - 392.0kbit/s avg ( 392.0[minew,avg]) cmd 35 rtt3254.0ms
5.0s - 650.0kbit/s avg ( 650.0[minew,avg]) cmd 36 rtt1065.0ms
6.0s - 650.0kbit/s avg ( 650.0[minew,avg]) cmd 37 rtt1066.0ms
7.0s - 650.0kbit/s avg ( 650.0[minew,avg]) cmd 38 rtt1112.0ms
8.0s - 650.0kbit/s avg ( 650.0[minew,avg]) cmd 39 rtt1113.0ms
9.0s - 832.0kbit/s avg ( 832.0[minew,avg]) cmd 40 rtt1954.0ms
10.0s - 867.0kbit/s avg ( 867.0[minew,avg]) cmd 41 rtt2071.0ms
11.0s - 867.0kbit/s avg ( 867.0[minew,avg]) cmd 42 rtt2072.0ms
12.0s - 886.0kbit/s avg ( 886.0[minew,avg]) cmd 43 rtt3094.0ms
13.0s - 886.0kbit/s avg ( 886.0[minew,avg]) cmd 44 rtt3095.0ms
14.0s - 886.0kbit/s avg ( 886.0[minew,avg]) cmd 45 rtt3096.0ms
15.0s - 886.0kbit/s avg ( 886.0[minew,avg]) cmd 46 rtt3097.0ms
16.0s - 886.0kbit/s avg ( 886.0[minew,avg]) cmd 47 rtt3098.0ms
17.0s - 886.0kbit/s avg ( 886.0[minew,avg]) cmd 48 rtt3099.0ms
^C
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/tcp# []

```

## نرخ 4.5 Mbps برای h2

```

host: h3
Node: h3
packet received = 1819 / 2142 sent: 15.07% loss
278.3s - received: 1264 sent: 1511 pkts (loss 0.000%), 41.5 kbit/s
279.3s - received: 1274 sent: 1511 pkts (loss 0.000%), 1002.7 kbit/s
339.4s - received: 224* sent: 224 pkts (loss 0.000%), 768.4 kbit/s
340.4s - received: 123* sent: 123 pkts (loss 0.000%), 623.1 kbit/s
341.4s - received: 122* sent: 122 pkts (loss 0.000%), 621.0 kbit/s
342.4s - received: 122* sent: 122 pkts (loss 0.000%), 621.0 kbit/s
343.4s - received: 187* sent: 187 pkts (loss 0.000%), 513.5 kbit/s
344.4s - received: 651* sent: 551 pkts (loss 0.000%), 498.0 kbit/s
345.4s - received: 651* sent: 551 pkts (loss 0.000%), 498.0 kbit/s
346.4s - received: 680* sent: 789 pkts (loss 15.81%), 635.0 kbit/s
347.4s - received: 721* sent: 911 pkts (loss 20.85%), 525.2 kbit/s
348.4s - received: 721* sent: 911 pkts (loss 20.85%), 525.2 kbit/s
349.4s - received: 937* sent: 1434 pkts (loss 41.77%), 646.7 kbit/s
350.4s - received: 937* sent: 1434 pkts (loss 41.77%), 646.7 kbit/s
351.4s - received: 967* sent: 1568 pkts (loss 39.85%), 623.3 kbit/s
352.4s - received: 100* sent: 100 pkts (loss 0.000%), 794.0 kbit/s
353.4s - received: 100* sent: 100 pkts (loss 0.000%), 794.0 kbit/s
354.4s - received: 100* sent: 100 pkts (loss 0.000%), 794.0 kbit/s
355.4s - received: 120* sent: 120 pkts (loss 57.49%), 415.6 kbit/s
356.4s - received: 120* sent: 120 pkts (loss 57.49%), 415.6 kbit/s
357.4s - received: 1513* sent: 1882 pkts (loss 39.64%), 647.4 kbit/s
packet received = 1824 / 2199 sent: 33.79% loss
host: h1
Node: h1
16.1s - sent: 2015 pkts, 995.6 kbit/s
17.1s - sent: 2141 pkts, 1001.2 kbit/s
^Packets sent = 2142, avg rate= 1000.5kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10001 2000
1.0s - sent: 125 pkts, 1000.0 kbit/s
2.0s - sent: 127 pkts, 1000.0 kbit/s
3.0s - sent: 528 pkts, 1000.5 kbit/s
4.0s - sent: 501 pkts, 1001.0 kbit/s
5.0s - sent: 629 pkts, 998.8 kbit/s
6.0s - sent: 1007 pkts, 1000.0 kbit/s
7.0s - sent: 881 pkts, 1000.5 kbit/s
8.0s - sent: 1007 pkts, 1000.2 kbit/s
9.0s - sent: 1007 pkts, 1000.5 kbit/s
10.0s - sent: 1250 pkts, 1000.0 kbit/s
11.0s - sent: 1395 pkts, 1000.6 kbit/s
12.0s - sent: 1513 pkts, 1015.5 kbit/s
13.0s - sent: 1765 pkts, 1000.6 kbit/s
14.0s - sent: 1765 pkts, 1000.6 kbit/s
15.0s - sent: 1891 pkts, 1000.5 kbit/s
16.0s - sent: 2014 pkts, 1001.3 kbit/s
17.0s - sent: 2142 pkts, 1000.0 kbit/s
^Packets sent = 2142, avg rate= 881.1kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# []
host: h3
Node: h3
246.4s - received: 1/ sent: 1 pkts (loss 0.000%), 33.3 kbit/s
247.4s - received: 291* sent: 291 pkts (loss 0.000%), 291.0 kbit/s
248.4s - received: 592* sent: 592 pkts (loss 0.000%), 2355.9 kbit/s
249.4s - received: 884* sent: 884 pkts (loss 0.000%), 2335.5 kbit/s
250.4s - received: 1174* sent: 1174 pkts (loss 0.000%), 2315.7 kbit/s
251.4s - received: 1464* sent: 1464 pkts (loss 0.000%), 2295.9 kbit/s
252.4s - received: 1743* sent: 1801 pkts (loss 2.807%), 2259.0 kbit/s
253.4s - received: 2048* sent: 2572 pkts (loss 13.744%), 2254.8 kbit/s
254.4s - received: 2048* sent: 2572 pkts (loss 13.744%), 2254.8 kbit/s
255.4s - received: 2524* sent: 5711 pkts (loss 45.546%), 2274.7 kbit/s
256.4s - received: 2524* sent: 5711 pkts (loss 45.546%), 2274.7 kbit/s
257.4s - received: 3015* sent: 3015 pkts (loss 39.457%), 2249.5 kbit/s
258.4s - received: 4109* sent: 6598 pkts (loss 37.705%), 2338.8 kbit/s
259.4s - received: 4399* sent: 6598 pkts (loss 36.117%), 2319.7 kbit/s
260.4s - received: 4399* sent: 6598 pkts (loss 36.117%), 2319.7 kbit/s
261.4s - received: 5445* sent: 5445 pkts (loss 34.465%), 2299.0 kbit/s
262.4s - received: 5014* sent: 8000 pkts (loss 37.325%), 2453.0 kbit/s
263.4s - received: 5399* sent: 8572 pkts (loss 38.052%), 2357.5 kbit/s
264.4s - received: 5399* sent: 8572 pkts (loss 38.052%), 2357.5 kbit/s
265.4s - received: 5706* sent: 8706 pkts (loss 38.374%), 2320.7 kbit/s
266.4s - received: 5706* sent: 8706 pkts (loss 38.374%), 2320.7 kbit/s
267.4s - received: 5863* sent: 3706 pkts (loss 38.338%), 2639.4 kbit/s
packet received = 3558 / 3822 sent: 38.304% loss
host: h2
Node: h2
16.1s - sent: 4020 pkts, 2001.8 kbit/s
17.1s - sent: 4211 pkts, 2001.8 kbit/s
^Packets sent = 4337, avg rate=1729.7kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# ./udpclient 10.10.1.3
10001 2000
1.0s - sent: 100 pkts, 2001.0 kbit/s
2.0s - sent: 100 pkts, 2001.0 kbit/s
3.0s - sent: 100 pkts, 2001.0 kbit/s
4.0s - sent: 2253 pkts, 4903.4 kbit/s
5.0s - sent: 2817 pkts, 4904.4 kbit/s
6.0s - sent: 4502 pkts, 4902.1 kbit/s
7.0s - sent: 3845 pkts, 4902.2 kbit/s
8.0s - sent: 5062 pkts, 4437.5 kbit/s
9.0s - sent: 5062 pkts, 4437.5 kbit/s
10.0s - sent: 5062 pkts, 4437.5 kbit/s
11.0s - sent: 5502 pkts, 4934.4 kbit/s
12.0s - sent: 6768 pkts, 4501.4 kbit/s
13.0s - sent: 7339 pkts, 4903.4 kbit/s
14.0s - sent: 7339 pkts, 4903.4 kbit/s
15.0s - sent: 8449 pkts, 4374.1 kbit/s
16.0s - sent: 9041 pkts, 4785.7 kbit/s
17.0s - sent: 9041 pkts, 4785.7 kbit/s
^Packets sent = 10033, avg rate=303.2kbit/s
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/udp# []
host: h2
Node: h2
15.3s - 806.0kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 139 rtt3182.0ms
16.3s - 806.0kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 139 rtt3246.0ms
17.3s - 531.0kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 1 rtt2840.0ms
^C
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/tcp# []
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/tcp# ./tcpserver 10003
Handling client 10.10.0.2
with child process: 24392
Handling client 10.10.0.2
with child process: 24401
Handling client 10.10.0.2
with child process: 24409
10001 2000
0.0s - 0.0kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 10 rtt1216.0ms
2.0s - -0.4kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 10 rtt1215.0ms
3.0s - -0.4kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 10 rtt1214.0ms
4.0s - 27.4kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 20 rtt2062.0ms
5.0s - 21.8kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 20 rtt2062.0ms
6.0s - 50.0kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 20 rtt2062.0ms
7.0s - 50.0kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 20 rtt2062.0ms
8.0s - 50.0kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 20 rtt2062.0ms
9.0s - 50.0kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 20 rtt2062.0ms
10.0s - 50.0kbit/s avg ( 0.0[inst], 0.0[minew,avg]) cmd 20 rtt2062.0ms
11.0s - 28.4kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 38 rtt1989.0ms
12.0s - 25.7kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 38 rtt1987.0ms
13.0s - 23.7kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 38 rtt1987.0ms
14.0s - 23.7kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 38 rtt1987.0ms
15.0s - 23.7kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 38 rtt1987.0ms
16.0s - 13.3kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 38 rtt1987.0ms
16.2s - 13.3kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 38 rtt1987.0ms
17.0s - 19.5kbit/s avg ( -0.0[inst], 0.0[minew,avg]) cmd 1 rtt2897.0ms
^C
root@TCP-IP-VH:/home/mininet/Desktop/shared/Session7/tcp# []

```

همان طور که مشاهده می شود، نتایج تفاوت زیادی با مقادیر تئوری ندارند.

## گزارشکار آزمایش 7: بررسی رفتار جریان های TCP و UDP

سیده شکیبا انارکی فیروز - ۹۹۴۴۲۰۴۷

بهاره کاووسی نژاد - ۹۹۴۳۱۲۱۷

ج) بررسی تأثیر مکانیزم «اخطار صریح ازدحام» بر RTT و «پنجره ازدحام» پس از خروج از mininet و پاک کردن توبولوژی قبلی، خط زیر را به کد اضافه کرده و دوباره فایل پایتون را اجرا می کنیم. با اضافه کردن این خط، طول صفحه 1 کاهش می باید و پنهانی باند interface eth1 را برابر با 5 Mbps قرار می دهیم. همچنین مقدار enable\_ecn را نیز False می گذاریم.

```
link_r1sw2.intf1.config(bw=5, max_queue_size=1000, enable_ecn=False)
```

سوال 4: مقدار نرخ ماشین منبع (یعنی h1) چقدر است؟ حدود مقادیر RTT و نیز محدوده مقادیر «پنجره ازدحام» را مشخص نمایید. یک سرور tcp روی h3 و یک کلاینت tcp روی h1 اجرا می کنیم. مقدار نرخ h1 به حدود 4200 kbps می رسد و همچنین مقدار rtt نیز تقریبا 2200 ms خواهد بود. همچنین مقدار پنجره ازدحام از 10 شروع شده و به صورت نمایی افزایش می یابد.

```
host: h3
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7# cd tcp
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7/tcp# ./tcpserver 10000
Handling client 10.10.0.1
with child process: 25731
[]

host: h1
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7/tcp# ./udpcclient 10.10.1.3 10000 1000
bash: ./udpcclient: No such file or directory
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7/tcp# ./tcpclient 10.10.1.3 10000 1000
0.0: 0.0kbps avg ( 0.0[inst], 0.0[mov,avg]) cwnd 10 rtt 16.0ms
0.0: 0.0kbps avg ( 0.0[inst], 0.0[mov,avg]) cwnd 24 rtt 21.0ms
0.1: 0.0kbps avg ( 0.0[inst], 0.0[mov,avg]) cwnd 38 rtt 33.0ms
0.2: 0.0kbps avg ( 0.0[inst], 0.0[mov,avg]) cwnd 96 rtt103.0ms
0.3: 0.0kbps avg ( 0.0[inst], 0.0[mov,avg]) cwnd 138 rtt152.5ms
0.3: 0.0kbps avg ( 0.0[inst], 0.0[mov,avg]) cwnd 140 rtt154.5ms
0.8: 0.0kbps avg ( 0.0[inst], 0.0[mov,avg]) cwnd 352 rtt409.0ms
1.2: 4655.9kbps avg ( 4655.9[inst], 4655.9[mov,avg]) cwnd 496 rtt587.5ms
1.2: 4655.9kbps avg ( 4655.9[inst], 4655.9[mov,avg]) cwnd 498 rtt589.5ms
1.2: 4655.9kbps avg ( 4655.9[inst], 4655.9[mov,avg]) cwnd 500 rtt592.0ms
1.2: 4655.9kbps avg ( 4655.9[inst], 4655.9[mov,avg]) cwnd 502 rtt594.0ms
1.2: 4655.9kbps avg ( 4655.9[inst], 4655.9[mov,avg]) cwnd 504 rtt597.0ms
1.2: 4655.9kbps avg ( 4655.9[inst], 4655.9[mov,avg]) cwnd 506 rtt599.0ms
2.2: 4657.5kbps avg ( 4659.3[inst], 4659.3[mov,avg]) cwnd 918 rtt1096.0ms
2.2: 4657.5kbps avg ( 4659.3[inst], 4659.3[mov,avg]) cwnd 920 rtt1099.0ms
3.2: 4658.3kbps avg ( 4660.0[inst], 4660.0[mov,avg]) cwnd 1332 rtt1597.0ms
3.2: 4658.3kbps avg ( 4660.0[inst], 4660.0[mov,avg]) cwnd 1336 rtt1606.5ms
3.2: 4658.3kbps avg ( 4660.0[inst], 4660.0[mov,avg]) cwnd 1338 rtt1609.5ms
4.3: 4275.0kbps avg ( 3138.4[inst], 3138.4[mov,avg]) cwnd 1632 rtt2251.0ms
```

```

host: h3
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# cd tcp
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/tcp# ./tcpserver 10000
Handling client 10.10.0.1
with child process: 25731
[]

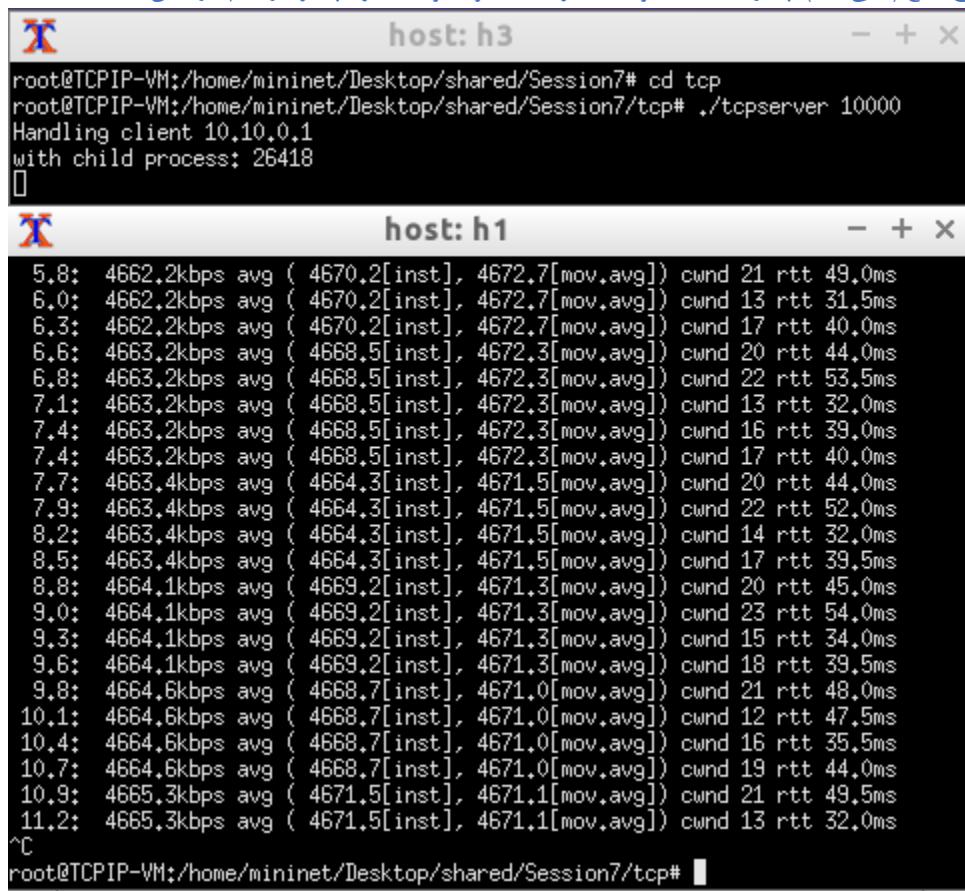
host: h1
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# ./tcpclient 10000
34.8: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 610 rtt3633.0ms
35.5: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 501 rtt2609.5ms
35.5: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 500 rtt2604.0ms
35.5: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 497 rtt2590.0ms
35.5: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 495 rtt2581.0ms
35.5: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 494 rtt2578.5ms
35.5: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 493 rtt2571.5ms
35.5: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 492 rtt2569.5ms
35.5: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 490 rtt2559.0ms
35.6: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 487 rtt2541.0ms
35.6: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 482 rtt2525.5ms
35.6: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 475 rtt2508.5ms
35.7: 2552.3kbps avg (12408.7[inst], 3393.9[mov,avg]) cwnd 465 rtt2479.5ms
35.8: 2509.8kbps avg ( 1050.6[inst], 3159.6[mov,avg]) cwnd 452 rtt2412.0ms
35.8: 2509.8kbps avg ( 1050.6[inst], 3159.6[mov,avg]) cwnd 449 rtt2397.5ms
35.8: 2509.8kbps avg ( 1050.6[inst], 3159.6[mov,avg]) cwnd 448 rtt2388.5ms
35.9: 2509.8kbps avg ( 1050.6[inst], 3159.6[mov,avg]) cwnd 431 rtt2341.5ms
36.0: 2509.8kbps avg ( 1050.6[inst], 3159.6[mov,avg]) cwnd 410 rtt2283.0ms
37.1: 2444.4kbps avg ( 624.7[inst], 2906.1[mov,avg]) cwnd 305 rtt1306.5ms
37.1: 2444.4kbps avg ( 624.7[inst], 2906.1[mov,avg]) cwnd 306 rtt1231.0ms
38.6: 2597.3kbps avg ( 6290.6[inst], 3244.5[mov,avg]) cwnd 306 rtt1362.5ms
39.8: 2637.2kbps avg ( 3950.0[inst], 3315.1[mov,avg]) cwnd 306 rtt817.5ms
^C
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/tcp# []

```

مقدار enable\\_ecn را برابر با True قرار داده و تپولوژی را پاکسازی کرده و دوباره فایل پایتون را اجرا می کنیم.

```
link_r1sw2.intf1.config(bw=5, max_queue_size=1000, enable_ecn=True)
```

سوال 5: مقدار نرخ منبع (یعنی  $h_1$ ) چقدر است؟ محدوده مقادیر RTT و حدود مقادیر «پنجره ازدحام» را بیان کنید.



The image shows two terminal windows. The top window, titled 'host: h3', displays a TCP server log: 'root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# cd tcp', 'root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/tcp# ./tcpserver 10000', 'Handling client 10.10.0.1', and 'with child process: 26418'. The bottom window, titled 'host: h1', displays a TCP client log showing a sequence of 4660 kbps transmissions over 11.2 seconds, followed by a '^C' interrupt and a return to the prompt.

```
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# cd tcp
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/tcp# ./tcpserver 10000
Handling client 10.10.0.1
with child process: 26418
[]

root@TCPIP-VM:/home/mininet/Desktop/shared/Session7# host: h1
5.8: 4662.2kbps avg ( 4670.2[inst], 4672.7[mov,avg]) cwnd 21 rtt 49.0ms
6.0: 4662.2kbps avg ( 4670.2[inst], 4672.7[mov,avg]) cwnd 13 rtt 31.5ms
6.3: 4662.2kbps avg ( 4670.2[inst], 4672.7[mov,avg]) cwnd 17 rtt 40.0ms
6.6: 4663.2kbps avg ( 4668.5[inst], 4672.3[mov,avg]) cwnd 20 rtt 44.0ms
6.8: 4663.2kbps avg ( 4668.5[inst], 4672.3[mov,avg]) cwnd 22 rtt 53.5ms
7.1: 4663.2kbps avg ( 4668.5[inst], 4672.3[mov,avg]) cwnd 13 rtt 32.0ms
7.4: 4663.2kbps avg ( 4668.5[inst], 4672.3[mov,avg]) cwnd 16 rtt 39.0ms
7.4: 4663.2kbps avg ( 4668.5[inst], 4672.3[mov,avg]) cwnd 17 rtt 40.0ms
7.7: 4663.4kbps avg ( 4664.3[inst], 4671.5[mov,avg]) cwnd 20 rtt 44.0ms
7.9: 4663.4kbps avg ( 4664.3[inst], 4671.5[mov,avg]) cwnd 22 rtt 52.0ms
8.2: 4663.4kbps avg ( 4664.3[inst], 4671.5[mov,avg]) cwnd 14 rtt 32.0ms
8.5: 4663.4kbps avg ( 4664.3[inst], 4671.5[mov,avg]) cwnd 17 rtt 39.5ms
8.8: 4664.1kbps avg ( 4669.2[inst], 4671.3[mov,avg]) cwnd 20 rtt 45.0ms
9.0: 4664.1kbps avg ( 4669.2[inst], 4671.3[mov,avg]) cwnd 23 rtt 54.0ms
9.3: 4664.1kbps avg ( 4669.2[inst], 4671.3[mov,avg]) cwnd 15 rtt 34.0ms
9.6: 4664.1kbps avg ( 4669.2[inst], 4671.3[mov,avg]) cwnd 18 rtt 39.5ms
9.8: 4664.6kbps avg ( 4668.7[inst], 4671.0[mov,avg]) cwnd 21 rtt 48.0ms
10.1: 4664.6kbps avg ( 4668.7[inst], 4671.0[mov,avg]) cwnd 12 rtt 47.5ms
10.4: 4664.6kbps avg ( 4668.7[inst], 4671.0[mov,avg]) cwnd 16 rtt 35.5ms
10.7: 4664.6kbps avg ( 4668.7[inst], 4671.0[mov,avg]) cwnd 19 rtt 44.0ms
10.9: 4665.3kbps avg ( 4671.5[inst], 4671.1[mov,avg]) cwnd 21 rtt 49.5ms
11.2: 4665.3kbps avg ( 4671.5[inst], 4671.1[mov,avg]) cwnd 13 rtt 32.0ms
^C
root@TCPIP-VM:/home/mininet/Desktop/shared/Session7/tcp#
```

مطابق نتایج نشان داده شده در تصویر، مقدار نرخ  $h_1$  تقریباً برابر با 4660 kbps است؛ مقدار  $rtt$  4660 ms مطابق نتایج نشان داده شده در تصویر، مقدار نرخ  $h_1$  تقریباً برابر با 40 ms مطابق نتایج نشان داده شده در تصویر است.

همچنین مقدار پنجره ازدحام بین 13 تا 23 متغیر می باشد.

سوال 6: با مقایسه مقادیر مشاهده شده در سوال 5 با مقادیری که نظیر حالت ECN غیرفعال هستند (سوال 4)، چه نتیجه ای می توان گرفت؟

با مقایسه مقادیر دو سوال می توان به این نتیجه رسید که اگر مقدار enable\_ecn برای Router (interface eth1) True باشد، برای  $rtt$  باشد، برای  $cwnd$  باشد، برای  $ecn$  باشد. فرستنده قبل از به وجود آمدن ازدحام در شبکه و بعد از آن، متوجه drop شدن بسته ها می شود و پنجره ارسال را مطابق با آن تنظیم می کند. به همین دلیل، مقدار  $rtt$  و  $cwnd$  به صورت قابل ملاحظه ای کاهش پیدا می کنند؛ اما در صورتیکه مقدار enable\_ecn برای Router False باشد، مقدار  $cwnd$  و  $rtt$  به شدت افزایش می یابند.

د) عدالت در TCP و تاثیر RTT

د-1) افزودن تأخیر به یک اینترفیس

سوال 7: به مقدار 300ms تأخیر به اینترفیس eth0 از  $h_3$  اضافه نمایید. سپس، از سوی  $h_1$ ، ماشین  $h_3$  را پینگ کنید. مقدار RTT مشاهده شده چقدر است؟

کد پایتون را به صورت زیر تغییر داده و با استفاده از دستور delay در ترمینال  $h_3$  تأخیر اضافه می کنیم.

```
link_r1sw2.intf1.config(bw=3, max_queue_size=1000, enable_ecn=True)
```

```
host: h1
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7# ping 10.10.1.3 -c 10
PING 10.10.1.3 (10.10.1.3) 56(84) bytes of data.
64 bytes from 10.10.1.3: icmp_seq=1 ttl=63 time=604 ms
64 bytes from 10.10.1.3: icmp_seq=2 ttl=63 time=300 ms
64 bytes from 10.10.1.3: icmp_seq=3 ttl=63 time=300 ms
64 bytes from 10.10.1.3: icmp_seq=4 ttl=63 time=300 ms
64 bytes from 10.10.1.3: icmp_seq=5 ttl=63 time=300 ms
64 bytes from 10.10.1.3: icmp_seq=6 ttl=63 time=300 ms
64 bytes from 10.10.1.3: icmp_seq=7 ttl=63 time=300 ms
64 bytes from 10.10.1.3: icmp_seq=8 ttl=63 time=300 ms
64 bytes from 10.10.1.3: icmp_seq=9 ttl=63 time=300 ms
64 bytes from 10.10.1.3: icmp_seq=10 ttl=63 time=301 ms

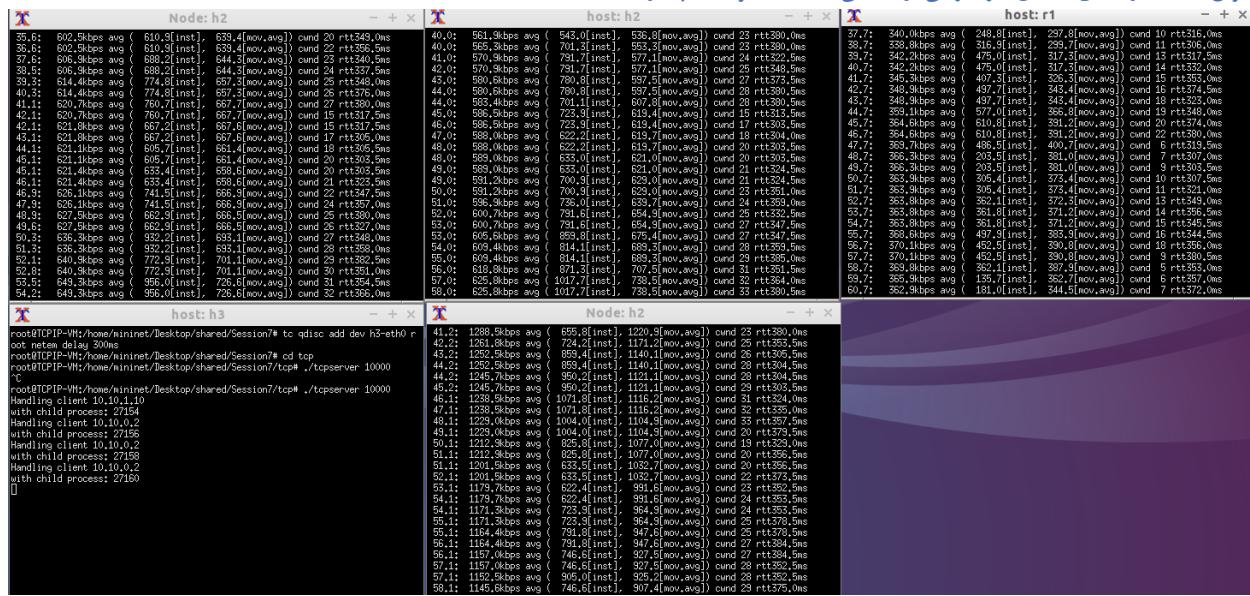
--- 10.10.1.3 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9009ms
rtt min/avg/max/mdev = 300.094/330.878/604.009/91.044 ms
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7# 

host: h3
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7# tc qdisc add dev h3-eth0 root netem delay 300ms
root@TCPPIP-VM:/home/mininet/Desktop/shared/Session7# 
```

مقدار RTT مشاهده شده همان 300 ms می باشد. تاخیر بیشتر بسته اول به دلیل ARP کردن می باشد و پس از پر شدن تاخیر به همان مقدار تعیین شده (یعنی 300 ms) تبدیل می شود.

د - 2) عدالت میان جریان های TCP و تأثیر تأخیر بر آن

سؤال 8: مقدار goodput هر جریان از ماشین های h1 و h2 چقدر است؟



سؤال 9: آیا مقادیر حاصل از آزمایش با مقادیر نظری همخوانی دارند؟

خیر؛ مقادیر نظری یا مقادیر په دست آمده اندکی تفاوت دارند.

سؤال 10: آپا می توانید متوجہ شوید کہ اساساً تأخیر صف ہم داریم یا خیر؟

کاهش ماکسیمم اندازه صفر در  $r1$  باعث کاهش Goodput نسبت به مقادیر تئوری شده است و صفحی که در  $r1$  interface ایجاد می شود، باعث کاهش Goodput باء، client ایجاد شود.



## دانشکده مهندسی کامپیوتر

### آزمایشگاه شبکه های کامپیوتری

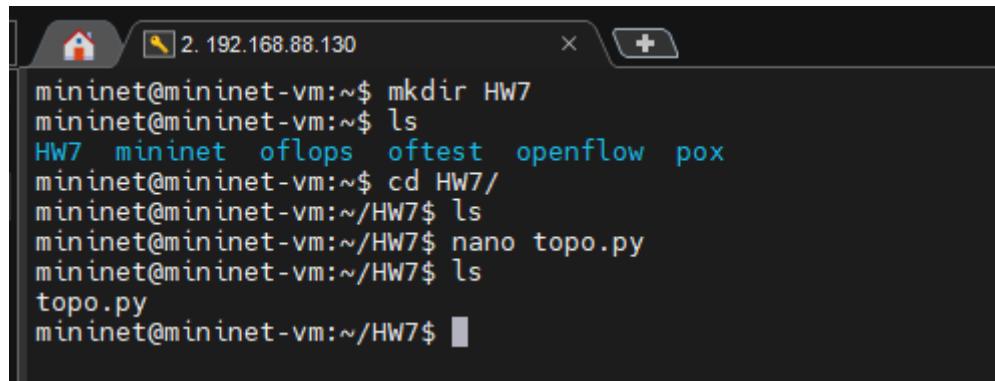
استاد : شکوفه نوروزی

پوریا رحیمی (99521289)

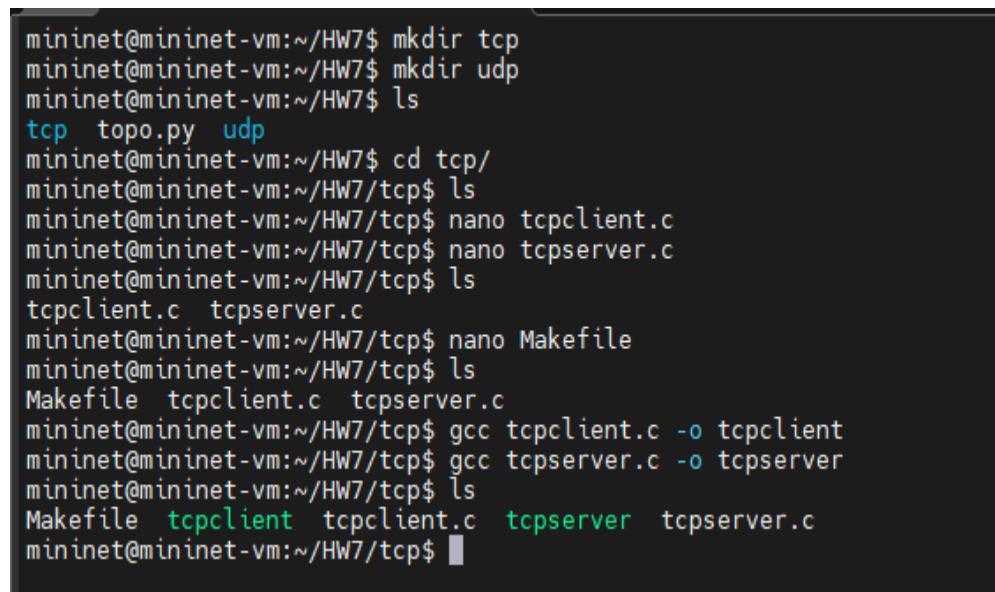
بکتاش انصاری (99521082)

پاییز 1402

در ابتدا محیط برنامه را آماده می کنیم و فایل ها و فولدر های مناسب را ایجاد می کنیم که در تصویر زیر مشاهده می شود :



```
mininet@mininet-vm:~$ mkdir HW7
mininet@mininet-vm:~$ ls
HW7  mininet  oflops  oftest  openflow  pox
mininet@mininet-vm:~$ cd HW7/
mininet@mininet-vm:~/HW7$ ls
mininet@mininet-vm:~/HW7$ nano topo.py
mininet@mininet-vm:~/HW7$ ls
topo.py
mininet@mininet-vm:~/HW7$
```



```
mininet@mininet-vm:~/HW7$ mkdir tcp
mininet@mininet-vm:~/HW7$ mkdir udp
mininet@mininet-vm:~/HW7$ ls
tcp  topo.py  udp
mininet@mininet-vm:~/HW7$ cd tcp/
mininet@mininet-vm:~/HW7/tcp$ ls
mininet@mininet-vm:~/HW7/tcp$ nano tcpclient.c
mininet@mininet-vm:~/HW7/tcp$ nano tcpserver.c
mininet@mininet-vm:~/HW7/tcp$ ls
tcpclient.c  tcpserver.c
mininet@mininet-vm:~/HW7/tcp$ nano Makefile
mininet@mininet-vm:~/HW7/tcp$ ls
Makefile  tcpclient.c  tcpserver.c
mininet@mininet-vm:~/HW7/tcp$ gcc tcpclient.c -o tcpclient
mininet@mininet-vm:~/HW7/tcp$ gcc tcpserver.c -o tcpserver
mininet@mininet-vm:~/HW7/tcp$ ls
Makefile  tcpclient  tcpserver  tcpserver.c
mininet@mininet-vm:~/HW7/tcp$
```

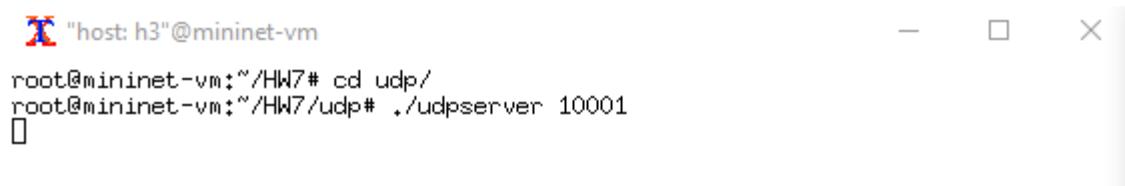


```
mininet@mininet-vm:~/HW7$ cd udp/
mininet@mininet-vm:~/HW7/udp$ ls
mininet@mininet-vm:~/HW7/udp$ nano Makefile
mininet@mininet-vm:~/HW7/udp$ nano udpclient.c
mininet@mininet-vm:~/HW7/udp$ nano udpserver.c
mininet@mininet-vm:~/HW7/udp$ gcc udpclient.c -o udpclient
mininet@mininet-vm:~/HW7/udp$ gcc udpserver.c -o udpserver
mininet@mininet-vm:~/HW7/udp$ ls
Makefile  udpclient  udpserver  udpserver.c
mininet@mininet-vm:~/HW7/udp$
```

حال بعد از راه اندازی برنامه و ورود به محیط mininet یک ترمینال جدید برای host h3 ایجاد می کنیم و سناریو های جدول 1 را به ترتیب اجرا می کنیم و خروجی را نمایش می دهیم :

ترمینالی از h1 که h3 به آن در خواست می فرستد را با عنوان host h3 می شناسیم و ترمینالی که h2 با مقادیر نرخ مختلف و سناریو های مختلف به آن درخواست می فرستد با عنوان Node h3 شناخته می شود.

حال سرور های UDP را روی port های 10001 و 10002 ایجاد می کنیم :



```
X "host: h3"@mininet-vm
root@mininet-vm:~/HW7# cd udp/
root@mininet-vm:~/HW7/udp# ./udpserver 10001
[
```


```
X "Node: h3"@mininet-vm
root@mininet-vm:~/HW7# cd udp/
root@mininet-vm:~/HW7/udp# ./udpserver 10002
[
```

باید توجه داشت که در هر سه سناریو این بخش، host h1 با نرخ 1Mbps داده ارسال می شود و host h2 در سناریو اول با 1Mbps, 2Mbps و 4.5Mbps داده ارسال می شود.

(1) سوال

بخش اول :

X "host: h1"@mininet-vm

— □ ×

```
77.6s - sent: 9697 pkts, 1001.6 kbytes/s
78.6s - sent: 9823 pkts, 1001.1 kbytes/s
79.6s - sent: 9949 pkts, 1000.5 kbytes/s
80.6s - sent: 10075 pkts, 1000.0 kbytes/s
81.6s - sent: 10201 pkts, 999.8 kbytes/s
82.6s - sent: 10327 pkts, 1000.9 kbytes/s
83.6s - sent: 10453 pkts, 1000.8 kbytes/s
84.6s - sent: 10579 pkts, 1001.0 kbytes/s
85.6s - sent: 10705 pkts, 1000.8 kbytes/s
86.6s - sent: 10830 pkts, 999.9 kbytes/s
87.6s - sent: 10956 pkts, 1001.3 kbytes/s
88.6s - sent: 11082 pkts, 1002.6 kbytes/s
89.7s - sent: 11208 pkts, 1000.8 kbytes/s
90.7s - sent: 11334 pkts, 1002.7 kbytes/s
91.7s - sent: 11460 pkts, 999.5 kbytes/s
92.7s - sent: 11586 pkts, 1006.1 kbytes/s
93.7s - sent: 11712 pkts, 1001.0 kbytes/s
94.7s - sent: 11838 pkts, 1000.4 kbytes/s
95.7s - sent: 11963 pkts, 999.6 kbytes/s
96.7s - sent: 12089 pkts, 1000.5 kbytes/s
97.7s - sent: 12215 pkts, 1001.5 kbytes/s
98.7s - sent: 12341 pkts, 1000.7 kbytes/s
^Cpackets sent = 12345, avg rate=1000.0kbps
root@mininet-vm:~/HW7/udp#
```

X "host: h3"@mininet-vm

— □ ×

```
296.0s - received: 9689/ sent: 9689 pkts (loss 0.000%), 1000.7 kbit/s
297.0s - received: 9814/ sent: 9814 pkts (loss 0.000%), 1000.0 kbit/s
298.0s - received: 9939/ sent: 9939 pkts (loss 0.000%), 999.8 kbit/s
299.0s - received: 10064/ sent: 10064 pkts (loss 0.000%), 999.0 kbit/s
300.0s - received: 10190/ sent: 10190 pkts (loss 0.000%), 1001.5 kbit/s
301.0s - received: 10315/ sent: 10315 pkts (loss 0.000%), 999.3 kbit/s
302.0s - received: 10441/ sent: 10441 pkts (loss 0.000%), 1000.9 kbit/s
303.0s - received: 10566/ sent: 10566 pkts (loss 0.000%), 999.9 kbit/s
304.0s - received: 10691/ sent: 10691 pkts (loss 0.000%), 999.3 kbit/s
305.0s - received: 10817/ sent: 10817 pkts (loss 0.000%), 999.7 kbit/s
306.0s - received: 10943/ sent: 10943 pkts (loss 0.000%), 1000.5 kbit/s
307.0s - received: 11068/ sent: 11068 pkts (loss 0.000%), 1000.0 kbit/s
308.0s - received: 11193/ sent: 11193 pkts (loss 0.000%), 999.4 kbit/s
309.0s - received: 11319/ sent: 11319 pkts (loss 0.000%), 1001.6 kbit/s
310.1s - received: 11444/ sent: 11444 pkts (loss 0.000%), 999.2 kbit/s
311.1s - received: 11569/ sent: 11569 pkts (loss 0.000%), 999.4 kbit/s
312.1s - received: 11695/ sent: 11695 pkts (loss 0.000%), 1000.3 kbit/s
313.1s - received: 11821/ sent: 11821 pkts (loss 0.000%), 999.9 kbit/s
314.1s - received: 11947/ sent: 11947 pkts (loss 0.000%), 1001.0 kbit/s
315.1s - received: 12072/ sent: 12072 pkts (loss 0.000%), 999.5 kbit/s
316.1s - received: 12198/ sent: 12198 pkts (loss 0.000%), 1000.7 kbit/s
317.1s - received: 12323/ sent: 12323 pkts (loss 0.000%), 999.6 kbit/s
packet received = 12345 / 12345 sent: 0.000% loss
```

همانطور که در تصویر مشاهده می کنید مقادیر loss برابر با 0 و goodput برای host h3 برابر با 1000 Kbps می باشد.

```

X "host: h2"@mininet-vm
-
X
100.7s - sent: 12589 pkts, 999.2 kbytes/s
101.7s - sent: 12715 pkts, 1003.5 kbytes/s
102.7s - sent: 12840 pkts, 999.3 kbytes/s
103.7s - sent: 12966 pkts, 1001.9 kbytes/s
104.7s - sent: 13092 pkts, 999.6 kbytes/s
105.7s - sent: 13218 pkts, 1001.2 kbytes/s
106.7s - sent: 13344 pkts, 1000.9 kbytes/s
107.8s - sent: 13470 pkts, 1000.6 kbytes/s
108.8s - sent: 13596 pkts, 1000.9 kbytes/s
109.8s - sent: 13722 pkts, 1000.2 kbytes/s
110.8s - sent: 13848 pkts, 1001.3 kbytes/s
111.8s - sent: 13973 pkts, 999.6 kbytes/s
112.8s - sent: 14099 pkts, 1001.0 kbytes/s
113.8s - sent: 14225 pkts, 1005.2 kbytes/s
114.8s - sent: 14351 pkts, 1000.9 kbytes/s
115.8s - sent: 14477 pkts, 1000.2 kbytes/s
116.8s - sent: 14603 pkts, 998.4 kbytes/s
117.8s - sent: 14729 pkts, 1004.9 kbytes/s
118.8s - sent: 14855 pkts, 1000.8 kbytes/s
119.8s - sent: 14981 pkts, 1001.1 kbytes/s
120.8s - sent: 15107 pkts, 1000.5 kbytes/s
121.8s - sent: 15232 pkts, 999.4 kbytes/s
^Cpackets sent = 15330, avg rate=1000.0kbps
root@mininet-vm:~/HW7/udp# █

```

```

X "Node: h3"@mininet-vm
-
X
104.4s - received: 12685/ sent: 12685 pkts (loss 0.000%), 999.5 kbit/s
105.4s - received: 12811/ sent: 12811 pkts (loss 0.000%), 1000.5 kbit/s
106.4s - received: 12937/ sent: 12937 pkts (loss 0.000%), 1000.4 kbit/s
107.4s - received: 13062/ sent: 13062 pkts (loss 0.000%), 999.8 kbit/s
108.4s - received: 13187/ sent: 13187 pkts (loss 0.000%), 997.1 kbit/s
109.4s - received: 13313/ sent: 13313 pkts (loss 0.000%), 1002.9 kbit/s
110.4s - received: 13438/ sent: 13438 pkts (loss 0.000%), 999.5 kbit/s
111.4s - received: 13564/ sent: 13564 pkts (loss 0.000%), 1000.7 kbit/s
112.4s - received: 13689/ sent: 13689 pkts (loss 0.000%), 999.2 kbit/s
113.4s - received: 13815/ sent: 13815 pkts (loss 0.000%), 999.4 kbit/s
114.4s - received: 13941/ sent: 13941 pkts (loss 0.000%), 1000.7 kbit/s
115.4s - received: 14066/ sent: 14066 pkts (loss 0.000%), 1000.0 kbit/s
116.4s - received: 14192/ sent: 14192 pkts (loss 0.000%), 1000.3 kbit/s
117.4s - received: 14318/ sent: 14318 pkts (loss 0.000%), 1000.4 kbit/s
118.4s - received: 14444/ sent: 14444 pkts (loss 0.000%), 1000.2 kbit/s
119.4s - received: 14569/ sent: 14569 pkts (loss 0.000%), 999.7 kbit/s
120.4s - received: 14695/ sent: 14695 pkts (loss 0.000%), 999.5 kbit/s
121.4s - received: 14821/ sent: 14821 pkts (loss 0.000%), 1000.8 kbit/s
122.4s - received: 14947/ sent: 14947 pkts (loss 0.000%), 1000.3 kbit/s
123.5s - received: 15073/ sent: 15073 pkts (loss 0.000%), 1000.6 kbit/s
124.5s - received: 15198/ sent: 15198 pkts (loss 0.000%), 999.8 kbit/s
125.5s - received: 15323/ sent: 15323 pkts (loss 0.000%), 999.2 kbit/s
packet received = 15330 / 15330 sent: 0.000% loss █

```

همانطور که در تصویر مشاهده می کنید مقادیر loss برابر با 0 و goodput برابر با 1000 Kbps Node h3 می باشد.

## بخش دوم :

```
X "host: h1"@mininet-vm
-
X
223.1s - sent: 27890 pkts, 1000.0 kbytes/s
224.1s - sent: 28015 pkts, 999.5 kbytes/s
225.1s - sent: 28140 pkts, 999.5 kbytes/s
226.1s - sent: 28266 pkts, 1000.2 kbytes/s
227.1s - sent: 28392 pkts, 1001.0 kbytes/s
228.1s - sent: 28517 pkts, 999.8 kbytes/s
229.1s - sent: 28643 pkts, 999.9 kbytes/s
230.1s - sent: 28769 pkts, 1000.0 kbytes/s
231.1s - sent: 28894 pkts, 999.6 kbytes/s
232.1s - sent: 29020 pkts, 1002.6 kbytes/s
233.2s - sent: 29146 pkts, 998.7 kbytes/s
234.2s - sent: 29272 pkts, 1002.4 kbytes/s
235.2s - sent: 29397 pkts, 1000.0 kbytes/s
236.2s - sent: 29522 pkts, 999.5 kbytes/s
237.2s - sent: 29648 pkts, 1000.4 kbytes/s
238.2s - sent: 29774 pkts, 1000.2 kbytes/s
239.2s - sent: 29899 pkts, 1000.0 kbytes/s
240.2s - sent: 30025 pkts, 999.3 kbytes/s
241.2s - sent: 30151 pkts, 1000.8 kbytes/s
242.2s - sent: 30277 pkts, 1000.0 kbytes/s
243.2s - sent: 30402 pkts, 999.8 kbytes/s
244.2s - sent: 30528 pkts, 1000.6 kbytes/s
^Cpackets sent = 30543, avg rate= 995.3kbps
root@mininet-vm:~/HW7/udp# █
```

```
X "host: h3"@mininet-vm
-
X
224.8s - received: 26886/ sent: 27936 pkts (loss 3.759%), 959.4 kbit/s
225.8s - received: 27006/ sent: 28056 pkts (loss 3.743%), 959.5 kbit/s
226.8s - received: 27127/ sent: 28177 pkts (loss 3.726%), 960.0 kbit/s
227.8s - received: 27247/ sent: 28297 pkts (loss 3.711%), 958.7 kbit/s
228.8s - received: 27367/ sent: 28417 pkts (loss 3.695%), 959.5 kbit/s
229.8s - received: 27488/ sent: 28538 pkts (loss 3.679%), 960.4 kbit/s
230.8s - received: 27608/ sent: 28658 pkts (loss 3.664%), 959.0 kbit/s
231.8s - received: 27728/ sent: 28778 pkts (loss 3.649%), 959.2 kbit/s
232.8s - received: 27848/ sent: 28898 pkts (loss 3.633%), 959.8 kbit/s
233.8s - received: 27968/ sent: 29018 pkts (loss 3.618%), 959.6 kbit/s
234.8s - received: 28088/ sent: 29138 pkts (loss 3.604%), 959.5 kbit/s
235.8s - received: 28208/ sent: 29258 pkts (loss 3.589%), 958.7 kbit/s
236.8s - received: 28329/ sent: 29379 pkts (loss 3.574%), 960.9 kbit/s
237.8s - received: 28449/ sent: 29499 pkts (loss 3.559%), 959.5 kbit/s
238.8s - received: 28570/ sent: 29620 pkts (loss 3.545%), 959.6 kbit/s
239.8s - received: 28691/ sent: 29741 pkts (loss 3.530%), 960.0 kbit/s
240.8s - received: 28811/ sent: 29861 pkts (loss 3.516%), 958.8 kbit/s
241.8s - received: 28931/ sent: 29981 pkts (loss 3.502%), 959.3 kbit/s
242.8s - received: 29052/ sent: 30102 pkts (loss 3.488%), 960.2 kbit/s
243.8s - received: 29173/ sent: 30223 pkts (loss 3.474%), 959.9 kbit/s
244.8s - received: 29293/ sent: 30343 pkts (loss 3.460%), 959.4 kbit/s
245.8s - received: 29413/ sent: 30463 pkts (loss 3.447%), 959.7 kbit/s
packet received = 29493 / 30543 sent: 3.438% loss
█
```

برای goodput host h3 مقدار loss تقریبا برابر با 3.4 می باشد و مقدار host h1 , host h3 تقریبا برابر با 3.4 می باشد.

برای host h1 تقریبا برابر با 995 Kbps می باشد.

X "host: h2"@mininet-vm

```
163.6s - sent: 40891 pkts, 2005.1 kbytes/s
164.6s - sent: 41142 pkts, 2001.4 kbytes/s
165.6s - sent: 41393 pkts, 2000.7 kbytes/s
166.6s - sent: 41643 pkts, 1999.2 kbytes/s
167.6s - sent: 41894 pkts, 2003.6 kbytes/s
168.6s - sent: 42144 pkts, 1999.9 kbytes/s
169.6s - sent: 42395 pkts, 2000.3 kbytes/s
170.6s - sent: 42646 pkts, 2002.7 kbytes/s
171.6s - sent: 42897 pkts, 1999.1 kbytes/s
172.6s - sent: 43148 pkts, 2002.9 kbytes/s
173.6s - sent: 43399 pkts, 1999.7 kbytes/s
174.6s - sent: 43650 pkts, 2003.2 kbytes/s
175.6s - sent: 43901 pkts, 2000.6 kbytes/s
176.6s - sent: 44152 pkts, 2001.7 kbytes/s
177.6s - sent: 44402 pkts, 1999.6 kbytes/s
178.6s - sent: 44653 pkts, 2003.3 kbytes/s
179.6s - sent: 44904 pkts, 2000.8 kbytes/s
180.6s - sent: 45155 pkts, 2001.5 kbytes/s
181.6s - sent: 45406 pkts, 2000.8 kbytes/s
182.6s - sent: 45657 pkts, 2001.3 kbytes/s
183.6s - sent: 45908 pkts, 2002.5 kbytes/s
184.6s - sent: 46159 pkts, 2001.5 kbytes/s
^Cpackets sent = 46179, avg rate=1987.3kbps
root@mininet-vm:~/HW7/udp#
```

X "Node: h3"@mininet-vm

```
172.4s - received: 40733/ sent: 41004 pkts (loss 0.661%), 1919.9 kbit/s
173.4s - received: 40971/ sent: 41242 pkts (loss 0.657%), 1900.4 kbit/s
174.4s - received: 41211/ sent: 41482 pkts (loss 0.653%), 1919.3 kbit/s
175.4s - received: 41451/ sent: 41722 pkts (loss 0.650%), 1919.2 kbit/s
176.4s - received: 41691/ sent: 41962 pkts (loss 0.646%), 1914.9 kbit/s
177.4s - received: 41931/ sent: 42202 pkts (loss 0.642%), 1916.3 kbit/s
178.4s - received: 42172/ sent: 42443 pkts (loss 0.639%), 1924.8 kbit/s
179.4s - received: 42412/ sent: 42683 pkts (loss 0.635%), 1918.7 kbit/s
180.4s - received: 42652/ sent: 42923 pkts (loss 0.631%), 1916.4 kbit/s
181.4s - received: 42892/ sent: 43163 pkts (loss 0.628%), 1917.8 kbit/s
182.4s - received: 43132/ sent: 43403 pkts (loss 0.624%), 1919.0 kbit/s
183.4s - received: 43373/ sent: 43644 pkts (loss 0.621%), 1921.0 kbit/s
184.4s - received: 43613/ sent: 43884 pkts (loss 0.618%), 1917.9 kbit/s
185.4s - received: 43854/ sent: 44125 pkts (loss 0.614%), 1922.0 kbit/s
186.4s - received: 44094/ sent: 44365 pkts (loss 0.611%), 1919.9 kbit/s
187.4s - received: 44335/ sent: 44606 pkts (loss 0.608%), 1917.2 kbit/s
188.5s - received: 44575/ sent: 44846 pkts (loss 0.604%), 1919.4 kbit/s
189.5s - received: 44815/ sent: 45086 pkts (loss 0.601%), 1918.6 kbit/s
190.5s - received: 45056/ sent: 45327 pkts (loss 0.598%), 1924.0 kbit/s
191.5s - received: 45296/ sent: 45567 pkts (loss 0.595%), 1918.5 kbit/s
192.5s - received: 45536/ sent: 45807 pkts (loss 0.592%), 1919.2 kbit/s
193.5s - received: 45777/ sent: 46048 pkts (loss 0.589%), 1915.8 kbit/s
packet received = 45908 / 46179 sent: 0.587% loss
```

برای host h2, Node h3 goodput مقدار loss تقریباً برابر با 0.6 می باشد و مقدار loss تقریباً برابر با 1987 Kbps می باشد.

### بخش سوم :

```
X "host: h1"@mininet-vm
-
X
X
188.0s - sent: 23496 pkts, 1000.0 kbytes/s
189.0s - sent: 23622 pkts, 1000.0 kbytes/s
190.0s - sent: 23747 pkts, 999.9 kbytes/s
191.0s - sent: 23873 pkts, 1000.4 kbytes/s
192.0s - sent: 23999 pkts, 1000.3 kbytes/s
193.0s - sent: 24124 pkts, 1000.0 kbytes/s
194.0s - sent: 24250 pkts, 1000.3 kbytes/s
195.0s - sent: 24376 pkts, 1000.2 kbytes/s
196.0s - sent: 24501 pkts, 999.8 kbytes/s
197.0s - sent: 24627 pkts, 1000.5 kbytes/s
198.0s - sent: 24752 pkts, 999.8 kbytes/s
199.0s - sent: 24878 pkts, 999.8 kbytes/s
200.0s - sent: 25003 pkts, 999.7 kbytes/s
201.0s - sent: 25129 pkts, 1000.5 kbytes/s
202.0s - sent: 25255 pkts, 999.5 kbytes/s
203.0s - sent: 25381 pkts, 1000.1 kbytes/s
204.0s - sent: 25507 pkts, 1000.3 kbytes/s
205.0s - sent: 25632 pkts, 999.7 kbytes/s
206.1s - sent: 25758 pkts, 1000.1 kbytes/s
207.1s - sent: 25884 pkts, 1000.6 kbytes/s
208.1s - sent: 26009 pkts, 999.7 kbytes/s
209.1s - sent: 26134 pkts, 999.7 kbytes/s
^Cpackets sent = 26187, avg rate= 994.4kbps
root@mininet-vm:~/HW7/udp#
```

```
X "host: h3"@mininet-vm
-
X
X
194.3s - received: 12918/ sent: 23268 pkts (loss 44.482%), 463.4 kbit/s
195.3s - received: 12984/ sent: 23394 pkts (loss 44.499%), 523.6 kbit/s
196.3s - received: 13051/ sent: 23521 pkts (loss 44.513%), 528.2 kbit/s
197.3s - received: 13119/ sent: 23648 pkts (loss 44.524%), 535.4 kbit/s
198.3s - received: 13178/ sent: 23775 pkts (loss 44.572%), 465.4 kbit/s
199.3s - received: 13240/ sent: 23901 pkts (loss 44.605%), 491.7 kbit/s
200.3s - received: 13308/ sent: 24027 pkts (loss 44.612%), 539.6 kbit/s
201.3s - received: 13365/ sent: 24152 pkts (loss 44.663%), 455.4 kbit/s
202.3s - received: 13427/ sent: 24279 pkts (loss 44.697%), 489.3 kbit/s
203.3s - received: 13485/ sent: 24405 pkts (loss 44.745%), 460.3 kbit/s
204.3s - received: 13547/ sent: 24531 pkts (loss 44.776%), 490.9 kbit/s
205.4s - received: 13608/ sent: 24656 pkts (loss 44.809%), 487.7 kbit/s
206.4s - received: 13673/ sent: 24785 pkts (loss 44.834%), 504.9 kbit/s
207.4s - received: 13744/ sent: 24910 pkts (loss 44.825%), 567.3 kbit/s
208.4s - received: 13806/ sent: 25037 pkts (loss 44.858%), 487.6 kbit/s
209.4s - received: 13873/ sent: 25163 pkts (loss 44.867%), 532.4 kbit/s
210.4s - received: 13936/ sent: 25290 pkts (loss 44.895%), 497.1 kbit/s
211.4s - received: 13997/ sent: 25415 pkts (loss 44.926%), 480.4 kbit/s
212.4s - received: 14058/ sent: 25540 pkts (loss 44.957%), 487.7 kbit/s
213.4s - received: 14119/ sent: 25665 pkts (loss 44.987%), 487.7 kbit/s
214.4s - received: 14162/ sent: 25792 pkts (loss 45.092%), 343.9 kbit/s
215.4s - received: 14221/ sent: 25918 pkts (loss 45.131%), 468.9 kbit/s
packet received = 14259 / 26187 sent: 45.549% loss

```

برای h1 مقدار loss تقریباً برابر با 46 درصد بوده و مقدار goodput نیز تقریباً برابر با 994 Kbps می باشد.

X "host: h2"@mininet-vm

```

240.4s - sent:135216 pkts, 4499.4 kbytes/s
241.4s - sent:135779 pkts, 4503.4 kbytes/s
242.4s - sent:136343 pkts, 4503.8 kbytes/s
243.4s - sent:136906 pkts, 4502.7 kbytes/s
244.4s - sent:137469 pkts, 4498.7 kbytes/s
245.4s - sent:138033 pkts, 4505.2 kbytes/s
246.4s - sent:138596 pkts, 4499.0 kbytes/s
247.4s - sent:139160 pkts, 4504.0 kbytes/s
248.4s - sent:139724 pkts, 4504.9 kbytes/s
249.4s - sent:140287 pkts, 4501.8 kbytes/s
250.4s - sent:140850 pkts, 4498.2 kbytes/s
251.4s - sent:141414 pkts, 4504.2 kbytes/s
252.4s - sent:141977 pkts, 4503.0 kbytes/s
253.4s - sent:142541 pkts, 4505.2 kbytes/s
254.4s - sent:143104 pkts, 4502.1 kbytes/s
255.4s - sent:143667 pkts, 4500.0 kbytes/s
256.4s - sent:144230 pkts, 4504.0 kbytes/s
257.4s - sent:144793 pkts, 4502.5 kbytes/s
258.4s - sent:145356 pkts, 4501.2 kbytes/s
259.4s - sent:145919 pkts, 4499.6 kbytes/s
260.4s - sent:146482 pkts, 4503.6 kbytes/s
261.4s - sent:147045 pkts, 4500.4 kbytes/s
^Cpackets sent = 147401, avg rate=4452.5kbps
root@mininet-vm:~/HW7/udp# █

```

X "Node: h3"@mininet-vm

```

251.3s - received: 75152/ sent:135373 pkts (loss 44.485%), 2874.7 kbit/s
252.3s - received: 75512/ sent:135895 pkts (loss 44.434%), 2877.2 kbit/s
253.3s - received: 75872/ sent:136457 pkts (loss 44.399%), 2879.4 kbit/s
254.3s - received: 76232/ sent:137021 pkts (loss 44.365%), 2879.5 kbit/s
255.3s - received: 76592/ sent:137584 pkts (loss 44.331%), 2877.9 kbit/s
256.3s - received: 76953/ sent:138148 pkts (loss 44.297%), 2879.3 kbit/s
257.3s - received: 77313/ sent:138711 pkts (loss 44.263%), 2879.7 kbit/s
258.3s - received: 77673/ sent:139273 pkts (loss 44.230%), 2879.4 kbit/s
259.3s - received: 78034/ sent:139838 pkts (loss 44.197%), 2879.7 kbit/s
260.3s - received: 78394/ sent:140401 pkts (loss 44.164%), 2878.7 kbit/s
261.3s - received: 78754/ sent:140963 pkts (loss 44.131%), 2876.1 kbit/s
262.3s - received: 79114/ sent:141526 pkts (loss 44.099%), 2878.8 kbit/s
263.3s - received: 79474/ sent:142088 pkts (loss 44.067%), 2877.3 kbit/s
264.3s - received: 79834/ sent:142652 pkts (loss 44.036%), 2877.2 kbit/s
265.3s - received: 80195/ sent:143217 pkts (loss 44.005%), 2881.7 kbit/s
266.3s - received: 80555/ sent:143779 pkts (loss 43.973%), 2876.8 kbit/s
267.3s - received: 80916/ sent:144343 pkts (loss 43.942%), 2880.9 kbit/s
268.3s - received: 81276/ sent:144906 pkts (loss 43.911%), 2878.3 kbit/s
269.3s - received: 81637/ sent:145470 pkts (loss 43.881%), 2880.1 kbit/s
270.3s - received: 81997/ sent:146033 pkts (loss 43.850%), 2879.0 kbit/s
271.3s - received: 82357/ sent:146596 pkts (loss 43.820%), 2879.4 kbit/s
272.3s - received: 82717/ sent:147158 pkts (loss 43.790%), 2877.5 kbit/s
packet received = 82872 / 147400 sent: 43.777% loss █

```

برای h2 نیز مقدار loss تقریباً برابر با 44 درصد می باشد و مقدار goodput نیز تقریباً برابر با 4453 Kbps می باشد.

## سوال (2)

تفاوت های کوچکی وجود دارند که بزرگترین آن ها مقدار loss در بخش سوم می باشد که احتمالا به دلیل این است که برای آن ها باند های جدایگانه در نظر گرفتیم. ولی در کل به مقادیر تحلیلی خیلی نزدیک است برای بخش اول که دقیقا مقدار عملی با مقدار تحلیلی یکی بود برای بخش دوم نیز کمی تفاوت وجود داشت و برای بخش سوم نیز کمی تفاوت وجود دارد.

## سوال (3)

برای این بخش در ابتدا یک سرور TCP بر روی port 10003 بر روی host h3 و همچنین یک TCP client برای h2 ایجاد می کنیم که به این سرور داده ارسال می کند و سپس نتایج سناریو جدول 3 آزمایش را تست می کنیم و نتایج را با مقادیر تحلیلی موجود در جدول 4 مقایسه می کنیم که به صورت زیر می باشد :

بخش اول :

```
X "host: h1"@mininet-vm - □ ×
120.8s - sent: 15102 pkts, 999.9 kbits/s
121.8s - sent: 15227 pkts, 1000.0 kbits/s
122.8s - sent: 15352 pkts, 999.8 kbits/s
123.8s - sent: 15478 pkts, 1000.7 kbits/s
124.8s - sent: 15603 pkts, 999.8 kbits/s
125.8s - sent: 15729 pkts, 1000.2 kbits/s
126.8s - sent: 15854 pkts, 999.6 kbits/s
127.8s - sent: 15979 pkts, 997.2 kbits/s
128.8s - sent: 16105 pkts, 1003.3 kbits/s
129.8s - sent: 16231 pkts, 1000.1 kbits/s
130.8s - sent: 16357 pkts, 1000.8 kbits/s
131.9s - sent: 16483 pkts, 1001.0 kbits/s
132.9s - sent: 16608 pkts, 999.4 kbits/s
133.9s - sent: 16734 pkts, 1002.0 kbits/s
134.9s - sent: 16860 pkts, 1000.7 kbits/s
135.9s - sent: 16985 pkts, 1000.0 kbits/s
136.9s - sent: 17110 pkts, 999.7 kbits/s
137.9s - sent: 17235 pkts, 999.9 kbits/s
138.9s - sent: 17361 pkts, 999.9 kbits/s
139.9s - sent: 17487 pkts, 1001.2 kbits/s
140.9s - sent: 17613 pkts, 1000.3 kbits/s
141.9s - sent: 17739 pkts, 999.6 kbits/s
^Cpackets sent = 17756, avg rate= 977.5kbps
root@mininet-vm:~/HW7/udp# █
```

☒ "host: h2"@mininet-vm

— □ ×

```
192.3s - sent: 24043 pkts, 1000.9 kbytes/s
193.3s - sent: 24169 pkts, 1000.7 kbytes/s
194.3s - sent: 24294 pkts, 999.7 kbytes/s
195.3s - sent: 24420 pkts, 1001.2 kbytes/s
196.4s - sent: 24546 pkts, 1000.5 kbytes/s
197.4s - sent: 24672 pkts, 1000.2 kbytes/s
198.4s - sent: 24798 pkts, 1000.6 kbytes/s
199.4s - sent: 24924 pkts, 1000.9 kbytes/s
200.4s - sent: 25050 pkts, 1000.5 kbytes/s
201.4s - sent: 25176 pkts, 998.9 kbytes/s
202.4s - sent: 25301 pkts, 997.4 kbytes/s
203.4s - sent: 25428 pkts, 1013.5 kbytes/s
204.4s - sent: 25554 pkts, 1001.3 kbytes/s
205.4s - sent: 25679 pkts, 1000.0 kbytes/s
206.4s - sent: 25805 pkts, 1004.2 kbytes/s
207.4s - sent: 25931 pkts, 1000.2 kbytes/s
208.4s - sent: 26057 pkts, 1000.5 kbytes/s
209.5s - sent: 26183 pkts, 1001.4 kbytes/s
210.5s - sent: 26309 pkts, 1000.0 kbytes/s
211.5s - sent: 26435 pkts, 1001.5 kbytes/s
212.5s - sent: 26561 pkts, 1002.3 kbytes/s
213.5s - sent: 26687 pkts, 1000.8 kbytes/s
^Cpackets sent = 26793, avg rate= 978.8kbps
root@mininet-vm:~/HW7/udp#
```

بخش دوم :

☒ "host: h1"@mininet-vm

— □ ×

```
337.9s - sent: 42238 pkts, 1000.0 kbytes/s
338.9s - sent: 42364 pkts, 1001.1 kbytes/s
339.9s - sent: 42490 pkts, 1000.5 kbytes/s
340.9s - sent: 42616 pkts, 1000.7 kbytes/s
341.9s - sent: 42742 pkts, 1000.8 kbytes/s
342.9s - sent: 42868 pkts, 1000.9 kbytes/s
343.9s - sent: 42994 pkts, 1000.6 kbytes/s
344.9s - sent: 43120 pkts, 1001.4 kbytes/s
346.0s - sent: 43246 pkts, 1000.6 kbytes/s
347.0s - sent: 43372 pkts, 1000.5 kbytes/s
348.0s - sent: 43498 pkts, 1003.3 kbytes/s
349.0s - sent: 43624 pkts, 1000.1 kbytes/s
350.0s - sent: 43750 pkts, 1001.2 kbytes/s
351.0s - sent: 43875 pkts, 999.3 kbytes/s
352.0s - sent: 44000 pkts, 998.5 kbytes/s
353.0s - sent: 44126 pkts, 1002.7 kbytes/s
354.0s - sent: 44252 pkts, 1001.0 kbytes/s
355.0s - sent: 44378 pkts, 1000.4 kbytes/s
356.0s - sent: 44504 pkts, 1001.0 kbytes/s
357.0s - sent: 44629 pkts, 999.3 kbytes/s
358.0s - sent: 44755 pkts, 1000.5 kbytes/s
359.0s - sent: 44881 pkts, 1000.0 kbytes/s
^Cpackets sent = 44977, avg rate= 992.3kbps
root@mininet-vm:~/HW7/udp#
```

X "host: h2"@mininet-vm

— □ ×

```
412.6s - sent:103148 pkts, 2000.9 kbits/s
413.6s - sent:103399 pkts, 2002.9 kbits/s
414.6s - sent:103650 pkts, 2002.2 kbits/s
415.6s - sent:103901 pkts, 2000.5 kbits/s
416.6s - sent:104152 pkts, 2001.9 kbits/s
417.6s - sent:104403 pkts, 2000.4 kbits/s
418.6s - sent:104654 pkts, 2002.4 kbits/s
419.6s - sent:104905 pkts, 2001.1 kbits/s
420.6s - sent:105156 pkts, 2003.6 kbits/s
421.6s - sent:105406 pkts, 1999.5 kbits/s
422.6s - sent:105657 pkts, 2000.3 kbits/s
423.6s - sent:105908 pkts, 2000.9 kbits/s
424.6s - sent:106159 pkts, 1999.3 kbits/s
425.6s - sent:106410 pkts, 2002.2 kbits/s
426.6s - sent:106661 pkts, 2001.9 kbits/s
427.6s - sent:106912 pkts, 2002.3 kbits/s
428.6s - sent:107163 pkts, 2002.2 kbits/s
429.6s - sent:107414 pkts, 2007.3 kbits/s
430.6s - sent:107664 pkts, 1999.3 kbits/s
431.7s - sent:107915 pkts, 2007.3 kbits/s
432.7s - sent:108166 pkts, 1998.8 kbits/s
433.7s - sent:108415 pkts, 1991.7 kbits/s
^Cpackets sent = 108416, avg rate=1984.8kbps
root@mininet-vm:~/HW7/udp#
```

بخش سوم :

X "host: h1"@mininet-vm

— □ ×

```
419.1s - sent: 52384 pkts, 1000.0 kbits/s
420.1s - sent: 52509 pkts, 1000.0 kbits/s
421.1s - sent: 52634 pkts, 999.6 kbits/s
422.1s - sent: 52760 pkts, 1000.8 kbits/s
423.1s - sent: 52885 pkts, 999.7 kbits/s
424.1s - sent: 53010 pkts, 999.6 kbits/s
425.1s - sent: 53136 pkts, 999.5 kbits/s
426.1s - sent: 53262 pkts, 1001.1 kbits/s
427.1s - sent: 53387 pkts, 999.8 kbits/s
428.1s - sent: 53513 pkts, 999.8 kbits/s
429.1s - sent: 53638 pkts, 999.5 kbits/s
430.1s - sent: 53764 pkts, 1000.2 kbits/s
431.1s - sent: 53890 pkts, 1000.8 kbits/s
432.1s - sent: 54016 pkts, 1000.2 kbits/s
433.1s - sent: 54141 pkts, 999.7 kbits/s
434.1s - sent: 54267 pkts, 1000.5 kbits/s
435.1s - sent: 54392 pkts, 999.4 kbits/s
436.1s - sent: 54518 pkts, 996.5 kbits/s
437.1s - sent: 54644 pkts, 1004.2 kbits/s
438.1s - sent: 54769 pkts, 999.9 kbits/s
439.1s - sent: 54895 pkts, 1000.0 kbits/s
440.1s - sent: 55020 pkts, 1000.0 kbits/s
^Cpackets sent = 55120, avg rate= 993.7kbps
root@mininet-vm:~/HW7/udp#
```

X "host: h2"@mininet-vm

— □ ×

```
447.9s - sent:251916 pkts, 4503.3 kbytes/s
448.9s - sent:252480 pkts, 4509.2 kbytes/s
449.9s - sent:253043 pkts, 4501.7 kbytes/s
450.9s - sent:253607 pkts, 4509.2 kbytes/s
451.9s - sent:254170 pkts, 4502.5 kbytes/s
452.9s - sent:254733 pkts, 4502.6 kbytes/s
453.9s - sent:255296 pkts, 4501.1 kbytes/s
454.9s - sent:255859 pkts, 4501.0 kbytes/s
455.9s - sent:256423 pkts, 4504.3 kbytes/s
456.9s - sent:256987 pkts, 4509.9 kbytes/s
457.9s - sent:257550 pkts, 4500.4 kbytes/s
458.9s - sent:258113 pkts, 4501.5 kbytes/s
459.9s - sent:258677 pkts, 4505.1 kbytes/s
460.9s - sent:259240 pkts, 4501.4 kbytes/s
461.9s - sent:259803 pkts, 4501.9 kbytes/s
462.9s - sent:260367 pkts, 4504.9 kbytes/s
463.9s - sent:260930 pkts, 4499.1 kbytes/s
464.9s - sent:261493 pkts, 4501.2 kbytes/s
465.9s - sent:262057 pkts, 4510.6 kbytes/s
466.9s - sent:262620 pkts, 4499.7 kbytes/s
467.9s - sent:263186 pkts, 4526.3 kbytes/s
468.9s - sent:263749 pkts, 4501.9 kbytes/s
^Cpackets sent = 264288, avg rate=4473.3kbps
root@mininet-vm:~/HW7/udp# █
```

همانطور که مشاهده می شود مقادیر خیلی نزدیک به مقادیر تحلیلی می باشند و رفته رفته هر چی زمان می گذرد این مقادیر به مقادیر تحلیلی نزدیک تر نیز می شوند.

بخش اول :

UDP h1	UDP h2	TCP h2
978	979	996

بخش دوم :

UDP h1	UDP h2	TCP h2
992	1984	67

بخش سوم :

UDP h1	UDP h2	TCP h2
994	4473	کمتر از 0.1

در نتیجه طبق جدول نیز مشاهده می کنیم که تفاوت زیادی با مقادیر تحلیلی ندارد و پس از گذر زمان این تفاوت کمتر نیز می شود.

(ج) بخش (ج)

پس از خروج از mininet و پاک کردن topo قبلی ، حالا topo جدید را جوری عوض می کنیم که طول صفحه روتر 1 به مقدار 1000 کاهش یابد، همچنین پنهانی باند اینترفیس آن را برابر 5 Mbps قرار داده و ویژگی enab\_ecn را نیز برای آن مقدار False را قرار می دهیم که در تصویر زیر مشاهده می شود :

```
res.intf1.config(bw=5 , max_queue_size=1000 , enable_ecn=False)
```

حالا یک سرور TCP را روی ماشین h3 اجرا می کنیم و یک client TCP را نیز بر روی ماشین h1 ایجاد می کنیم.

(4) سوال

```
X "host: h3"@mininet-vm
root@mininet-vm:~/HW7# cd tcp/
root@mininet-vm:~/HW7/tcp# ./tcpserver 10000
Handling client 10.10.0.1
with child process: 72617
■
```

```

X "host: h1"@mininet-vm
285.4: 4664.2kbps avg ( 4660.7[inst], 4833.2[mov,avg]) cwnd 2459 rtt4669.1ms
286.5: 4654.4kbps avg ( 1945.7[inst], 4544.4[mov,avg]) cwnd 2410 rtt4745.4ms
287.5: 4637.9kbps avg ( -0.0[inst], 4090.0[mov,avg]) cwnd 2188 rtt4756.8ms
288.5: 4621.4kbps avg ( -0.0[inst], 3681.0[mov,avg]) cwnd 1948 rtt4770.5ms
289.5: 4605.0kbps avg ( -0.0[inst], 3312.9[mov,avg]) cwnd 1726 rtt4779.0ms
290.6: 4588.8kbps avg ( -0.0[inst], 2981.6[mov,avg]) cwnd 1586 rtt4790.3ms
291.6: 4600.1kbps avg ( 7790.0[inst], 3462.4[mov,avg]) cwnd 1772 rtt4320.9ms
292.6: 4615.9kbps avg ( 9110.9[inst], 4027.3[mov,avg]) cwnd 1772 rtt3765.1ms
293.6: 4624.6kbps avg ( 7114.2[inst], 4336.0[mov,avg]) cwnd 1772 rtt3607.9ms
294.7: 4627.2kbps avg ( 5388.6[inst], 4441.2[mov,avg]) cwnd 1505 rtt3595.8ms
295.7: 4622.9kbps avg ( 3392.5[inst], 4336.4[mov,avg]) cwnd 1345 rtt3594.2ms
296.7: 4611.7kbps avg ( 1369.5[inst], 4039.7[mov,avg]) cwnd 1270 rtt3581.7ms
297.7: 4612.7kbps avg ( 4886.4[inst], 4124.4[mov,avg]) cwnd 1005 rtt3519.3ms
298.7: 4626.9kbps avg ( 8772.9[inst], 4589.2[mov,avg]) cwnd 1026 rtt3106.2ms
299.8: 4642.4kbps avg ( 9168.0[inst], 5047.1[mov,avg]) cwnd 1047 rtt2128.1ms
300.8: 4663.9kbps avg (10956.1[inst], 5638.0[mov,avg]) cwnd 1068 rtt2481.8ms
301.8: 4664.0kbps avg ( 4671.8[inst], 5541.4[mov,avg]) cwnd 1089 rtt2512.8ms
302.8: 4664.0kbps avg ( 4664.7[inst], 5453.7[mov,avg]) cwnd 1111 rtt2562.5ms
303.9: 4664.0kbps avg ( 4671.7[inst], 5375.5[mov,avg]) cwnd 1132 rtt2608.0ms
304.9: 4663.9kbps avg ( 4651.7[inst], 5303.1[mov,avg]) cwnd 1156 rtt2660.6ms
305.9: 4664.0kbps avg ( 4672.5[inst], 5240.0[mov,avg]) cwnd 1200 rtt2706.9ms
306.9: 4664.0kbps avg ( 4663.0[inst], 5182.3[mov,avg]) cwnd 1266 rtt2756.9ms
^C
root@mininet-vm:~/HW7/tcp# ■

```

مقدار نرخ ماشین h1 بعد از پایدار شدن و با گذشت کمی زمان تقریبا برابر با 4664 Kbps می باشد و همچنین مقدار rtt نیز تقریبا برابر با 2757 ms می باشد و همچنین مقدار پنجره ازدحام نیز مقداری از 10 شروع شده و به صورت نمایی زیاد می شود و در کل عددی بین 990 تا 1270 می باشد.

```
res.intf1.config(bw=5 , max_queue_size=1000 , enable_ecn=True)
```

سپس مقدار enable\_ecn را برابر با مقدار True می گذاریم و topo را پاکسازی کرده و دوباره آن را اجرا می کنیم.

## سؤال (5)

X "host: h3"@mininet-vm

root@mininet-vm:/HW7/tcp# ./tcpserver 10000  
Handling client 10.10.0.1  
with child process: 76964

---

X "host: h1"@mininet-vm

135.4: 4661.8kbps avg ( 4678.5[inst], 4667.0[mov,avg] ) cwnd 21 rtt 47.8ms  
135.6: 4661.8kbps avg ( 4678.5[inst], 4667.0[mov,avg] ) cwnd 23 rtt 52.4ms  
135.8: 4661.8kbps avg ( 4678.5[inst], 4667.0[mov,avg] ) cwnd 18 rtt 39.0ms  
136.1: 4661.8kbps avg ( 4669.9[inst], 4667.3[mov,avg] ) cwnd 20 rtt 43.8ms  
136.3: 4661.8kbps avg ( 4669.9[inst], 4667.3[mov,avg] ) cwnd 22 rtt 51.2ms  
136.5: 4661.8kbps avg ( 4669.9[inst], 4667.3[mov,avg] ) cwnd 11 rtt 43.5ms  
136.7: 4661.8kbps avg ( 4669.9[inst], 4667.3[mov,avg] ) cwnd 14 rtt 31.0ms  
136.9: 4661.8kbps avg ( 4669.9[inst], 4667.3[mov,avg] ) cwnd 17 rtt 38.1ms  
137.2: 4661.7kbps avg ( 4640.6[inst], 4664.6[mov,avg] ) cwnd 20 rtt 44.1ms  
137.4: 4661.7kbps avg ( 4640.6[inst], 4664.6[mov,avg] ) cwnd 15 rtt 49.0ms  
137.6: 4661.7kbps avg ( 4640.6[inst], 4664.6[mov,avg] ) cwnd 17 rtt 45.6ms  
137.8: 4661.7kbps avg ( 4640.6[inst], 4664.6[mov,avg] ) cwnd 19 rtt 43.0ms  
138.1: 4661.7kbps avg ( 4640.6[inst], 4664.6[mov,avg] ) cwnd 22 rtt 48.7ms  
138.3: 4661.3kbps avg ( 4618.0[inst], 4660.0[mov,avg] ) cwnd 16 rtt 48.6ms  
138.5: 4661.3kbps avg ( 4618.0[inst], 4660.0[mov,avg] ) cwnd 18 rtt 42.8ms  
138.7: 4661.3kbps avg ( 4618.0[inst], 4660.0[mov,avg] ) cwnd 21 rtt 47.8ms  
138.9: 4661.3kbps avg ( 4618.0[inst], 4660.0[mov,avg] ) cwnd 16 rtt 51.0ms  
139.1: 4661.3kbps avg ( 4618.0[inst], 4660.0[mov,avg] ) cwnd 18 rtt 40.8ms  
139.4: 4661.4kbps avg ( 4667.7[inst], 4660.7[mov,avg] ) cwnd 20 rtt 46.6ms  
139.6: 4661.4kbps avg ( 4667.7[inst], 4660.7[mov,avg] ) cwnd 23 rtt 52.7ms  
139.8: 4661.4kbps avg ( 4667.7[inst], 4660.7[mov,avg] ) cwnd 17 rtt 39.2ms  
139.9: 4661.4kbps avg ( 4667.7[inst], 4660.7[mov,avg] ) cwnd 18 rtt 40.4ms

مطابق تصویر بالا مقدار نرخ برای ماشین h1 برابر با مقدار تقریبا 4660 Kbps بوده و همچنین مقدار  $rtt$  نیز تقریبا برابر با 40 ms می باشد و در آخر نیز مقدار پنجره ازدحام مقداری بین 13 تا 22 می باشد که به صورت متغیر است.

## سوال (6)

با مقایسه مقادیر بدست آمده در سوال 4 و 5 به راحتی می توان دریافت کرد که اگر مقدار  $enable\_ecn=True$  باشد، برای اینترفیس eth1 از روت 1، فرستنده قبل از به وجود آمدن ازدحام در شبکه و بعد از آن، از drop شدن بسته ها، متوجه می شود و پنجره ارسال خود را مطابق با آن تنظیم می کند برای همین هم مقدار  $rtt$  و هم مقدار  $cwnd$  به صورت چشم گیری کاهش پیدا می کند ولی در صورتی که مقدار  $enable\_ecn=False$  باشد مقدار  $rtt$  و مقدار  $cwnd$  به اندازه قابل ملاحظه ای بالا می رود.

# Quagga

Software Routing Suite

# Internet core routers

- Powerful
- Handling large amounts of traffic
- Built by Huawei, Juniper or Cisco
- They use proprietary operating systems
  - such as Cisco IOS, or JunOS



# Quagga

- Free routing software suite
- Similar commands to the ones in Cisco's IOS
- MiniNExT
  - Mininet Extended
  - An extension layer to Mininet
  - Integrates Quagga into Mininet's virtual environment

# Quagga

- Implementations of several routing protocols (namely OSPF, RIP and BGP-4)
- Important Quagga processes (daemons):
  - zebra
    - Manage the network interfaces
  - ripd
    - Handles RIP version 2 implementation
  - ripngd
    - Handles RIP routing for IPv6
  - quagga
    - The main service, which is used to call the three daemons above

# Starting a Quagga process

- 4 files:
  1. daemons
  2. debian.conf
  3. zebra.conf
  4. ripd.conf
- 1. daemons:
  - zebra=yes
  - bgpd=no
  - ospfd=no
  - ospf6d=no
  - ripd=no
  - ripngd=no
  - isisd=no

# Starting a Quagga process

## 2. debian.conf

```
vtysh enable=no
```

```
zebra options="--daemon -A 127.0.0.1 -u quagga -g quagga"
```

```
bgpd options="--daemon -A 127.0.0.1 -u quagga -g quagga"
```

```
ospfd options="--daemon -A 127.0.0.1 -u quagga -g quagga"
```

```
ospf6d options="--daemon -A ::1 -u quagga -g quagga"
```

```
ripd options="--daemon -A 127.0.0.1 -u quagga -g quagga"
```

```
ripngd options="--daemon -A ::1 -u quagga -g quagga"
```

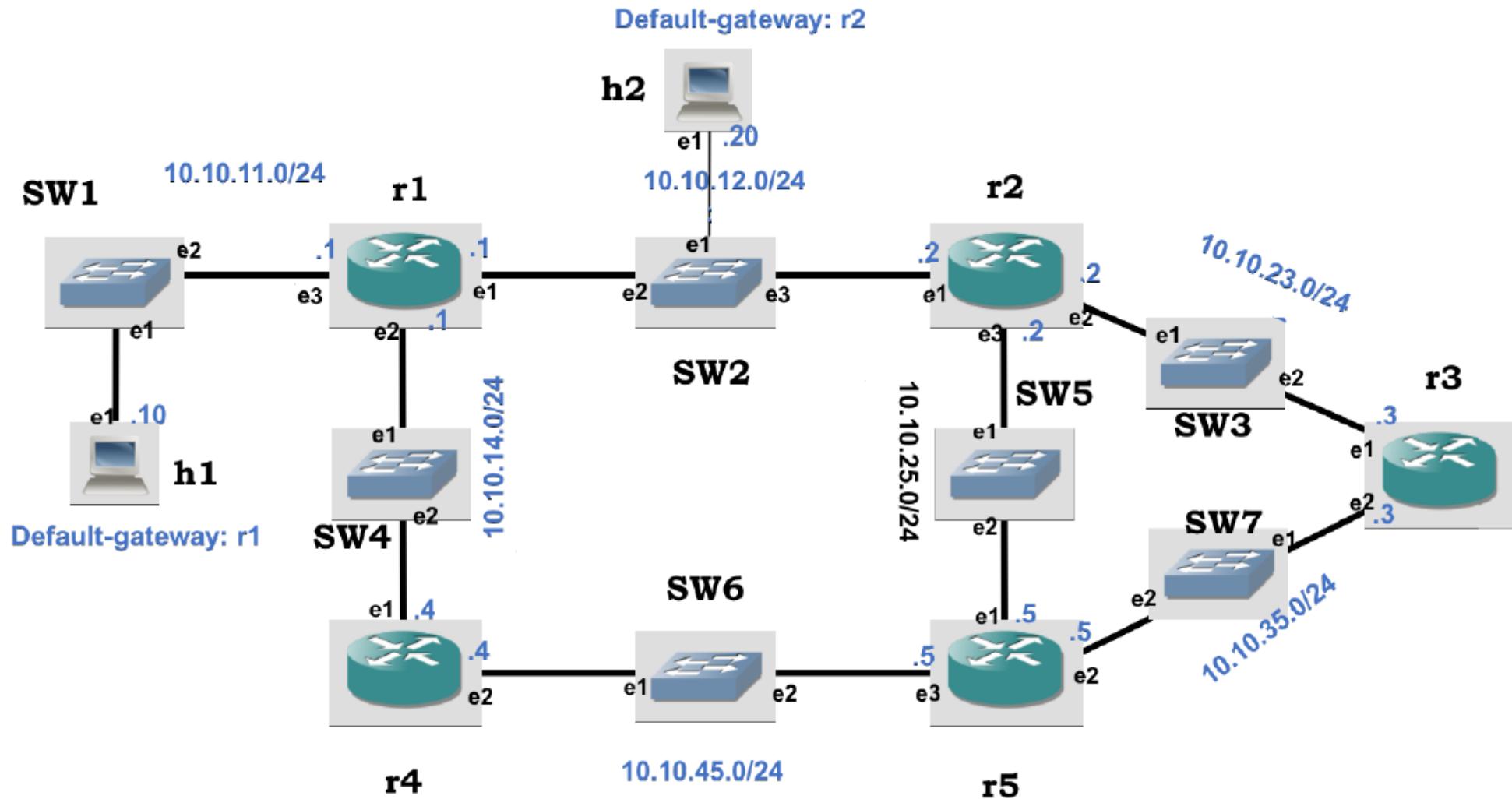
```
isisd options="--daemon -A 127.0.0.1 -u quagga -g quagga"
```

# Starting a Quagga process

## 3. zebra.conf

```
!  
! Zebra configuration file for r1  
!  
hostname r1  
password quagga  
enable password quagga  
  
log file /home/mininet/Desktop/shared/lab8/configs/r1/logs/zebra.log  
debug zebra packet  
  
interface r1-eth1  
no shutdown  
ip address 10.10.12.1/24  
  
line vty
```

# Network topology

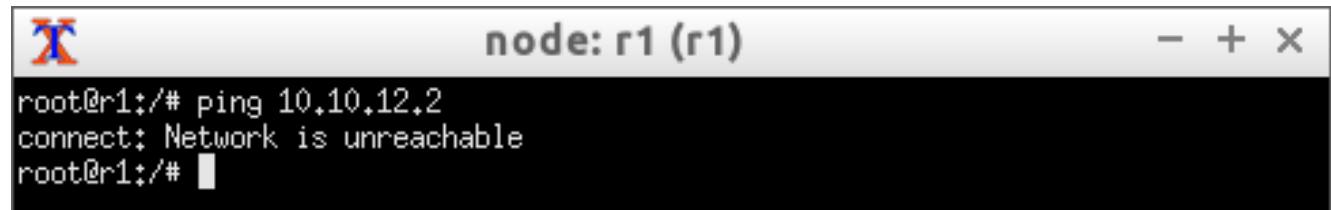


- \$ sudo python lab8.py

- mininext> net

```
mininext> net
h1 h1-eth1:SW1-eth1
h2 h2-eth1:SW2-eth1
r1 r1-eth3:SW1-eth2 r1-eth1:SW2-eth2 r1-eth2:SW4-eth1
r2 r2-eth1:SW2-eth3 r2-eth2:SW3-eth1 r2-eth3:SW5-eth1
r3 r3-eth1:SW3-eth2 r3-eth2:SW7-eth1
r4 r4-eth1:SW4-eth2 r4-eth2:SW6-eth1
r5 r5-eth1:SW5-eth2 r5-eth2:SW7-eth2 r5-eth3:SW6-eth2
SW1 lo: SW1-eth1:h1-eth1 SW1-eth2:r1-eth3
SW2 lo: SW2-eth1:h2-eth1 SW2-eth2:r1-eth1 SW2-eth3:r2-eth1
SW3 lo: SW3-eth1:r2-eth2 SW3-eth2:r3-eth1
SW4 lo: SW4-eth1:r1-eth2 SW4-eth2:r4-eth1
SW5 lo: SW5-eth1:r2-eth3 SW5-eth2:r5-eth1
SW6 lo: SW6-eth1:r4-eth2 SW6-eth2:r5-eth3
SW7 lo: SW7-eth1:r3-eth2 SW7-eth2:r5-eth2
c0
mininext>
```

- root@r1:/# ping 10.10.12.2



- Create/edit the configuration files for other routers:
  - \$ sudo leafpad configs/r2/daemons
  - \$ sudo leafpad configs/r2/debian.conf
  - \$ sudo leafpad configs/r2/zebra.conf
  - \$ sudo leafpad configs/r2/ripd.conf
- Start Quagga service on each router:
  - root@r1:/# /etc/init.d/quagga start

```
root@r1:/# /etc/init.d/quagga start
Loading capability module if not yet done.
Starting Quagga daemons (prio:10): zebra.
root@r1:/#
```

```
root@r2:/# /etc/init.d/quagga start
Loading capability module if not yet done.
Starting Quagga daemons (prio:10): zebra.
root@r2:/#
```

node: r1 (r1) - + ×

```
root@r1:/# ping 10.10.12.2
PING 10.10.12.2 (10.10.12.2) 56(84) bytes of data.
64 bytes from 10.10.12.2: icmp_seq=1 ttl=64 time=5.79 ms
64 bytes from 10.10.12.2: icmp_seq=2 ttl=64 time=0.670 ms
64 bytes from 10.10.12.2: icmp_seq=3 ttl=64 time=0.088 ms
64 bytes from 10.10.12.2: icmp_seq=4 ttl=64 time=0.086 ms
^C
--- 10.10.12.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.086/1.660/5.798/2.400 ms
root@r1:/#
```

# Quagga monitoring mode

- Connect to the Quagga process running on router:

- root@r1:/# telnet localhost zebra

```
root@r1:/# telnet localhost zebra
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r1> █
```

- Inspect the contents of the routing table:

- r1> show ip route

```
r1> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
      O - OSPF, I - IS-IS, B - BGP, A - Babel,
      > - selected route, * - FIB route

C>* 10.10.11.0/24 is directly connected, r1-eth3
C>* 10.10.12.0/24 is directly connected, r1-eth1
C>* 10.10.14.0/24 is directly connected, r1-eth2
C>* 127.0.0.0/8 is directly connected, lo
r1> █
```

- Check the status of the network interfaces:
  - r3> show interface

```
r3> show interface
Interface lo is up, line protocol detection is disabled
  index 1 metric 1 mtu 65536
  flags: <UP,LOOPBACK,RUNNING>
  inet 127.0.0.1/8
  inet6 ::1/128
Interface r3-eth1 is up, line protocol detection is disabled
  index 913 metric 1 mtu 1500
  flags: <UP,BROADCAST,RUNNING,MULTICAST>
  Hwaddr: fe:5e:63:a3:f1:59
  inet 10.10.23.3/24 broadcast 10.10.23.255
  inet6 fe80::fc5e:63ff:fea3:f159/64
Interface r3-eth2 is up, line protocol detection is disabled
  index 927 metric 1 mtu 1500
  flags: <UP,BROADCAST,RUNNING,MULTICAST>
  Hwaddr: 6a:aa:d8:e7:77:4d
  inet 10.10.35.3/24 broadcast 10.10.35.255
  inet6 fe80::68aa:d8ff:fee7:774d/64
r3> █
```

# Quagga configuration mode

- Enable the configuration mode:

- r3> enable
- (Password: quagga)

```
r3> enable  
Password:  
r3#
```

- Inspect the running configuration of router:

- r3# show running-config

```
r3# show running-config  
  
Current configuration:  
!  
hostname r3  
password quagga  
enable password quagga  
log file /home/mininet/Downloads/lab7/configs/r3/logs/zebra.log  
!  
debug zebra packet  
!  
interface lo  
!  
interface r3-eth1  
  ip address 10.10.23.3/24  
  ipv6 nd suppress-ra  
!  
interface r3-eth2  
  ip address 10.10.35.3/24  
  ipv6 nd suppress-ra  
!  
!  
!  
line vty  
  no login  
!  
end  
r3#
```

# On the fly configuration

- Enter the fly configuration mode:
  - r4# configure terminal
- Enter the configuration mode of interface:
  - r4(config)# interface r4-eth2
- Edit interface:
  - r4(config-if)# ip address 10.10.45.4/24

```
r4# configure terminal
r4(config)# interface r4-eth2
r4(config-if)# ip address 10.10.45.4/24
r4(config-if)# █
```

## آزمایشگاه شبکه

### آزمایش ۸: راهاندازی سرویس مسیریابی در Mininet

#### الف) معرفی بستر آزمایش

هسته شبکه جهانی اینترنت را روترهای قدرتمندی تشکیل می‌دهند که قادرند حجم بزرگی از ترافیک داده را مسیردهی نمایند. این روترا که نوعاً ساخت شرکت‌های مشهوری مثل: Juniper یا Huawei یا Cisco هستند از سیستم‌عامل‌های اختصاصی (مثل: Cisco IOS یا JunOS) استفاده می‌کنند که تعامل با آنها از طریق ترمینال‌های کنترلی ویژه پیکربندی شبکه صورت می‌گیرد. دستوراتی هم که برای پیکربندی این روترا استفاده می‌شود معمولاً با دستوراتی که شما در کنسول‌های Linux یا UNIX استفاده می‌کنید، تفاوت دارند.

در حالت ایده‌آل، می‌خواهیم امکان اجرای سیستم‌عاملی مثل IOS در یک محیط مجازی فراهم باشد. اگرچه این امکان به لحاظ فنی میسر است، اماً منع قانونی دارد چرا که شرکت‌هایی نظیر Cisco یا Juniper اجازه اجرای سیستم‌عامل‌های ایشان را روی تجهیزاتی غیر از روترهای خودشان نمی‌دهند.

در عوض، ما از Quagga استفاده خواهیم کرد که یک بستر رایگان نرم‌افزاری برای مسیریابی روی Linux است. Quagga یک ترمینال کنترلی به ما می‌دهد که در آن امکان اجرای دستوراتی است شبیه به آنچه در IOS سیسکو قابل اجراست. بعلاوه، Quagga را می‌توان از طریق MiniNExT با Mininet هم ترکیب کرد. MiniNExT یک گسترش از Quagga است که برای اجرای مستقل Quagga روی هر ماشین مجازی را فراهم می‌آورد.

\*توجه! محیط VM/ز قبیل روی VM‌هایی که در اختیار دارید، نصب شده است.

#### الف-۱) Quagga چیست و چگونه کار می‌کند؟

Quagga یک بستر نرم‌افزاری ویژه مسیریابی است که در آن چندین پروتکل مشهور (مثل: OSPF، BGP-4 و RIP) برای محیط‌های UNIX پیاده‌سازی شده‌اند. Quagga از چندین فرآیند تشکیل شده که می‌توانند به صورت daemon در پس‌زمینه اجرا شوند. سه مورد از Quagga daemon که برای انجام این آزمایش (و بعدی) اهمیت دارند، عبارتند از:

- zebra: که برای مدیریت اینترفیس‌های شبکه‌ای یک روتر استفاده می‌شود. این فرآیند امکان پیکربندی اینترفیس‌ها و مانیتور کردن وضعیت آنها را می‌دهد. بعلاوه، در مقایسه با دستور route-n، یک نمای جزئی‌تر از جداول مسیریابی را فراهم می‌آورد. به بیان دیگر، zebra یک جایگزین پیشرفته‌تر برای دستورات شبکه‌ای Linux است (یعنی: دستوراتی مثل: route و ifconfig و غیره).

- ripd: پیاده‌سازی ورژن 2.0 از پروتکل مسیریابی RIP را فراهم می‌آورد.
- quagga: سرویس اصلی که برای فراخوانی daemon‌های فوق بکار می‌رود.

جلوته، ملاحظه خواهیم کرد که MiniNExT برای هر «روتر مجازی<sup>1</sup>» یک namespace جدآگانه ایجاد می‌کند که به این معناست که هر روتر مجازی داخل محیط مجازی Mininet، سرویس quagga خاص خود را به طور مستقل اجرا می‌نماید.

## الف-۲) راهاندازی یک فرآیند Quagga

هر دارای فایل کانفیگ خاص خودش است که وجود آن برای اینکه Quagga به درستی کار کند، الزامی است (ولو به صورت خالی). در VM ای که در اختیار دارید، Quagga را به صورت یک سرویس Linux پیکربندی کرده‌ایم و برای راهاندازی آن باید دو فایل کانفیگ به نام‌های daemons و debian.conf را تنظیم نمایید.

در فایل daemons، شما محتوایی شبیه به زیر ملاحظه خواهید کرد:

```
zebra=yes
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no
isisd=no
```

به کمک این فایل، برای Quagga مشخص می‌کنید که تمایل دارید کدامیک از فرآیندها را فعال‌سازی نماید. توجه کنید که فرآیند zebra باید همواره فعال باشد.

در فایل debian.conf محتوایی نظیر زیر را ملاحظه خواهید نمود:

```
vtysh_enable=no
zebra_options=" --daemon -A 127.0.0.1 -u quagga -g quagga"
bgpd_options=" --daemon -A 127.0.0.1 -u quagga -g quagga"
ospfd_options=" --daemon -A 127.0.0.1 -u quagga -g quagga"
ospf6d_options="--daemon -A ::1 -u quagga -g quagga"
ripd_options=" --daemon -A 127.0.0.1 -u quagga -g quagga"
ripngd_options="--daemon -A ::1 -u quagga -g quagga"
isisd_options=" --daemon -A 127.0.0.1 -u quagga -g quagga"
```

<sup>1</sup> Virtual Router

در این فایل،

- -- به معنای این است که می‌خواهیم فرآیند مورد نظر در پس‌زمینه اجرا شود.
- A- برای مشخص کردن آدرس IP ویژه دسترسی به مُد پیکربندی است که به صورت پیش‌فرض همان localhost می‌باشد.
- u- نام کاربر آغاز‌کننده فرآیند را مشخص می‌کند (که به صورت پیش‌فرض همان quagga است).
- g- گروه کاربری (privilege) را مشخص می‌کند (که به صورت پیش‌فرض همان quagga است).

پس از پیکربندی صحیح فایل conf، می‌توانیم سرویس Quagga را مشابه هر سرویس تحت Linux دیگر راهاندازی نماییم:

```
/etc/init.d/quagga start
```

### الف-۳) خاتمه یک فرآیند Quagga

فرآیندهای Quagga را به دو روش می‌توان خاتمه داد:

۱- **توقف کامل سرویس quagga:** با استفاده از دستور سنتی توقف یک فرآیند در لینوکس:

```
/etc/init.d/quagga stop
```

۲- خاتمه انفرادی هر فرآیند: گاه‌آم ممکن است نیاز به شبیه‌سازی یک crash روی یک روتر باشد و باید بتوانیم که یک فرآیند خاص را kill کنیم. برای kill کردن انفرادی یک فرآیند خاص از Quagga که روی یک روتر مجازی در حال اجراست، دستور زیر را می‌توان در پنجره ترمینال تایپ نمود:

```
killall <process name 1> <process name 2> ... <process name N>
```

که در آن، عبارات `<process name 1>` تا `<process name N>` همان نام فرآیندهایی از Quagga هستند (مثل: `ripd` و `zebra` و غیره) که روی یک روتر مجازی در حال اجرا می‌باشند. توجه کنید که اگر بخواهید فرآیند مورد نظر را مجدداً راهاندازی نمایید، ناگزیر از reset سرویس quagga هستید چراکه این تنها گزینه برای آغاز کردن فرآیندهای Quagga است.

بالآخره اینکه، همیشه می‌توانید بررسی کنید که آیا یک فرآیند روی یک روتر مجازی در حال اجراست یا خیر. برای این منظور، کافی است که لیست فرآیندهای در حال اجرا را با استفاده از دستور زیر ملاحظه نمایید:

```
ps -A
```

## ب) ادغام Mininet در Quagga با استفاده از MiniNExT

یا همان MiniNExT (Mininet Extended) یک لایه گسترشی بر Mininet است که namespace های فرآیندها، runtime وfilesystem های mount کردن همینطور، ایزوله سازی log ها و فرآیندها را فراهم می نماید. این محیط با هدف ایجاد شبکه های پیچیده تر از Mininet تنها توسعه داده شده است. محیط MiniNExT دارای یک سری کتابخانه های Python خاص Quagga است که ما از آنها برای ادغام Quagga در محیط مجازی Mininet استفاده می نماییم.

برای آزمایش ۸، اسکریپت های Python مورد نیاز را برای فهم ساده تر آنها، به دو بخش تقسیم کرده ایم:

۱- چون توپولوژی در حین آزمایش تغییر نخواهد کرد، می توانیم از یکی از اسکریپت های Python برای ایجاد یک کلاس توپولوژی استفاده کنیم. به کمک آن، خواهیم توانست روترهای مبتنی بر quagga بسازیم، host و link اضافه کنیم.

۲- با اسکریپت دیگر می توان محیط مجازی را اجرا نمود؛ یعنی، توپولوژی مورد نظر را فراخوانی کرد و سپس، خط دستور و پنجره ترمینال برای پیکربندی روترهای را اضافه نمود.

هر دو اسکریپت به صورت آماده در اختیار شما قرار داده شده است.

- [فایل های lab8.py و lab8\\_topo.py را باز کرده و کامنت های برنامه و توضیحات بخش های ب-\(۱\) و ب-\(۲\) در زیر را مطالعه نمایید.](#)

### ب-۱) کلاس توپولوژی

در خصوص اسکریپت با عنوان lab8\_topo.py، توضیحات زیر را مدنظر قرار دهید:

- اگر دقت نمایید، کلاس Topo را از بسته MiniNExT و نه از Mininet وارد کرده ایم. علت این امر آن است که کلاس فراهم شده توسط MiniNExT دارای یک تابع برای اضافه کردن یک node service است که برای عملیات Quagga ضروری می باشد (خط ۱۳ را ملاحظه نمایید).

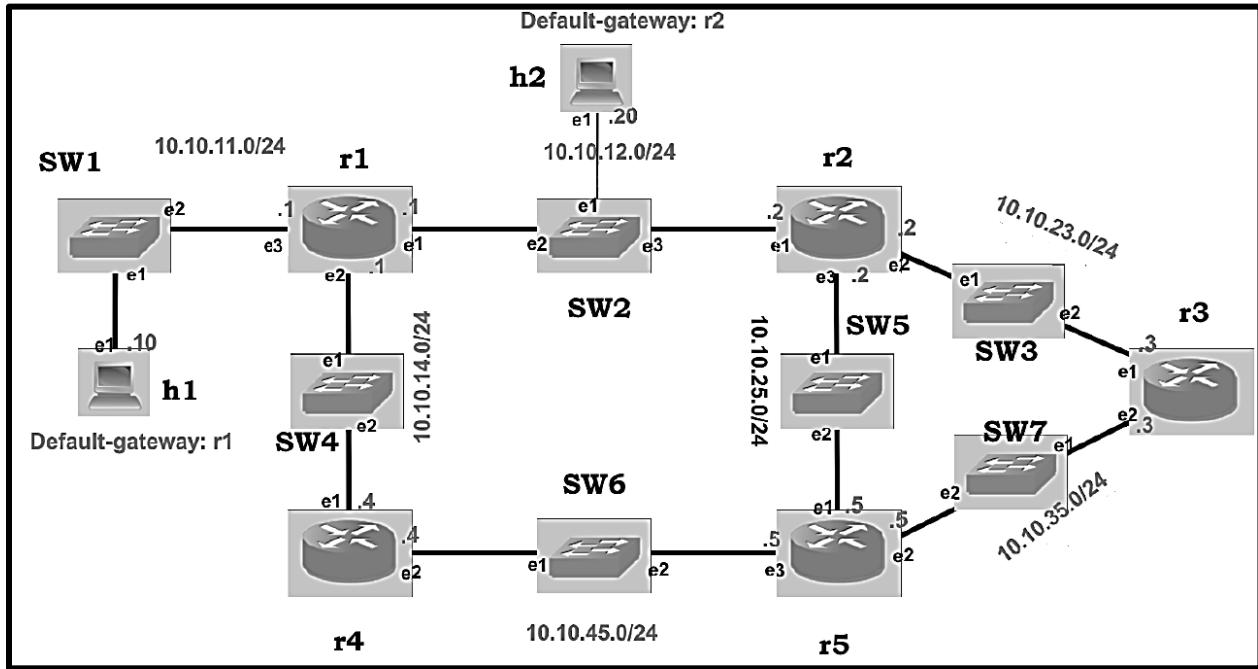
- این اسکریپت فرض کرده است که شما دارای یک فolder به نام configs در همان مسیر خود اسکریپت هستید و ضمناً داخل این فolder هم یک فolder به ازای هر روتر نیاز داریم که دقیقاً همانم با خود روتر است (مثلاً r1، r2 و غیره). خطوط ۴۷ و ۷۵-۷۶ را می توان برای تغییر این رفتار استفاده نمود.

- با استفاده از خط ۴۱ می توان فرآیند quagga را طوری پیکربندی کرد که به طور خودکار آغاز یا پایان یابد.

- از یک چندتایی نامدار (اصطلاحاً named tuple) برای مدیریت attribute هایی که تمایل داریم هنگام اضافه کردن روترهای مجازی در شبکه لحاظ کنیم، استفاده می‌شود. اگر بخواهید این attribute ها را اصلاح یا زیاد و کم کنید (مثل: آدرس لوپ‌بک برای هر روتر)، می‌توانید متغیر خط ۲۲ را اصلاح نموده و بعد به سراغ اصلاح بخشی بروید که با خط شماره ۵۶ آغاز می‌شود.
- هنگام اضافه کردن یک روتر مجازی به شبکه، باید قابلیت ایزوله‌سازی فرآیندها، logها و اجرا را فعال نماییم که این کار در خط شماره ۷۲ صورت گرفته است.

## ب-۲) اسکریپت شبیه‌سازی

- در خصوص اسکریپت با عنوان lab8.py، توضیحات زیر را مدنظر قرار دهید:
- در مستندات MiniNExT پیشنهاد شده است که ماجول isShellBuiltIn را پچ نمایید چراکه ممکن است هنگام اجرای گُد bash از ترمینال MiniNExT ایجاد اشکال نماید. این گُد در خطوط ۱۷ الی ۲۰ نشان داده شده است.
  - برخلاف یک اسکریپت متداول Mininet، در MiniNExT از تابع سازنده MiniNExT برای ساخت شبکه استفاده می‌شود (خطوط ۳۶ و ۵۸ را ملاحظه نمایید).
  - با وجودی که می‌توانید هر فرمانی را از طریق خط دستور mininext> اجرا نمایید، راحت‌تر خواهیم بود اگر که برای هر روتر یک پنجره ترمینال جداگانه باز کنیم. گُد نوشته شده در خطوط ۶۸ تا ۷۴ این قابلیت را فراهم می‌سازد.
  - تابع stopNetwork() به نحو مناسبی کلیه فایل‌های log را هنگام توقف اسکریپت حذف می‌نماید. برای تغییر این رفتار (یعنی: برای حفظ logها پس از خروج از اسکریپت) باید خطوط ۹۰ الی ۹۷ را کامنت کرد.
  - حال، آمده استفاده از اسکریپت‌های فوق هستیم. ابتدا بررسی کنید که سرویس quagga را طوری تنظیم کرده باشید که به صورت اتوماتیک شروع نشود و سپس، محیط شبیه‌سازی را اجرا نمایید.
  - **سؤال ۱: از خط دستور net دستور Mininet را اجرا نموده و بررسی کنید که شبکه مطابق با شکل ۱ ساخته شده باشد. آیا می‌توانید بین همگی روترهای مجازی ping کنید؟ توضیح دهید چرا می‌توانید یا چرا نمی‌توانید؟!**



شکل ۱- توبولوژی آزمایش ۸

### ج) پیکربندی شبکه با استفاده از Quagga

برای این بخش، برنامه را کامل ببندید. ادامه توضیحات را مطالعه نموده و مابقی مراحل آزمایش را طی کنید:

هر روتر مجازی، در واقع، مشابه یک ماشین فیزیکی Linux است و بنابراین می‌توان به همان طریق هم آن را پیکربندی نمود؛ یعنی، شما می‌توانید همان مجموعه دستوراتی را که در آزمایش‌های پیشین فرا گرفته‌اید برای پیکربندی اینترفیس‌های شبکه استفاده نموده و وضعیت آنها را مانیتور نمایید یا اینکه محتوای جداول مسیریابی آنها را مشاهده کنید و غیره.

با این حال، در این آزمایش، به جای استفاده از مجموعه دستورات شبکه‌ای Linux، از قابلیت‌های فراهم شده توسط Quagga بهره خواهیم گرفت. یکی از مزیت‌های Quagga این است که دستوراتی که توسط آن پشتیبانی می‌شوند، در واقع، زیرمجموعه‌ای از دستورات مورد استفاده برای پیکربندی تجهیزات شبکه‌ای Cisco هستند.

مستندات کامل کار با Quagga از طریق [www.nongnu.org/quagga/](http://www.nongnu.org/quagga/) قابل دستیابی است ضمن اینکه می‌توان از خود سایت Cisco هم به مراجع مفیدی در این رابطه دست یافت.

\* توجه! هرگز تلاش نکنید که اینترفیس‌های شبکه‌ای روی یک ماشین را به طور همزمان توسط هر دو نوع دستور ipaddr و دستورات فرآیند zebra پیکربندی نمایید. در واقع، تعامل این دو همیشه به خروجی قطعی و قابل اطمینانی منجر نمی‌گردد.

## ج-۱) پیکربندی اینترفیس‌های شبکه با استفاده از فایل‌های پیکربندی

پیش از اجرای توپولوژی، نیاز است که حداقل برای فرآیند zebra، یک فایل پیکربندی ساخته شده و ویرایش گردد. فایل با نام zebra.conf که در اختیار دارید، نمونه‌ای از چنین فایلی است که باید به طور اختصاصی برای هر روتر نوشته شود.

- **فایل zebra.conf ویژه روتر r1 را با استفاده از برنامه leafpad باز کنید و محتوای آن را به همراه توضیحات زیر مطالعه نمایید.**

- خطوطی که با نشانه ! شروع می‌شوند، کامنت هستند و از آنها صرف‌نظر خواهد شد.
  - :کلمه عبور مورد استفاده برای login کردن در فرآیند zebra و مشاهده تغییرات password
  - :کلمه عبور مورد استفاده برای فعال‌سازی قابلیت پیکربندی پویای فرآیند zebra enable password
  - :امکان می‌دهد که شما فایل log را که داخل آن رویدادهای مربوط به zebra ضبط می‌شود، مشخص نمایید (این رویدادها عمدتاً مربوط به اضافه و حذف مسیرها هستند). اطمینان حاصل کنید که مسیر کامل را برای این فایل‌ها مشخص نموده‌اید چراکه در غیر این صورت، سرویس Quagga برای استارت خود به مشکل می‌خورد.
  - :امکان debugging zebra packet جزئی‌تری را فراهم می‌نماید؛ یعنی، می‌توانید مشاهده کنید که چه مسیرهایی در جدول مسیریابی اضافه و حذف می‌شوند.
  - :این دستور مُد پیکربندی اینترفیس را آغاز می‌کند. interface <interface name>
  - :آدرس IP ورژن ۴ را برای اینترفیس مورد نظر تعیین می‌کند. ip address
  - :قابلیت دسترسی مبتنی بر telnet به فرآیند را می‌دهد. line vty
- کلیه اسکریپت‌های پیکربندی ویژه روترهای r1 و r4 از پیش تهیه شده‌اند و شما می‌توانید از طریق فolder8 که در اختیار دارید، به آنها دسترسی داشته باشید.
- **فایل‌های پیکربندی لازم برای روترهای r2 r3 و r5 را بسازید (شامل چهار فایل: .daemons، .ripd.conf و zebra.conf، debian.conf شبکه، اینترفیس‌های روترها را پیکربندی نمایید (آدرس IP تخصیص دهید). همچنین، دقت کنید که در کلیه فایل‌های zebra، صرفاً فرآیند zebra فعال شده باشد و مابقی فرآیندها را با "no" تنظیم کرده باشید.**

\* اطمینان حاصل کنید که اجازه دسترسی (permission) فایل‌های zebra.log طوری باشد که به شما امکان دسترسی read-write بدهد. برای اینکه بتوان یک اسکریپت را برای همه (anyone)، از نوع read-write کرد، می‌توانید دستور زیر را در پنجره ترمینال تایپ کنید (با داشتن دسترسی root):

```
chmod 666 <script name>
```

## ج-۲) مُد مانیتورینگ Quagga

حال اسکریپت با عنوان lab8.py را مجدداً اجرا نمایید.

- سرویس quagga را در کلیه روت‌ها استارت کنید (البته پیش از آغاز فرآیند quagga باید مطمئن شوید که در فایل daemons، گزینه zebra=yes وجود داشته باشد). برای آغاز کردن سرویس quagga می‌توان از ترمینال هر روت‌ی، دستور زیر را تایپ نمود:

```
/etc/init.d/quagga start
```

- البته به عنوان راهکار دیگر، می‌توانستیم از طریق فایل lab8\_topo.py مقرر کنیم که سرویس Quagga به طور خودکار استارت شود.

به منظور مانیتور کردن فعالیت یک فرآیند Quagga در حال اجرا، می‌توانید از طریق telnet به فرآیند مورد نظر متصل شده و وارد مُد مانیتورینگ شوید.

- با استفاده از دستور زیر به فرآیند zebra در حال اجرا روی روت‌r1 متصل شوید:

```
telnet localhost zebra
```

- به منظور مشاهده محتوای جداول مسیریابی IPv4 در روت‌r1، دستور show ip route را تایپ نمایید.

همواره می‌توانید لیست کامل دستورات مورد پشتیبانی را با تایپ دستور list مشاهده نمایید.

- **سؤال ۲: چه subnet‌هایی را می‌توان از روی جدول مسیریابی موجود در r1 تشخیص داد؟**

- حال، وضعیت اینترفیس‌های شبکه روی روت‌r3 را با تایپ دستور show interface بررسی کنید.

- **سؤال ۳: چند اینترفیس نمایش داده می‌شود؟ نام ببرید.**

- برای مشاهده پیکربندی جاری و در حال اجرا، باید ابتدا با استفاده از دستور enable، مُد پیکربندی را فعال کنید.

- در نهایت، پیکربندی در حال اجرای روتر r3 را با استفاده از دستور `show running-config` بررسی کنید.