

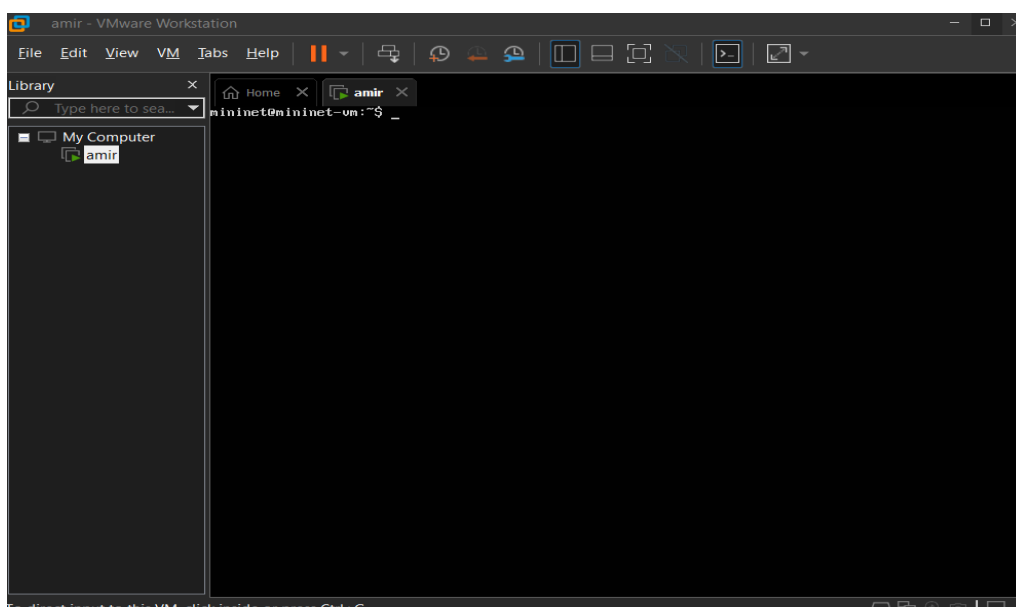
گزارش پروژه

امیرحسین ثابتی – 98243015

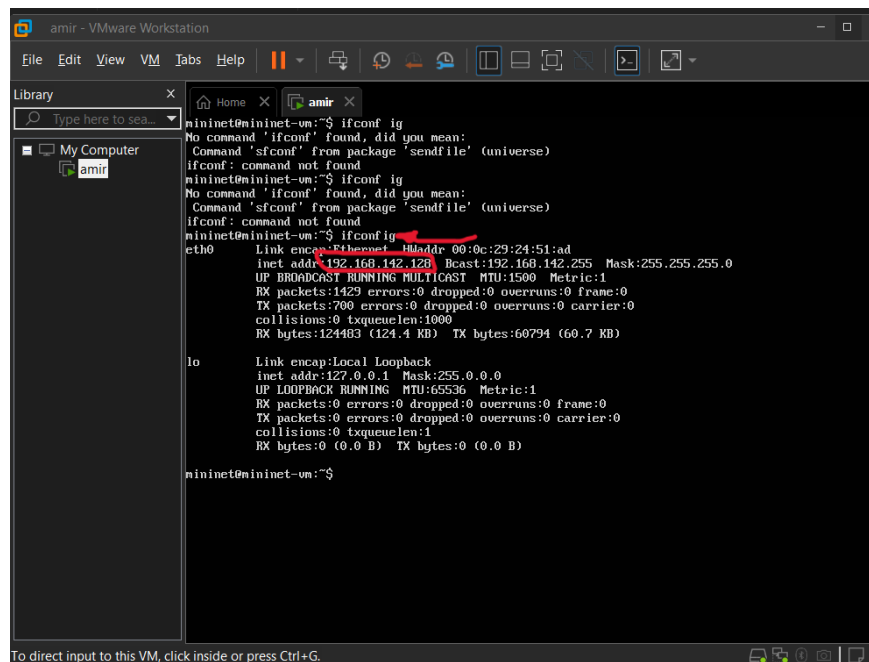
عرفان رفیعی اسکویی – 98243027

در ابتدا نرم افزار **vmware** را نصب و راه اندازی کردیم و در ادامه از لینک داده شده در صورت پروژه جهت دانلود **mininet** استفاده کردیم.

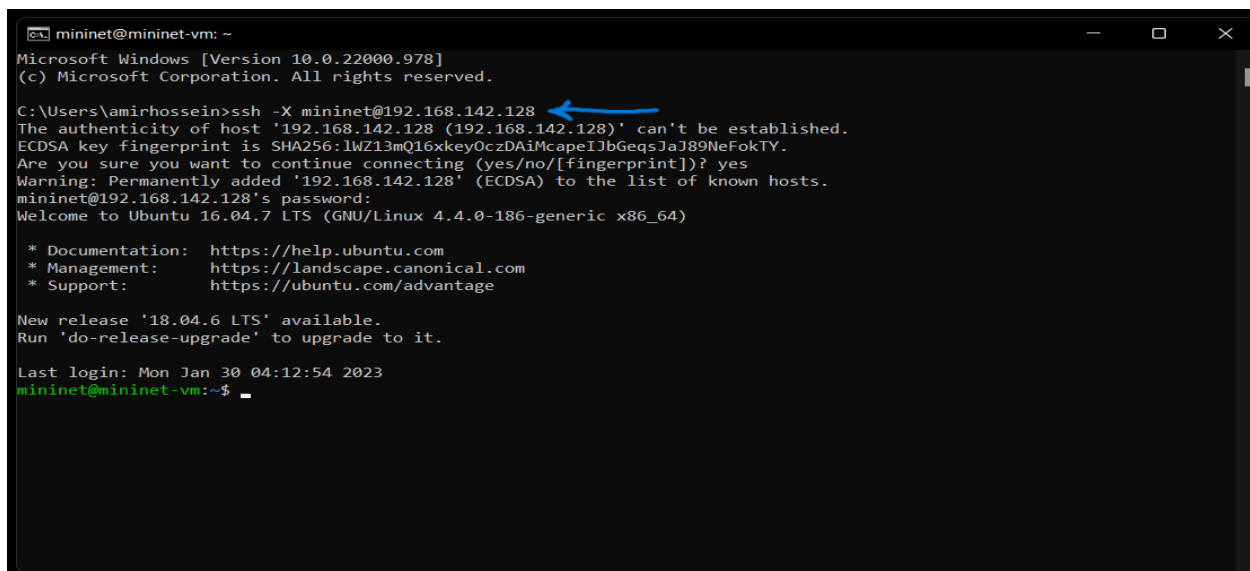
پس از دانلود با دو فایل با پسوند های **.ovf** و **.vmdk** روبرو شدیم و به کمک **vmware** فایل **.vmdk** را باز کردیم و با ترمینال لینوکس مواجه شدیم. لازم به ذکر است در این مرحله از ما یوزرنیم و پسورد می خواست که **mininet** بود جفتش.



در ادامه جهت ریموت زدن به این ترمینال نیاز به دانستن آی پی داشتیم که با دستور مشخص شده در عکس زیر آیپی لوکال سیستم را جهت ریموت زدن استخراج کردیم.



بعد از پیدا کردن ip ترمینال، با کمک command ویندوز و دستور ssh اتصال را برقرار می کنیم.



1) برای ساخت توپولوژی ذکر شده از دستور زیر کمک میگیریم و یک توپولوژی با 3 هاست و یک سویچ میسازیم.

```
mininet@mininet-vm: ~  
mininet@mininet-vm:~$ sudo mn --topo single,3  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2 h3  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1) (h3, s1)  
*** Configuring hosts  
h1 h2 h3  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet>
```

حالا برای نمایش نود ها از دستور `nodes` استفاده می کنیم و نود ها را نمایش می دهیم. که در نتیجه یک کنترلر سه هاست و یک سویچ میبینیم.

```
mininet@mininet-vm: ~  
c0  
*** Stopping 3 links  
...  
*** Stopping 1 switches  
s1  
*** Stopping 3 hosts  
h1 h2 h3  
*** Done  
completed in 75.204 seconds  
mininet@mininet-vm:~$ clear  
mininet@mininet-vm:~$ sudo mn --topo single,3  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2 h3  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1) (h3, s1)  
*** Configuring hosts  
h1 h2 h3  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet> nodes  
available nodes are:  
c0 h1 h2 h3 s1  
mininet>
```

برای نشان دادن لینک ها از دستور `links` استفاده میکنیم. و در تصویر مشخص است که لینک ها بین سویچ و هاست برقرار شده است.

```
mininet@mininet-vm: ~
s1
*** Stopping 3 hosts
h1 h2 h3
*** Done
completed in 75.204 seconds
mininet@mininet-vm:~$ clear
mininet@mininet-vm:~$ sudo mn --topo single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
h3-eth0<->s1-eth3 (OK OK)
mininet>
```

و در نهایت برای نشان دادن آدرس ip های اختصاص یافته شده از دستور `dump` استفاده می کنیم. و در عکس زیر آی پی مربوط به هر نود قابل رویت است.

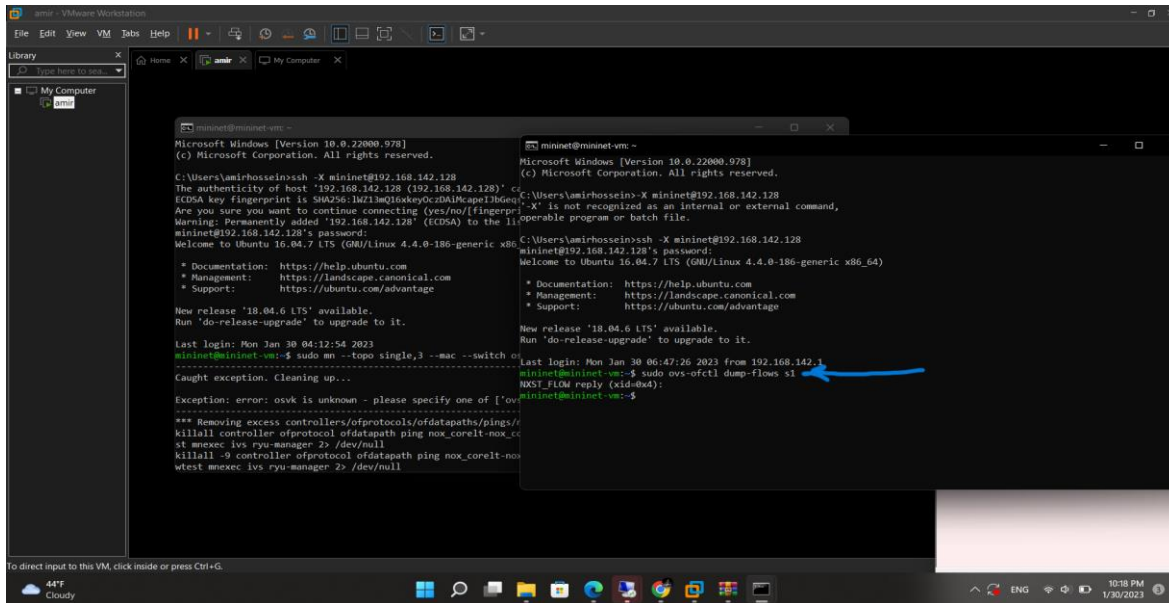
```
mininet@mininet-vm: ~  
mininet@mininet-vm:~$ sudo mn --topo single,3  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2 h3  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1) (h3, s1)  
*** Configuring hosts  
h1 h2 h3  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet> nodes  
available nodes are:  
c0 h1 h2 h3 s1  
mininet> links  
h1-eth0<->s1-eth1 (OK OK)  
h2-eth0<->s1-eth2 (OK OK)  
h3-eth0<->s1-eth3 (OK OK)  
mininet> dump  
<Host h1: h1-eth0:10.0.0.1 pid=23217>  
<Host h2: h2-eth0:10.0.0.2 pid=23220>  
<Host h3: h3-eth0:10.0.0.3 pid=23223>  
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=23229>  
<Controller c0: 127.0.0.1:6653 pid=23210>  
mininet>
```

سپس با دستور exit از توپولوژی ساخته شده خارج می شویم.

(2) شبکه ذکر شده به کمک کامند ایجاد کرده

```
mininet@mininet-vm: ~  
*** Removing all links of the pattern foo-ethX  
ip link show | egrep -o '([_.[:alnum:]]+-eth[[:digit:]]+)'  
ip link show  
*** Killing stale mininet node processes  
pkill -9 -f mininet:  
*** Shutting down stale tunnels  
pkill -9 -f Tunnel=Ethernet  
pkill -9 -f .ssh/mn  
rm -f ~/.ssh/mn/*  
*** Cleanup complete.  
mininet@mininet-vm:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remote  
*** Creating network  
*** Adding controller  
Unable to contact the remote controller at 127.0.0.1:6653  
Unable to contact the remote controller at 127.0.0.1:6653  
Setting remote controller to 127.0.0.1:6653  
*** Adding hosts:  
h1 h2 h3  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1) (h3, s1)  
*** Configuring hosts  
h1 h2 h3  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet>
```

از آنجا که سوییچ با کنترلر ارتباط ندارد پس امکان مسیریابی و ارسال بسته از یک هاست به هاست دیگر وجود ندارد. به همین دلیل ابتدا نیاز است جریان‌های موجود بین هاست‌ها را به صورت دستی تعریف کنیم. که برای اینکار نیاز به باز کردن ترمینال جدید داریم و دوباره از دستور ssh جهت انجام اینکار استفاده میکنیم.



جریان ها شامل تمام بسته های وارد شده از هاست یک و خارج شده از هاست 2 و همچنین وارد شده از هاست 2 و خارج شده از هاست 1 و همچنین وارد شده از هاست 2 و خارج شده از هاست 3 و همچنین وارد شده از هاست 3 و خارج شده از هاست 2 می باشد.

```

mininet@mininet-vm: ~
NXST_FLOW reply (xid=0x4):
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=output:2
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
sudo: ovs-ofctl: command not found
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=50.506s, table=0, n_packets=0, n_bytes=0, idle_age=50, in_port=1 act
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=output:1
sudo: ovs-ofctl: command not found
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=output:1
sudo: ovs-ofctl: command not found
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=output:1
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=207.281s, table=0, n_packets=0, n_bytes=0, idle_age=207, in_port=1 a
  cookie=0x0, duration=5.764s, table=0, n_packets=0, n_bytes=0, idle_age=5, in_port=2 acti
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=output:3
sudo: ovs-ofctl: command not found
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=output:3
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=281.689s, table=0, n_packets=3, n_bytes=182, idle_age=44, in_port=1
  cookie=0x0, duration=10.641s, table=0, n_packets=0, n_bytes=0, idle_age=44, in_port=2 act
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=3,actions=output:2
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=295.742s, table=0, n_packets=3, n_bytes=182, idle_age=58, in_port=1
  cookie=0x0, duration=24.694s, table=0, n_packets=0, n_bytes=0, idle_age=58, in_port=2 act
  cookie=0x0, duration=1.721s, table=0, n_packets=0, n_bytes=0, idle_age=1, in_port=3 acti
mininet@mininet-vm:~$

```

حالا طبق گفته ی صورت سوال ابتدا بسته ای را از h1 به h2 به کمک کامند مشخص شده در عکس زیر ارسال می کنیم و نتیجه را میبینیم که موفق بوده است. و بسته ارسال شده است.

```

mininet> h1 ping -c 1 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=2.10 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.108/2.108/2.108/0.000 ms

```

حالا طبق گفته ی صورت سوال بسته ای را از h2 به h3 به کمک کامند مشخص شده در عکس زیر ارسال می کنیم و نتیجه را میبینیم که موفق بوده است.

```

mininet> h2 ping -c 1 h3
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=2.01 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.015/2.015/2.015/0.000 ms
mininet>

```

با کمک دستور iperf میتوانیم پهنای باند ارتباط های ورودی و خروجی را ببینیم و از آنجا که هیچ ارتباطی بین هاست 1 و هاست 3 برقرار نیست، پس امکان برقراری ارتباط و محاسبه پهنای باند بین این دو وجود ندارد و پس این دستور با خطا مواجه شده و کل شبکه میپرد.

```

*** Unknown command: iperf
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
no route to 10.0.0.3:
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.0.0.0 * 255.0.0.0 U 0 0 0 h1-eth0

Caught exception. Cleaning up...

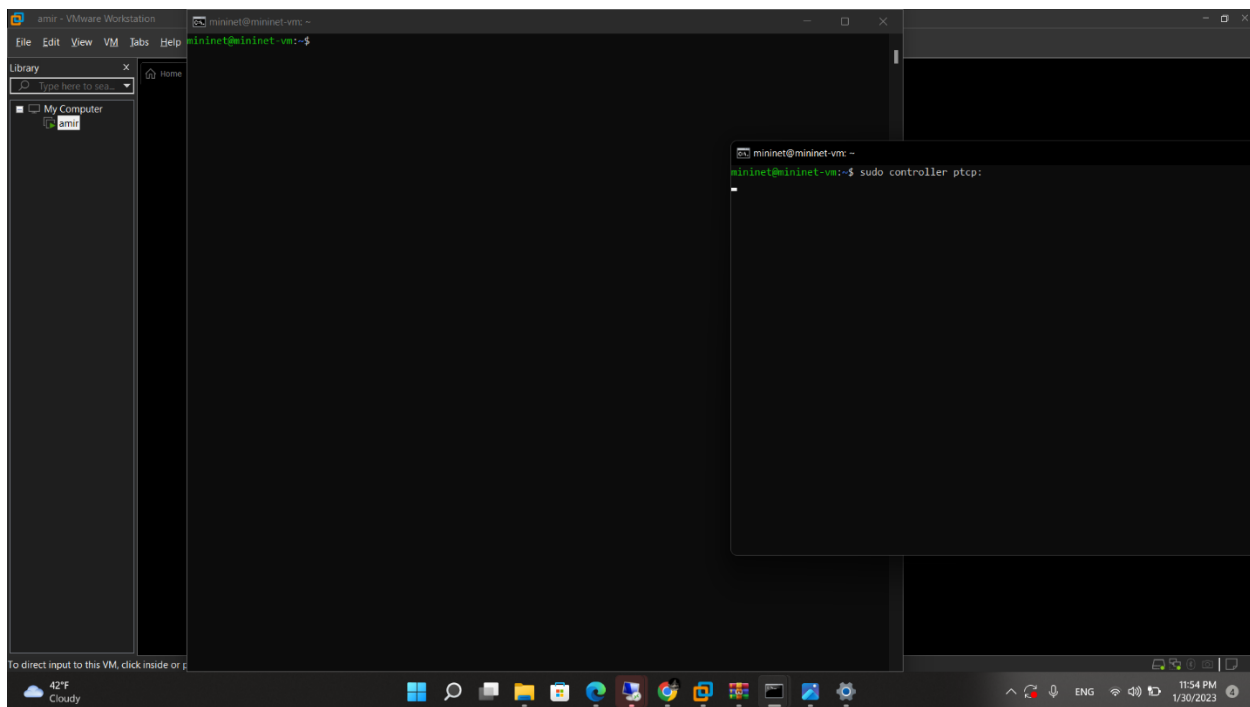
Exception: Could not connect to Iperf on port 5001

*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflow ovs-controllerovs-testcontroller udpbate
st mnexec lvs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflow ovs-controllerovs-testcontroller udpb
wtest mnexec lvs ryu-manager 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old xtl tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]*' | sed 's/dp/nl:/'
*** Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --if-exists del-br s1
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([a-z0-9]+)-eth[0-9]+' | sed 's/dp/nl:/'
( ip link del s1-eth1; ip link del s1-eth2; ip link del s1-eth3 ) 2> /dev/null
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
mininet@mininet ~$

--ofctl add-flow s1 in_port=1,actions=output:2
--ofctl dump-flows s1
--ofctl dump-flows s1
table=0, n_packets=0, n_bytes=0, idle_age=50, in_port=1 act
--ofctl add-flow s1 in_port=2,actions=output:1
round
--ofctl add-flow s1 in_port=2,actions=output:1
round
--ofctl dump-flows s1
table=0, n_packets=0, n_bytes=0, idle_age=207, in_port=1
table=0, n_packets=0, n_bytes=0, idle_age=5, in_port=2 act
--ofctl add-flow s1 in_port=2,actions=output:3
round
--ofctl add-flow s1 in_port=2,actions=output:3
--ofctl dump-flows s1
table=0, n_packets=3, n_bytes=182, idle_age=44, in_port=1
table=0, n_packets=0, n_bytes=0, idle_age=44, in_port=2 act
--ofctl add-flow s1 in_port=3,actions=output:2
--ofctl dump-flows s1
table=0, n_packets=3, n_bytes=182, idle_age=58, in_port=1
table=0, n_packets=0, n_bytes=0, idle_age=58, in_port=2 act
table=0, n_packets=0, n_bytes=0, idle_age=1, in_port=3 act

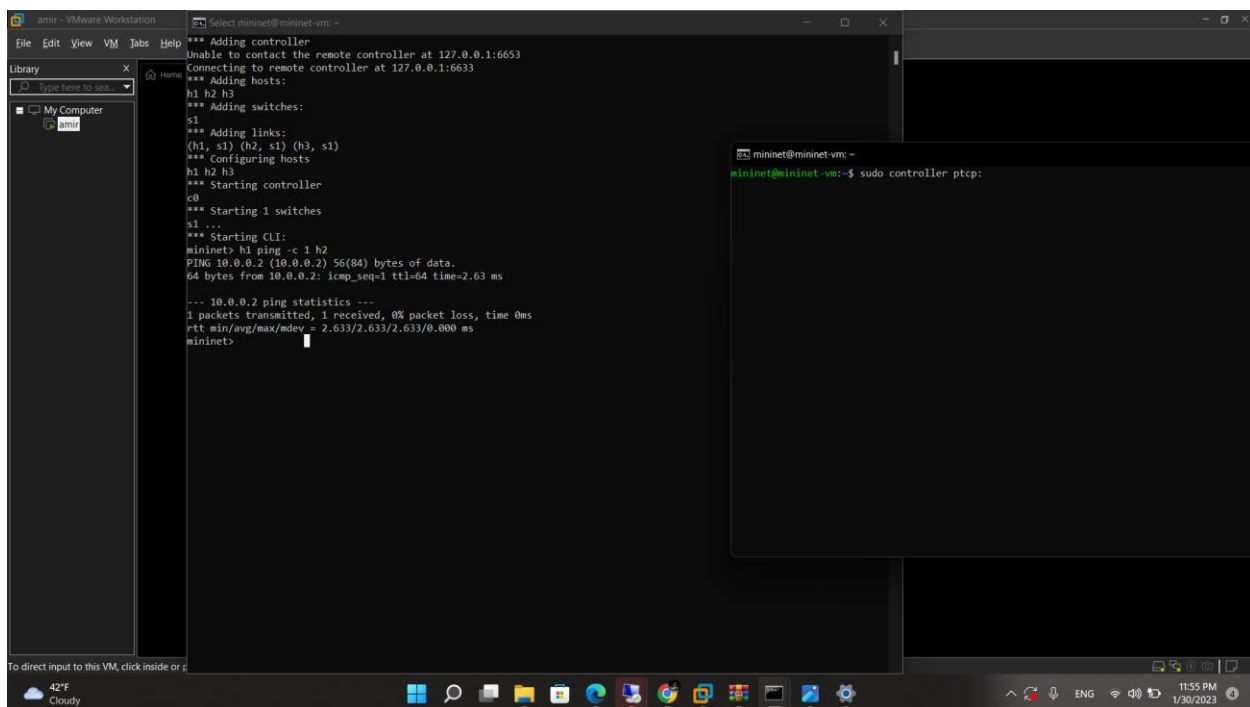
```

3) ابتدا با دستور صورت پروژه کنترلری که به صورت پیش فرض بر روی پورت TCP-6633 به درخواست وارد شده جهت اتصال از سمت سویچ ها گوش میدهد میسازیم.



سپس ترمینال جدید با کمک دستور ssh باز می کنیم و توپولوژی ذکر شده در مرحله ی قبل را دوباره می سازیم.
کنترلر ما بصورت tcp ست شده و الان در حال اجراست.

در این شرایط دیگر نیازی به تعریف ارتباط بصورت دستی نیست و بسته ها به درستی و بدون بروز مشکل ارسال می شوند.



دستور ipref در این مرحله با مشکل روبرو نمی شود زیرا تمامی هاست ها به هم متصل هستند.

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['17.5 Gbits/sec', '17.4 Gbits/sec']
mininet>
```

4) در این مرحله با توجه به اینکه باید از برنامه **wireshark** استفاده می کنیم که برای این است که تمامی پکت ها وارد به سیستم را دریافت و مانیتور میکند اما برای این کار نیاز به **linux** داشتیم و از آنجا که سیستم عامل قبلی که باهاش سوال ها را حل کردیم صرفا ترمینال لینوکس بود و **gui** نداشت پس مجبور به نصب لینوکس شدیم و روی لینوکس **mininet** و **wireshark** را نصب کردیم سپس دقیقا مانند قسمت قبل بالا توپولوژی را بالا آوردیم.

در ابتدا هنگام زدن دستور به مشکل پایین برخوردیم :

```
rfsk@ubuntu: ~
rfsk@ubuntu:~$ sudo controller ptcp:
[sudo] password for rfsk:
sudo: controller: command not found
rfsk@ubuntu:~$
```

برای حل آن با دستور زیر که در اینترنت پیدا کردیم **controller** اوپن فلو را نصب کردیم :

If you wish to go through the Mininet walkthrough, you will want to install additional software. The following cc

```
git clone https://github.com/mininet/mininet
mininet/util/install.sh -fw
```

will install the OpenFlow reference switch, reference controller and Wireshark dissector

حال دوباره دستور را اجرا میکنیم :

```
rfsk@ubuntu: ~  
rfsk@ubuntu:~$ sudo controller ptcp:  
[sudo] password for rfsk:
```

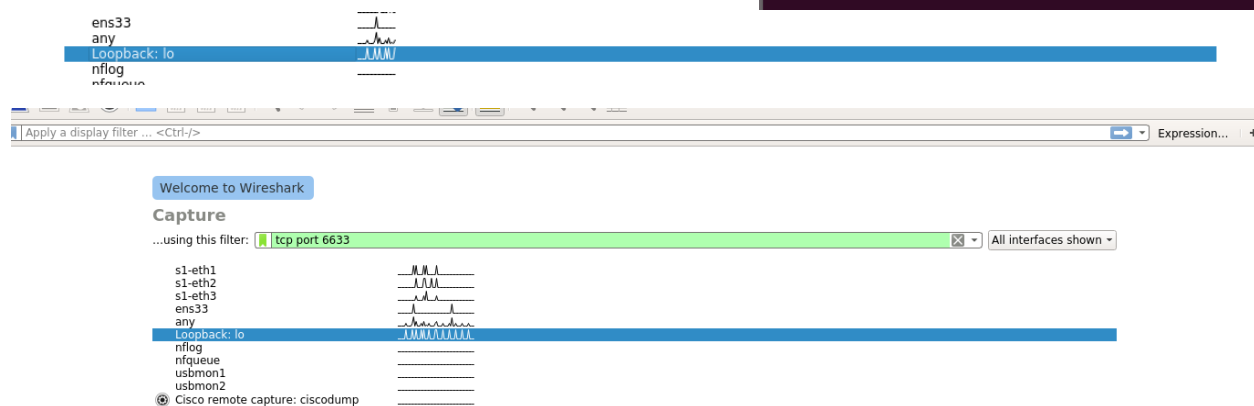
و بعد دستور مرتبط با اجرای mininet را اجرا میکنیم :

```
rfsk@ubuntu: ~  
rfsk@ubuntu:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remote  
[sudo] password for rfsk:
```

```
rfsk@ubuntu: ~  
rfsk@ubuntu:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remote  
[sudo] password for rfsk:  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2 h3  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1) (h3, s1)  
*** Configuring hosts  
h1 h2 h3  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet>
```

سپس در این مرحله نوبت به باز کردن نرم افزار wireshark و ست کردن کارت شبکه که lo است و همچنین capturefilter آن را بروی پورت 6633 قرار می دهیم.

```
rfsk@ubuntu: ~  
rfsk@ubuntu:~$ sudo wireshark  
[sudo] password for rfsk:
```



بعد از ست کردن wireshark نوبت به ارسال یک بسته و مشاهده روند آن در wireshark می شود که در اینجا بسته ای از h1 به h2 ارسال کرده و در وایر شارک تمامی بسته های وارده قابل مشاهده است. بسته های رد بدل شده بین سویچ و کنترلر شامل oftp_features_request , oft_set_config و تمامی بسته های مشخص شده در عکس زیر می باشد.

```

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping -c h2
Usage: ping [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
          [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
          [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
          [-w deadline] [-W timeout] [hop1 ...] destination
mininet>

```

