«به نام پروردگار»

گزارش پروژه دوم شبکه های کامپیوتری

«OpenFlow-tutorial»

استاد: دکتر پویا حجازی

دانشجو: محسن محمدیان

شماره دانشجویی: 9831502

بررسی اجمالی:

این پروژه به مفاهیمی چون SDN controller و مدیریت شبکه به صورت متمرکز یا centralized می پردازد و برای این کار از رابط (interface) open flow استفاده می کند که به کمک آن ما می توانیم روترها و سوییچ ها را program کنیم و به کمک آن جدول های مسیریابی یا flow table های این تجهیزات را مدیریت کنیم.

همانطور که می دانیم open flowاز 4 درایه یا ویژگی تشکیل شده است:

1. Pattern: ساختار فیلدهای مختلف در headerهای لایه های مختلف که هنگام رسیدن یک datagram به روتر، هدرهای آن با این ساختارها یا Match می شود یا خیر.
2. Actions: drop, modify, forward, …
3. Priority: هنگامی که یک datagram با چندین هدر Match می شود، این فیلد اولویت pattern ها را نسبت به یکدیگر مشخص می کند. (ممکن هست مثلا با دو pattern تطبیق یابد و در این حالت عمل نسبت به آن patternای که اولویت بالاتری دارد باید انجام شود)
4. برای log گیری از بسته ها استفاده می شود.

همچنین open flow امکاناتی نظیر امنیت بیشتر شبکه های default-off، شبکه های بی سیم بدون انتقال دستی، شبکه های مرکز داده مقیاس پذیر(scalable data center networks) و host mobility و... را فراهم می آورد.

بخش اول:

نیازمندی های نرم افزاری و سخت افزاری:

این بخش به ما اطلاعاتی نظیر حداقل RAM و حافظه ی مورد نیاز و سیستم عامل هایی که این پروژه روی آنها قابل پیاده سازی است را می دهد.

بخش دوم:

نصب virtual box و سایر برنامه های مورد نیاز:

چون از قبل این برنامه را در سیستم من نصب شده بود، تنها نیاز به دانلود و نصب سرور Xming و Putty terminal پیدا شد؛ که هر دو را نصب و راه اندازی کردم.

بخش سوم:

راه اندازی ماشین مجازی:

این بخش درباره کار با vBox و راه اندازی یک ماشین مجازی با import کردن فایل image موردنظر می باشد؛ که همانطور که در دستور کار ذکر شده، فایل tester.ova را import می کنیم و ماشین مجازی خود را بالا می آوریم.

یک کار دیگر نیز که در این مرحله باید انجام دهیم، باید در قسمت network، یک Ethernet interface دیگر نیز برای ماشین خود ایجاد کنیم و "Host-only adapter" را به آن attach کنیم.

با دستور ifconfig –a می توانیم interface های ماشین خود را مشاهده کنیم و اگر به آن ها آدرس ip اختصاص نیافته بود، با دستور sudo dhclient ethX که X شماره interface می باشد، می توانیم به آن آدرس ip اختصاص دهیم.

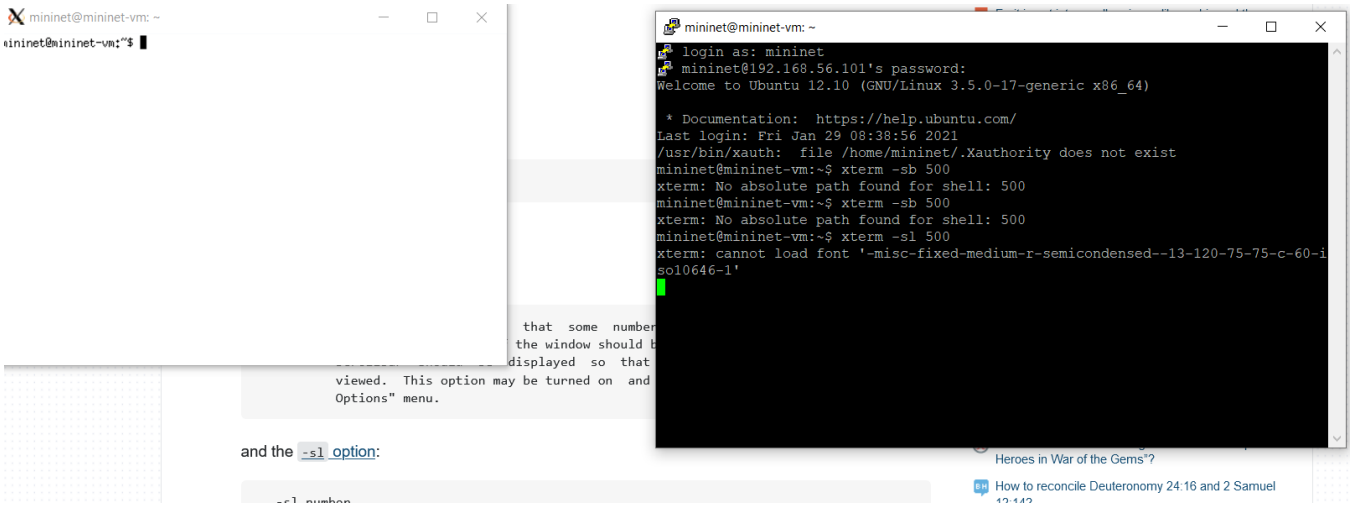
```
mininet@mininet-vm:~$ sudo dhclient eth1
mininet@mininet-vm:~$ ifconfig -a
eth0      Link encap:Ethernet  HWaddr 08:00:27:6e:79:9d
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe6e:799d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2540 (2.5 KB)  TX bytes:1956 (1.9 KB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:b2:01:1c
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feb2:11c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19 errors:0 dropped:0 overruns:0 frame:0
          TX packets:25 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3386 (3.3 KB)  TX bytes:2536 (2.5 KB)

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:16436  Metric:1
          RX packets:698 errors:0 dropped:0 overruns:0 frame:0
          TX packets:698 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:56212 (56.2 KB)  TX bytes:56212 (56.2 KB)

mininet@mininet-vm:~$
```

همانطور که مشاهده می شود، سه interface برای ما نمایش داده می شود که eth0، رابط NAT ما می باشد و آدرس 192.168.56.101 را دارد و eth1، رابط host-only ما می باشد و آدرس 10.0.2.15 را دارد.

دستیابی به VM از طریق SSH:

برای استفاده از برنامه های X11، نظیر wireshark و xterm، باید ابتدا سرور Xming اجرا شود و یک ارتباط ssh همراه با X11 forwarding، برقرار شود.

بنابراین ابتدا سرور Xming را اجرا می کنیم و سپس فایل Putty.exe را به صورت ادمین اجرا می کنیم و آدرس رابط NAT خود که در بالا ذکر شده را به آن داده و X11 forwarding را enable می کنیم و enter را می زنیم تا ترمینال mininet باز شود. پس از برقراری SSH connection، در ترمینال مربوطه دستور xterm –sl 500 را وارد می کنیم تا یک ترمینال X باز شود(به رنگ سفید)



آشنایی با ابزارهای توسعه دهنده:

توضیح مختصر درباره Mininet:

ابتدا با دستور

```
sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

یک توپولوژی با سه host و یک switch و یک controller را می سازیم. کاری که دستور فوق انجام می دهد، شامل پنج بخش می شود:

1. سه تا host می سازد که هرکدام آدرس IP جداگانه ی خود را دارند.
2. یک سوییچ نرم افزاری OpenFlow با سه پورت را در کرنل ایجاد می کند.
3. هر host مجازی را با یک کابل Ethernet مجازی به سوییچ متصل می کند.
4. آدرس MAC هر host را برابر با آدرس IP آن قرار می دهد.
5. سوییچ OpenFlow را پیکربندی می کند تا به remote controller متصل شود.

دستورات گوناگونی در Mininet وجود دارد. مانند:

```
mininet> nodes
mininet> help
mininet> h1 ifconfig
mininet> xterm h1 h2
```

```
mininet@mininet-vm: ~
login as: mininet
mininet@192.168.56.101's password:
Welcome to Ubuntu 12.10 (GNU/Linux 3.5.0-17-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Jan 29 05:51:58 2021 from 192.168.56.1
mininet@mininet-vm:~$ xterm -sl 500
xterm: cannot load font '-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-i
so10646-1'
mininet@mininet-vm:~$ sudo mn --topo single,3 --mac --switch ovsk --controller r
emote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
```

dcptl:

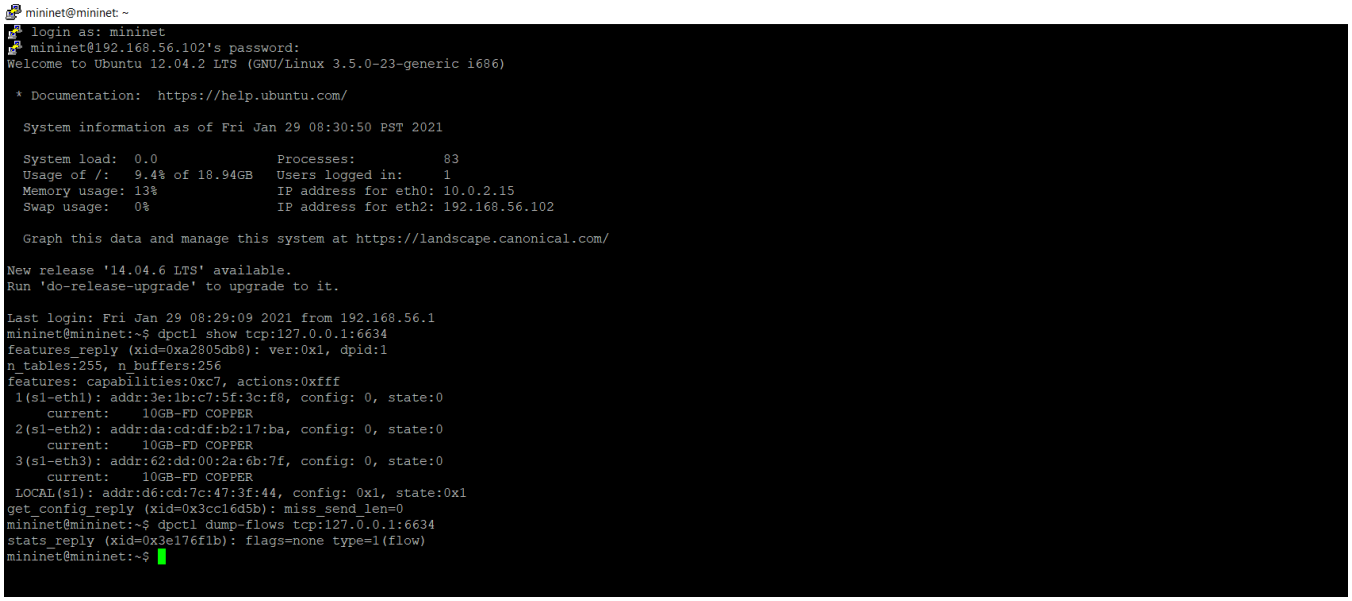
یک ابزار می باشد که همراه OpenFlow می آید و مشاهده پذیری و کنترل روی یک flow table یک سوییچ فراهم می کند و بسیار کارا برای debugging به کمک مشاهده وضعیت flowها و counterها می باشد.

حال یک پنجره ی SSH دیگر را می سازیم و در آن دستورات زیر را وارد می کنیم.

```
dpctl show tcp:127.0.0.1:6634
dpctl dump-flows tcp:127.0.0.1:6634
```

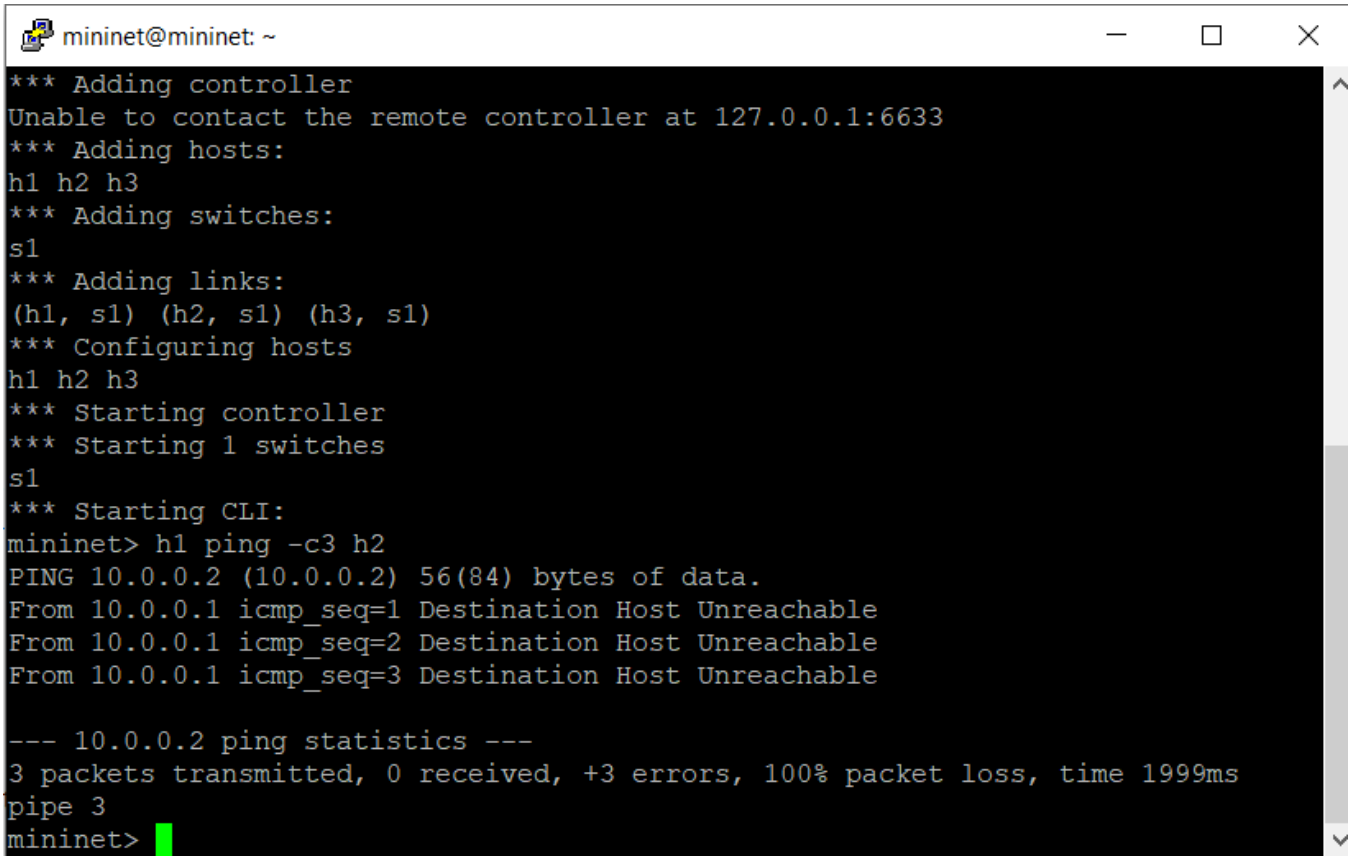
دستور show در بالا به یک سوییچ متصل می شود و پورت و گنجایش های آن سوییچ را نمایش می دهد.

همانطور که در تصویر زیر مشاهده می شود، چون هنوز ما controllerای را راه اندازی نکرده ایم، flow table خالی می باشد.



تست Ping:

اگر همین حالا از h1، h3 را ping کنیم، همانطور که در عکس زیر مشاهده می شود، همه ی packetها Lost خواهند شد چراکه هیچ flowای میان این دو host تعریف نشده است.



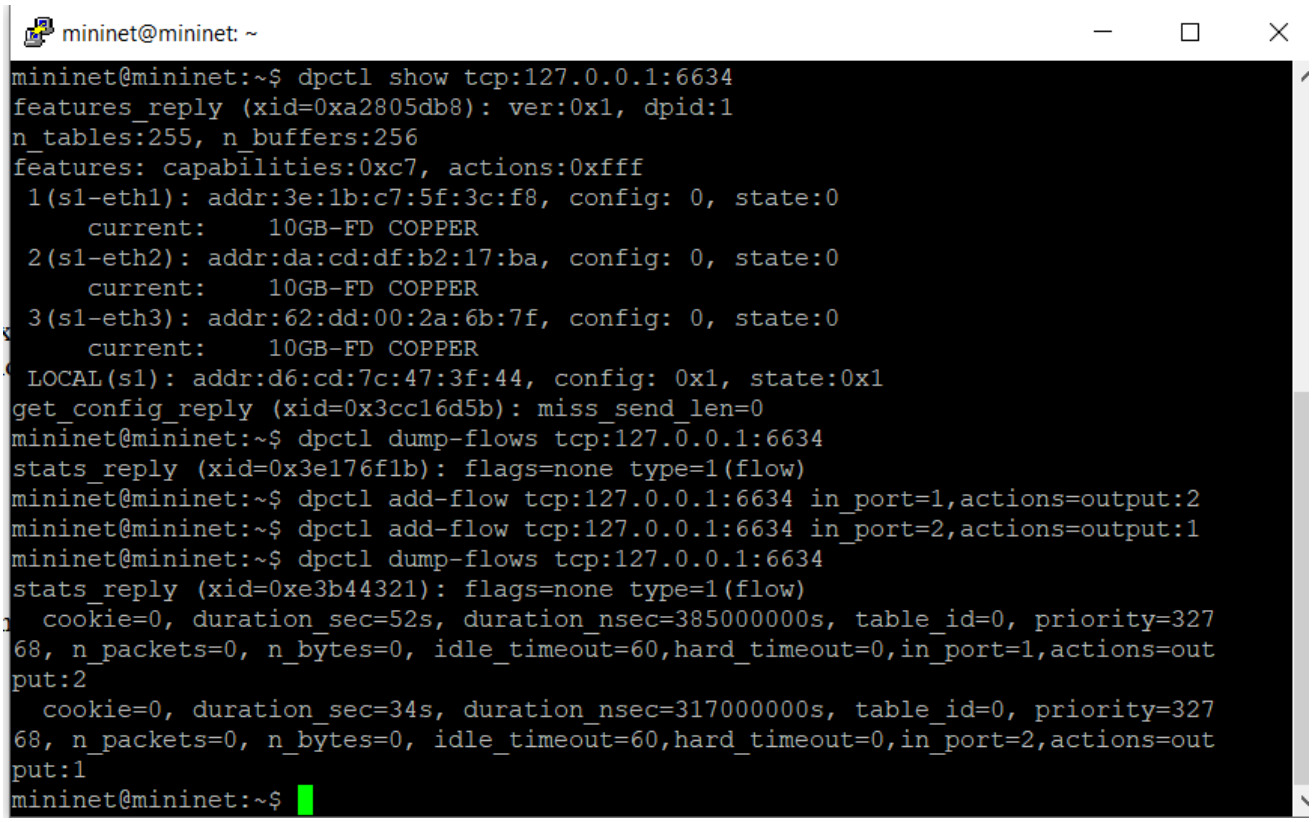
حال در ترمینال دو دستور زیر را وارد می کنیم.

```
$ dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2
$ dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1
```

حال این دو دستور باعث می شود که بسته های ورودی به پورت شماره 2 سوییچ روی پورت 1 ارسال شوند و بالعکس.

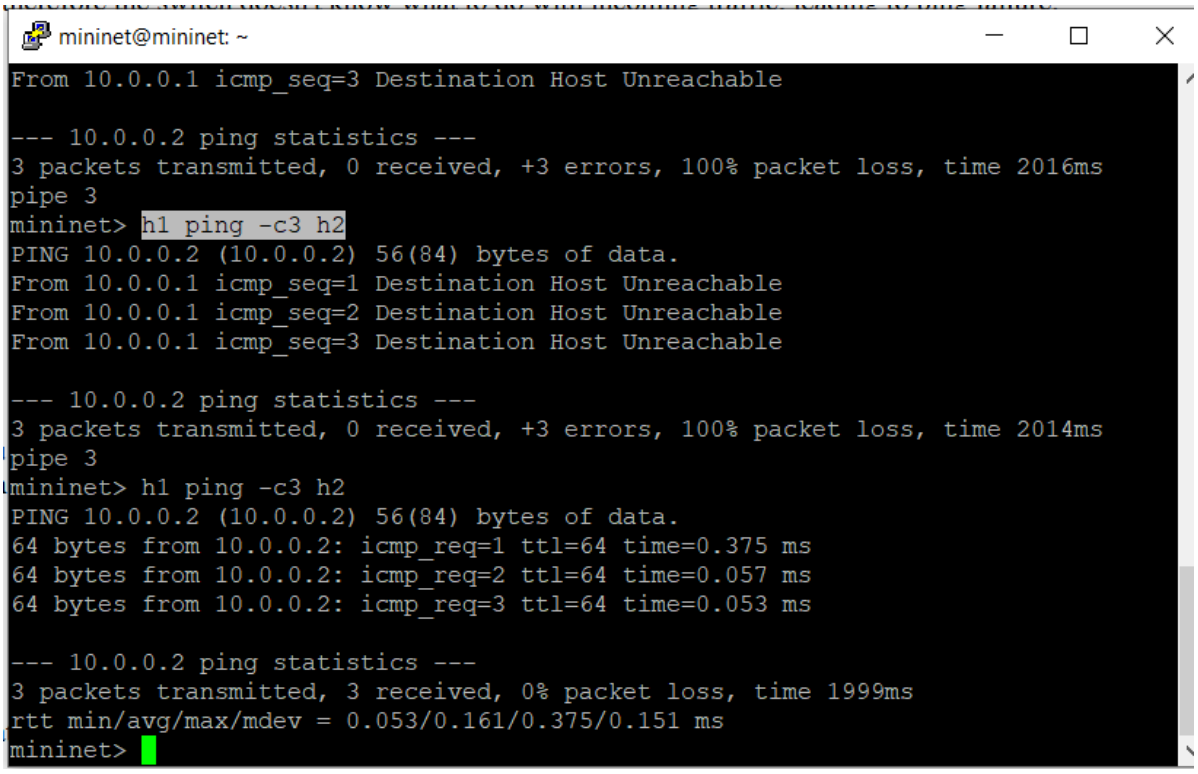
دوباره جدول flow خود را مشاهده می کنیم:

```
$ dpctl dump-flows tcp:127.0.0.1:6634
```



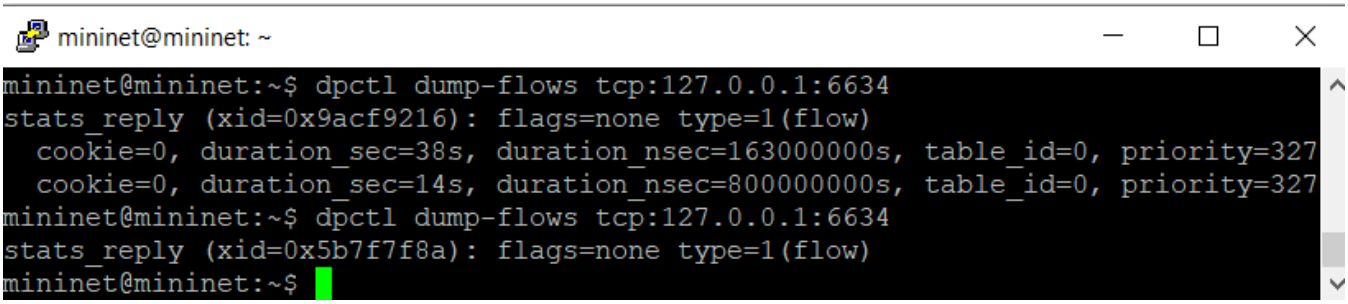
حال اگر دوباره Ping کنیم، داریم:

```
mininet> h1 ping -c3 h2
```



همانطور که در تصویر بالا مشاهده می شود به دلیل آنکه به صورت دستی روی سوییچ دو flow را تعریف کردیم، بسته ها به درستی به مقصد می رسند و reply آن ها به میزبان 1 می رسد.

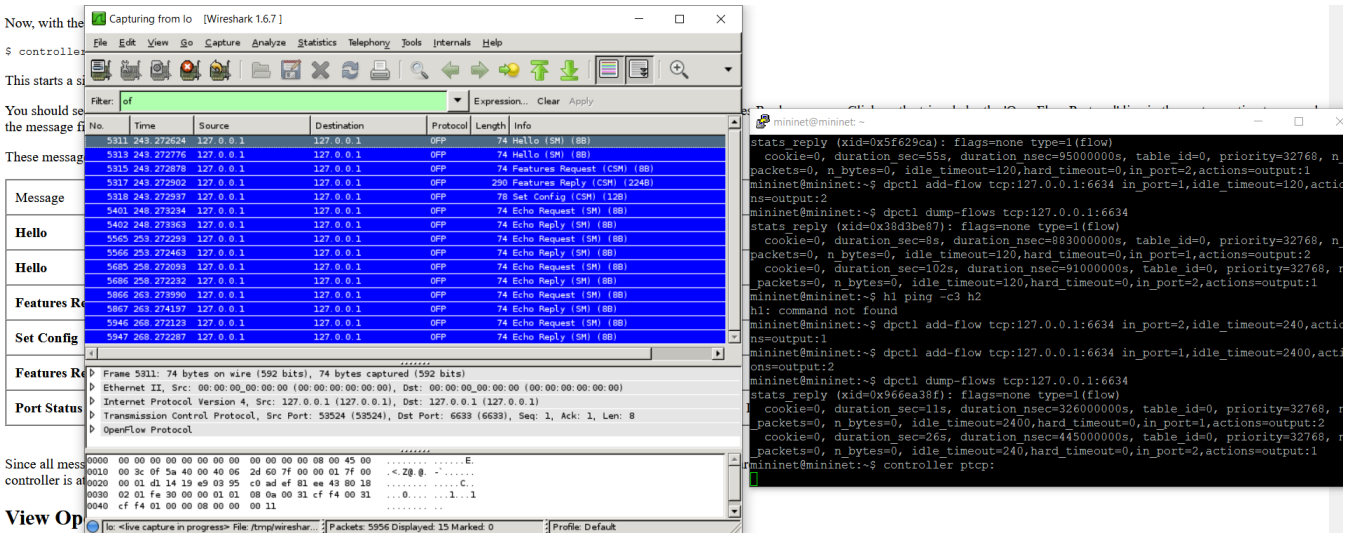
در تصویر زیر نیز اگر مشاهده کنید ابتدا flow entryها در جدول وجود دارند، اما پس از expire شدن زمان آنها از جدول حذف می شوند.



:Start Wireshark

ابتدا یک ترمینال دیگر را باز کرده و با دستور sudo wireshark &، وایرشارک را اجرا می کنیم. سپس ترافیک های کنترلی OpenFlow را با وارد کردن of در قسمت فیلتر، فیلتر می کنیم. حال درحالی که وایرشارک درحال شنود می باشد، با دستور controller ptcp، در ssh terminal، کنترلر را اجرا می کنیم.

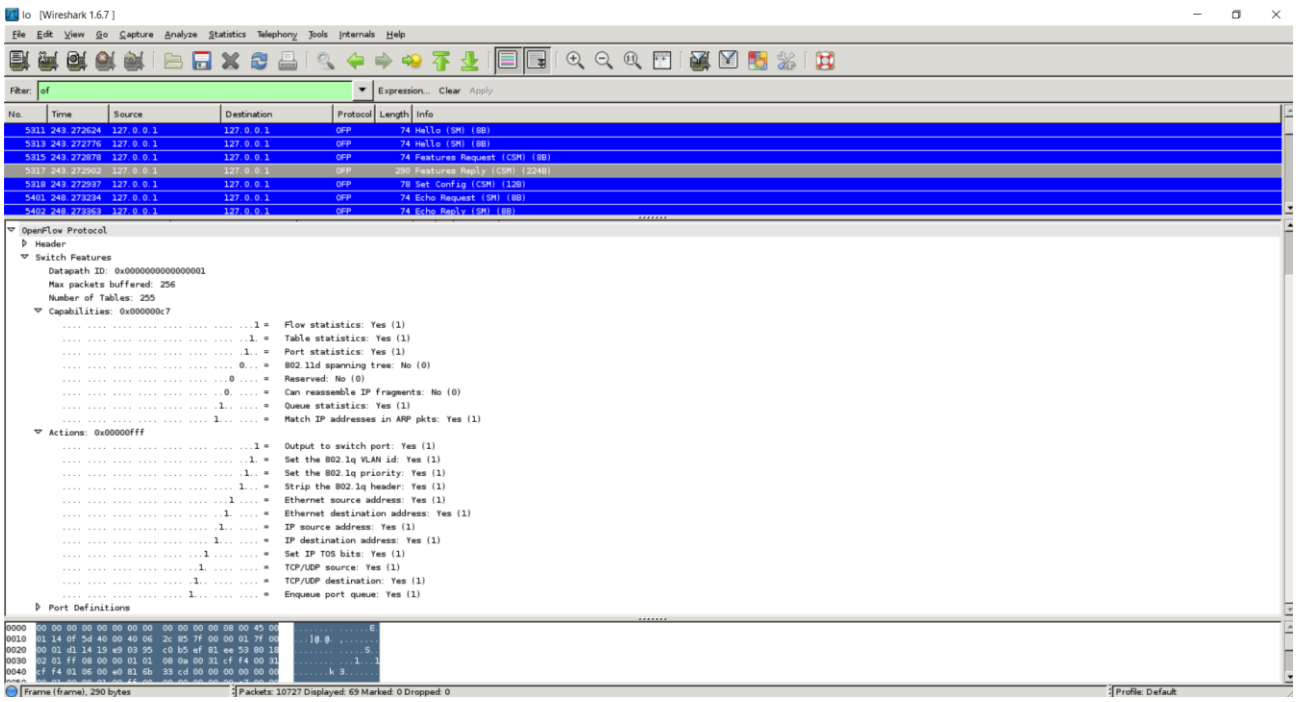
پیغام های متفاوتی را نظیر Hello message (همان tcp handshaking)، set config، Features request و... را دریافت و مشاهده می کنیم.



خلاصه ی پیام ها و توضیحات آنها را می توان در جدول زیر مشاهده کرد:

Message	Type	Description
Hello	Controller->Switch	following the TCP handshake, the controller sends its version number to the switch.
Hello	Switch->Controller	the switch replies with its supported version number.
Features Request	Controller->Switch	the controller asks to see which ports are available.
Set Config	Controller->Switch	in this case, the controller asks the switch to send flow expirations.
Features Reply	Switch->Controller	the switch replies with a list of ports, port speeds, and supported tables and actions.
Port Status	Switch->Controller	enables the switch to inform that controller of changes to port speeds or connectivity. Ignore this one, it appears to be a bug.

همانطور که در تصویر زیر نیز مشاهده می شود، با استفاده از این برنامه می توان اطلاعات بسیاری را درباره بسته ها بدست آورد. برای مثال در تصویر زیر در پیغام Features reply می توان ظرفیت های datapath را مشاهده کرد. اینکه flow تعریف شده ایستا است یا پویا، جدول سوییچ ایستا است یا خیر و پورت ایستا است یا خیر و ... .



مشاهده پیام های OpenFlow حاصل از Ping:

ابتدا برای چشم پوشی از echo-request/reply messageها، در قسمت فیلتر عبارت زیر را تایپ کرده و enter را می زنیم.

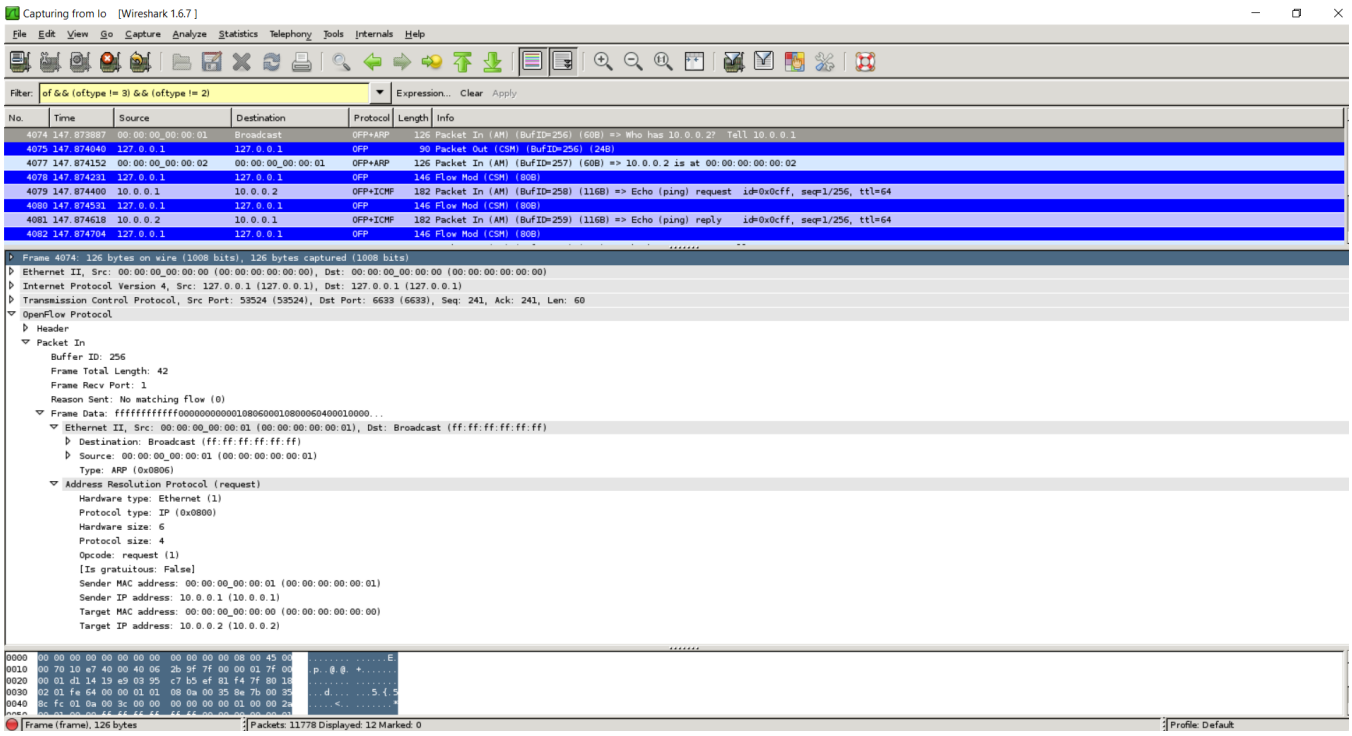
```
of && (of.type != 3) && (of.type != 2)
```

حال از h1، h2 را یکبار Ping می کنیم، پیام هایی مطابق جدول زیر را مشاهده خواهیم کرد.

Message	Type	Description
Packet-In	Switch->Controller	a packet was received and it didn't match any entry in the switch's flow table, causing the packet to be sent to the controller.
Packet-Out	Controller->Switch	controller send a packet out one or more switch ports.
Flow-Mod	Controller->Switch	instructs a switch to add a particular flow to its flow table.
Flow-Expired	Switch->Controller	a flow timed out after a period of inactivity.

در ابتدا یک پیغام ARP را مشاهده می کنیم، که در miss flow table شده و به همین دلیل broadcast شده. در گام بعدی یک پاسخ پیام ARP دریافت می شود که حالا هم آدرس MAC مبدا و هم مقصد برای کنترلر شناخته شده است.

سوییچ می تواند flowهایی را برای پیام های ICMP ایجاد کند و درخواست های ping در این حالت مستقیما از طریق dataPath منتقل می شوند و هیچ پیام اضافه را شامل نمی شوند.



Benchmark Controller w/iperf:

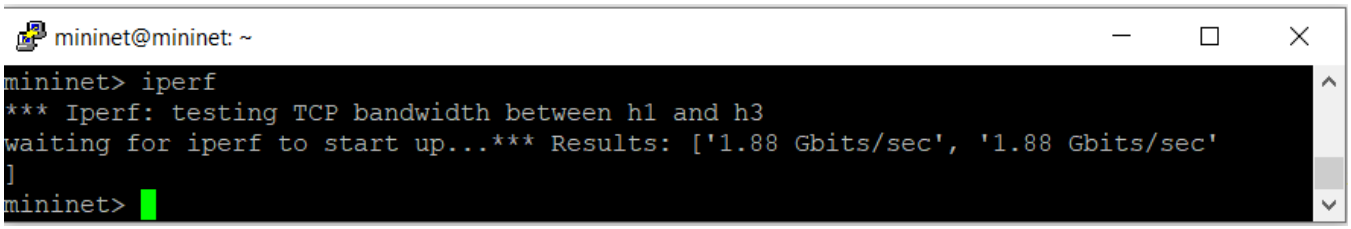
Iperf، یک ابزار cmd برای بررسی سرعت میان دو کامپیوتر می باشد.

```
mininet> iperf
```

اگر دستور بالا را برای توپولوژی حاضر و همچنین یک توپولوژی شبیه توپولوژی حاضر، منتهی با یک سوییچ فضای کاربر وارد کنیم، مشاهده خواهیم کرد که سرعت تبادل اطلاعات در سوییچ فضای کاربر بسیار پایین تر خواهد بود؛ چراکه در این سوییچ، بسته ها مدام باید میان فضای کاربر و فضای کرنل در هر گام (hop) رد و بدل شوند بجای آنکه هنگامی که از سوییچ مربوطه گذر می کنند، در همان کرنل بمانند.

سوییچ فضای کاربر برای دست کاری و تغییرات آسان تر است، اما شبیه سازی آن کند تر از سوییچ کرنل می باشد.

تصویر اول مربوط به سوییچ کرنل و تصویر دوم مربوط به سوییچ فضای کاربر می باشد.




```
mininet@mininet:~$ sudo mn --topo single,3 --controller remote --switch user
*** 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
*** Starting 1 switches
s1
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
waiting for iperf to start up...*** Results: ['478 Mbits/sec', '478 Mbits/sec']
```

بخش پنجم:

ساخت سوییچ مبتنی بر یادگیری:

در این بخش هدف برنامه کردن یک سوییچ هست که این سوییچ بررسی می کند اگر آدرس MAC مقصد بسته با پورتی تناظر داشت، از آن پورت بسته را خارج می کند ولی اگر تناظر نداشت، آن بسته را broadcast می کند.

در ادامه نیز این سوییچ را بیشتر توسعه داده و یک سوییچ برمبنای flow درواقع خواهیم داشت.

انتخاب کنترلر: POX

در ابتدا با وارد کردن ctrl+D در برنامه کنترلر یا زدن دستور sudo killall controller در ترمینال ssh، کنترلر ارجاعی را پایان می بخشیم.

سپس با دستور exit از mininet خارج شده و با دستور sudo mn –c مطمئن می شویم همه چیز پاک شده و mininet را درواقع restart می کنیم.

سپس با دستور

```
sudo mn --topo single,3 --mac --controller remote --switch ovsk
```

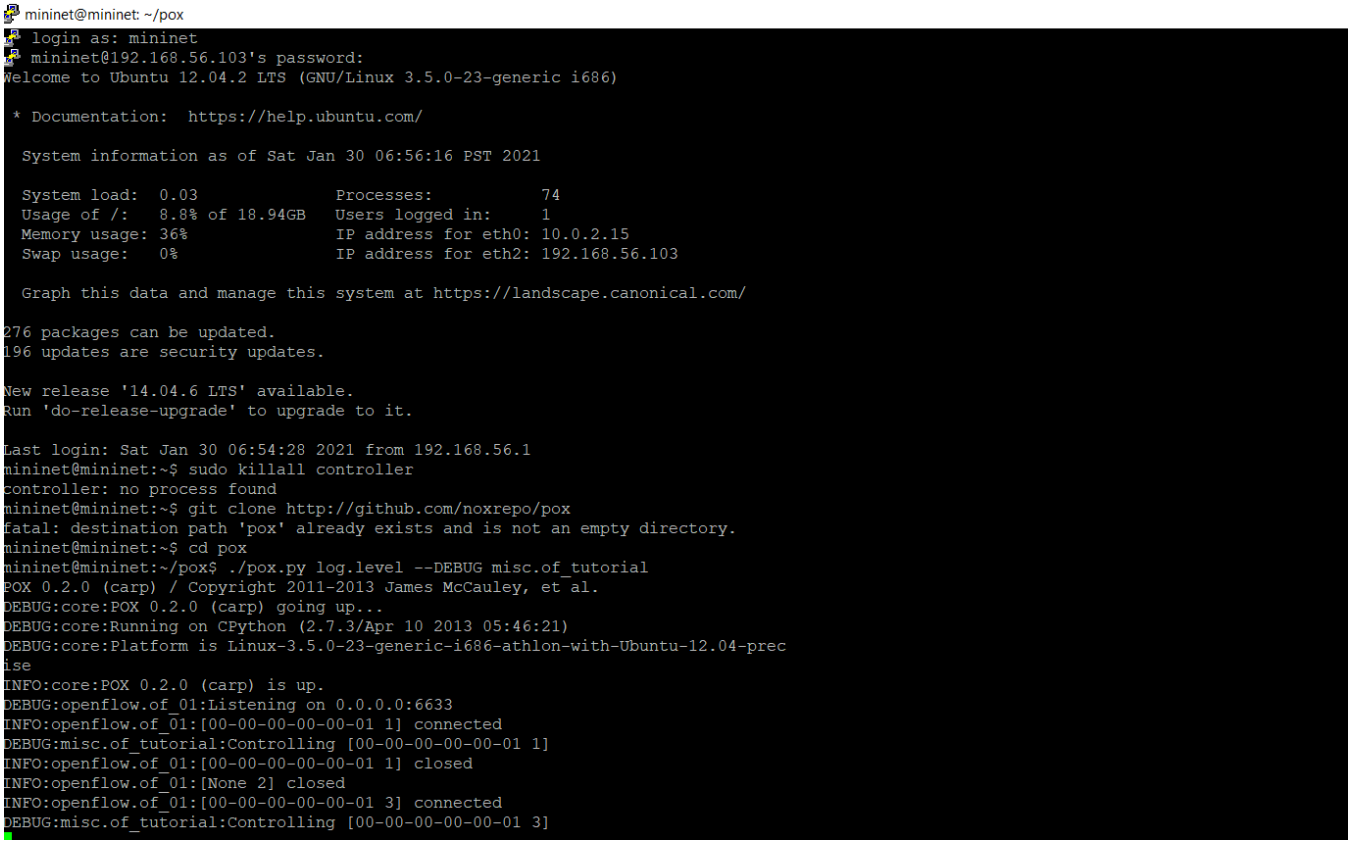
توپولوژی جدید را می سازیم.

چون mininet ما POX را به طور پیش فرض نصب شده دارد، تنها با دستور

```
./pox.py log.level --DEBUG misc.of_tutorial
```

یک hub ساده و مبنا را اجرا می کنیم.

این دستور به POX می گوید تا verbose logging را enable کند و فایل of_tutorial را که ما قرار هست آنرا توسعه دهیم را، اجرا کند.(که درحال حاضر مانند یک hub عمل می کند)



تایید رفتار hub با tcpdump:

حال برای آنکه تایید کنیم همه ی host می توانند یکدیگر را Ping کنند و پیام های یکدیگر را دریافت کنند، برای هر host ما یک xterm می سازیم و ترافیک را روی هرکدام مشاهده می کنیم.

ابتدا دستور

```
mininet> xterm h1 h2 h3
```

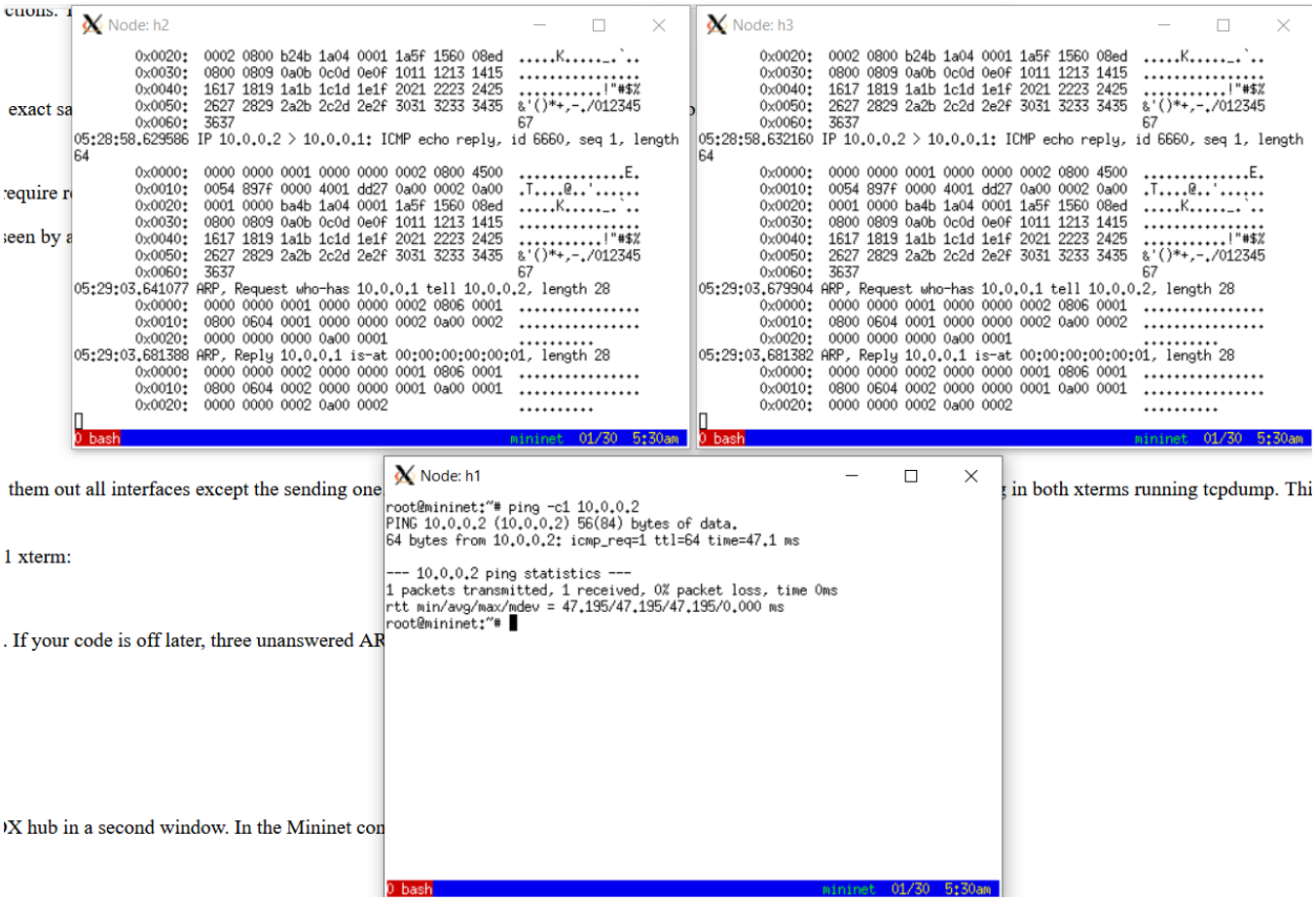
را وارد کرده و سپس در xterm مربوط به h2 و h3، دستورات زیر را به ترتیب وارد می کنیم.

```
# tcpdump -XX -n -i h2-eth0
# tcpdump -XX -n -i h3-eth0
```

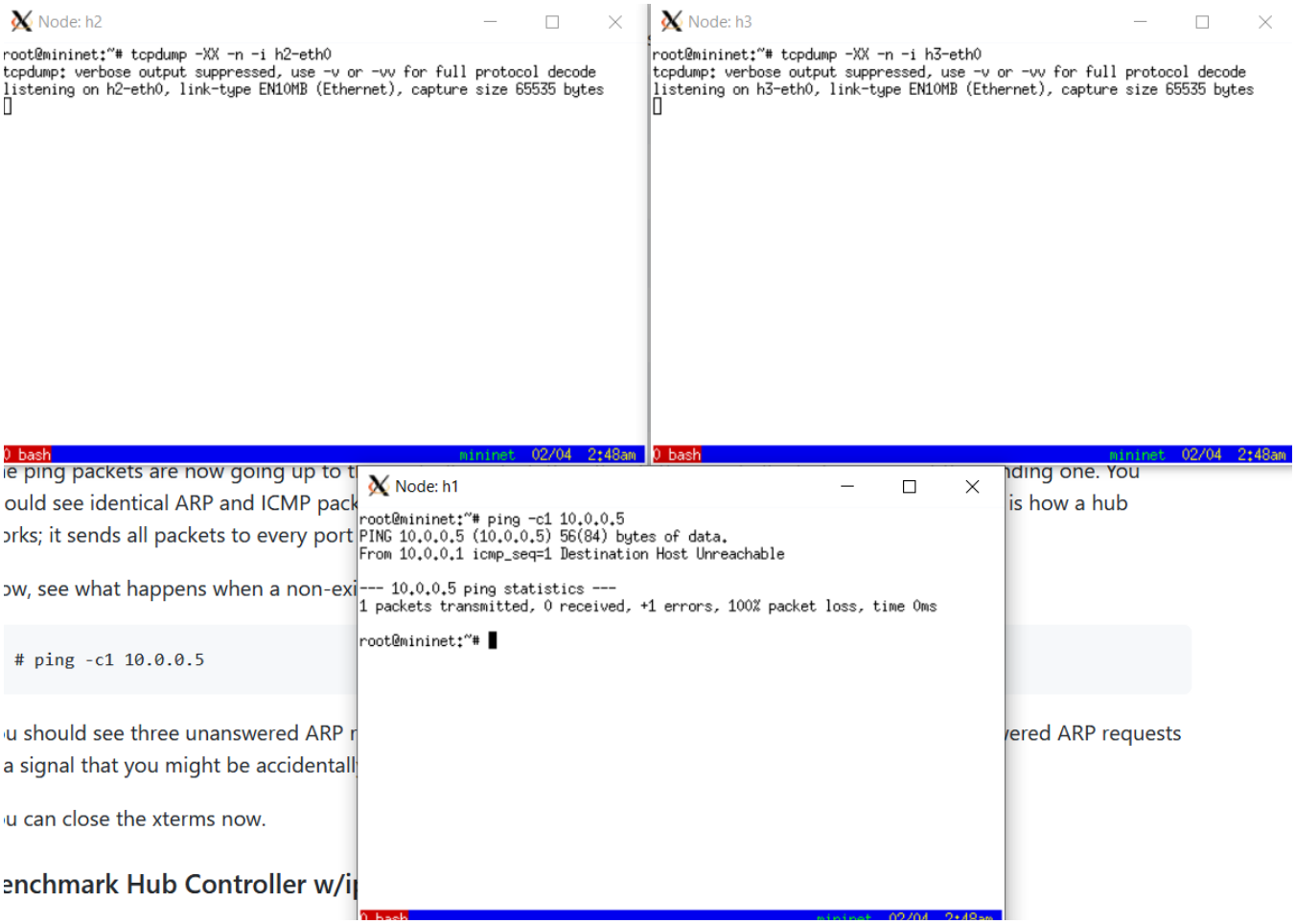
و در نهایت در xterm مربوط به h1، میزبان یا h2 را ping می کنیم با دستور:

```
# ping -c1 10.0.0.2
```

همانطور که در تصویر زیر مشاهده می شود، ping req برای هردو host دو و سه ارسال می شود. و ترافیک های دریافتی آنها دقیقا مشابه یکدیگر می باشد.

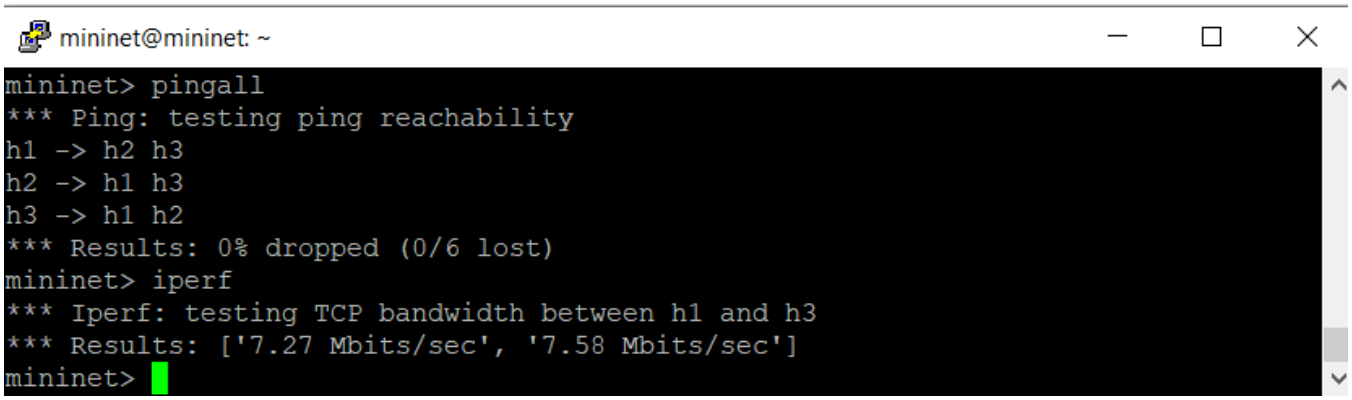


ولی اگر یک آدرس ناشناخته مانند 10.0.0.5 را ping کنیم، مشاهده خواهیم کرد که بسته ارسال lost می شود.



:Benchmark Hub Controller w/iperf

حال باید دسترسی ها را بررسی کنیم :



همانطور که مشاهده می شود، به دلیل باز تبادل پیام ها میان سوییچ و کنترلر سرعت ما کاهش یافته نسبت به حالت قبل.

بازکردن کد Hub:

ابتدا در مسیر pox/pox/misc، فایل of_tutorial.py را با دستور sudo vi of_tutorial.py باز کرده و شروع به ادیت تابع act_like_switch() می کنیم.

mininet@mininet: ~/pox/pox/misc

```
        # Note that if we didn't get a valid buffer_id, a slightly better
        # implementation would check that we got the full data before
        # sending it (len(packet_in.data) should be == packet_in.total_len)).

    def act_like_switch(self, packet, packet_in):
        """
        Implement switch-like behavior.
        """

        # Learn the port for the source MAC
        in_port = packet_in.in_port
        dl_src = str(packet.src)
        dl_dst = str(packet.dst)
        self.mac_to_port[dl_src] = in_port

        if dl_dst in self.mac_to_port:
            # Send packet out the associated port
            dst_in_port = self.mac_to_port[dl_dst]
            self.resend_packet(packet_in, dst_in_port)

            # Once you have the above working, try pushing a flow entry
            # instead of resending the packet (comment out the above and
            # uncomment and complete the below.)

            log.debug(
                "Installing flow for mac address {} on port {}. Destination mac is {}".format(dl_src, in_port, dl_dst))
            # Maybe the log statement should have source/destination/port?

            msg = of.ofp_flow_mod()

            # Set fields to match received packet
            # msg.match = of.ofp_match.from_packet(packet)
            msg.match = of.ofp_match.from_packet(packet)

            # < Set other fields of flow_mod (timeouts? buffer_id?) >

            # < Add an output action, and send -- similar to resend_packet() >
            action = of.ofp_action_output(port=dst_in_port)
            msg.actions.append(action)

            # log.debug(msg)
            self.connection.send(msg)

        else:
            # Flood the packet out everything but the input port
            # This part looks familiar, right?
            self.resend_packet(packet_in, of.OFPP_ALL)
```

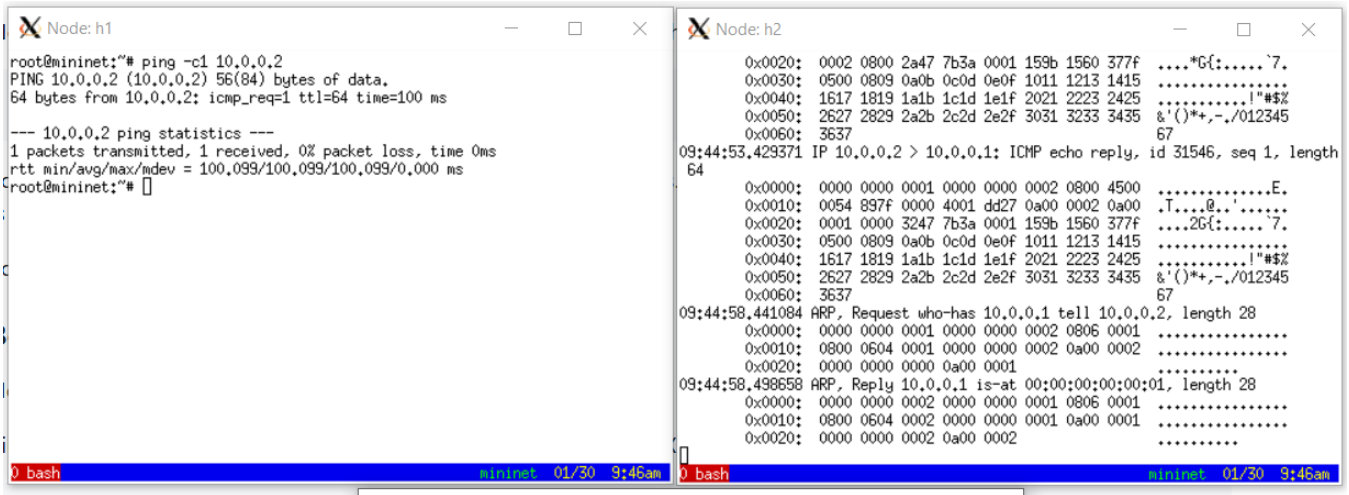
در این قسمت همانطور که مشاهده می شود، ابتدا ما پورت ورودی یک بسته را استخراج می کنیم و سپس آدرس Mac مبدا بسته را در dl_src و آدرس مقصد آن را در dl_dst می ریزیم و در نهایت این جفت را به عنوان یک درایه ی جدید به دیکشنری mac_to_port اضافه می کنیم.

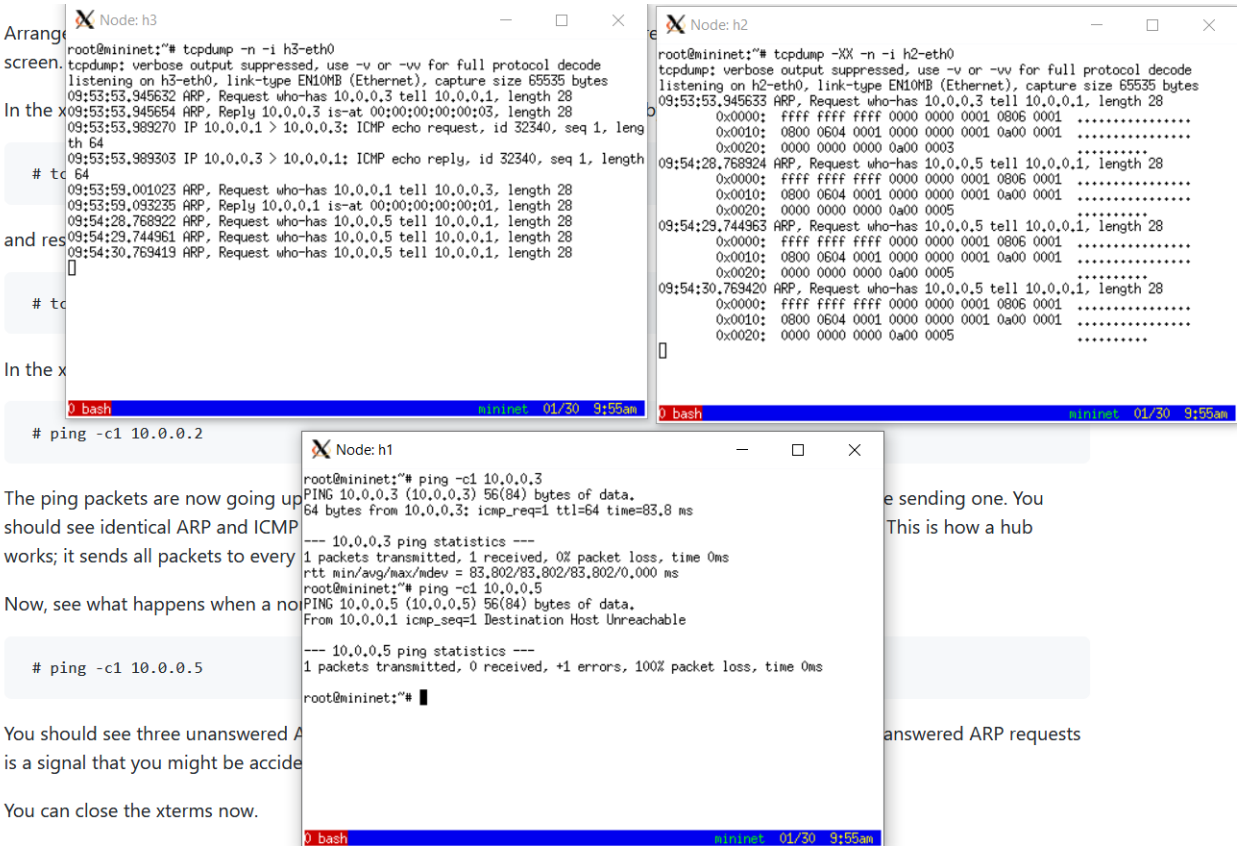
اگر آدرس مقصد بسته در دیکشنری ما موجود بود، پورت مقصد به راحتی همان پورت متناظر با آدرس مقصد خواهد شد و بدست می آید و درنهایت با فراخوانی تابع resend و دادن مقادیر پورت خروجی و بسته ی ورودی، بسته از پورت مناسب به سمت مقصد ارسال خواهد شد. در غیر اینصورت broadcast خواهد شد.

حال می توانیم به ازای هر بسته با مبدا و مقصد مشخص، یک flow entry به جدول flow خود اضافه کنیم تا دیگر مجبور نباشیم تمامی پیام ها را ابتدا به کنترلر بفرستیم و این کار پهنای باند ما را بسیار افزایش خواهد داد.

تست کنترلر:

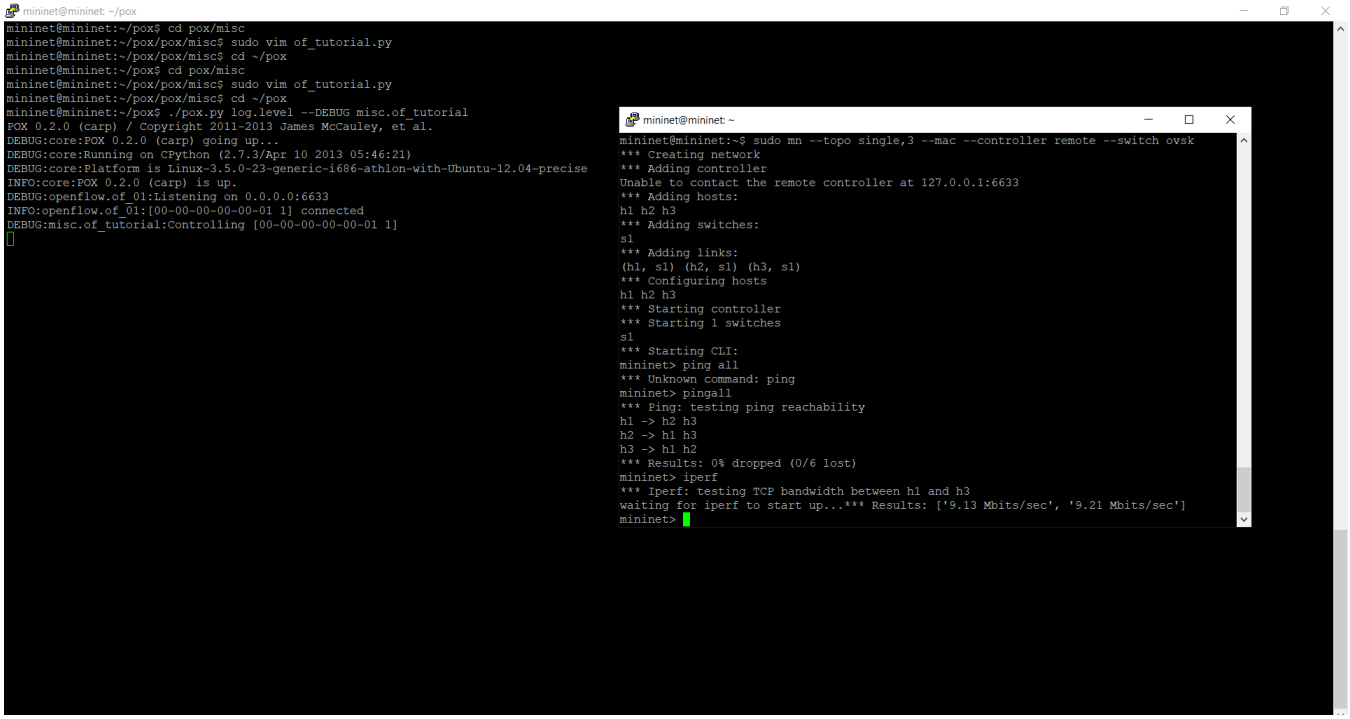
اکنون می توانیم با بهره گیری از POX و ادیتور Vim، کد of_tutorial را تغییر داده و یک learning switch بسازیم؛ و در تصاویر زیر تست های انجام شده روی این learning switch را مشاهده می کنید.





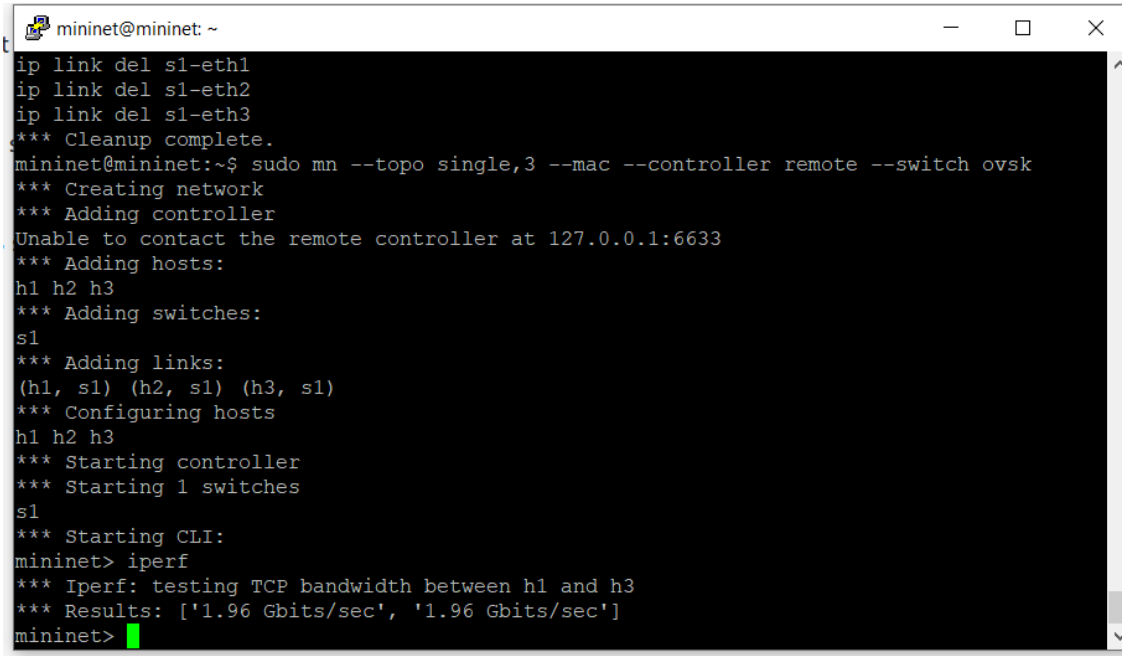
همانطور که در تصاویر بالا مشاهده می شود، دیگر پیام های مربوط به یک host خاص از تمامی پورت ها broadcast نمی شود و همینطور سه تا ARP Reply مربوط به ping آدرس ناشناخته ی 10.0.0.5 نیز کاملا قابل مشاهده است.

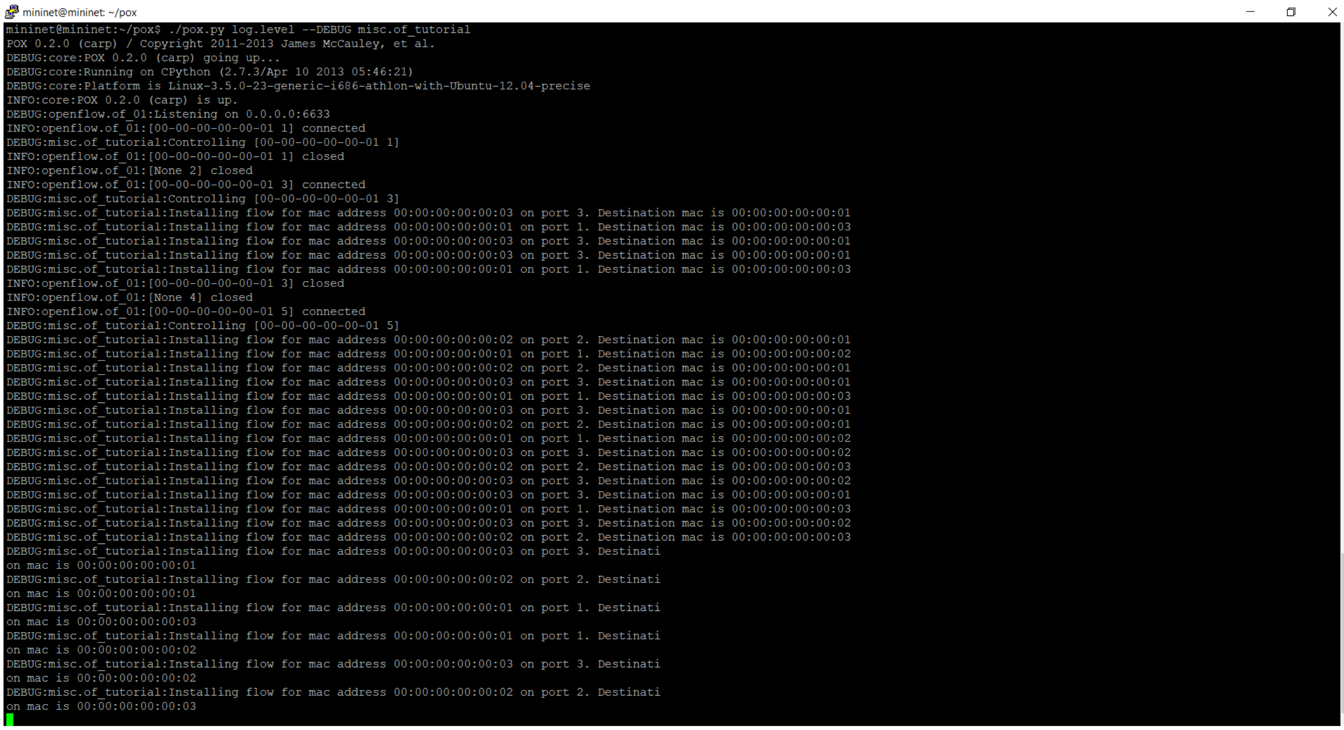
برای pingall و iperf نیز داریم:



حال یک مرحله دیگر نیز پیش رفته و flow entry ها را ایجاد می کنیم و یک سوییچ برمبنای flow را می سازیم.
همانطور که مشاهده می شود نرخ انتقال و پهنای باند بسیار افزایش می یابد و شبیه حالتی می شود که ما از یک کرنل سوییچ ارجاعی در توپولوژی های قدیمی تر استفاده می کردیم.

مطالب ذکر شده ی فوق در دو تصویر زیر قابل تصدیق است.





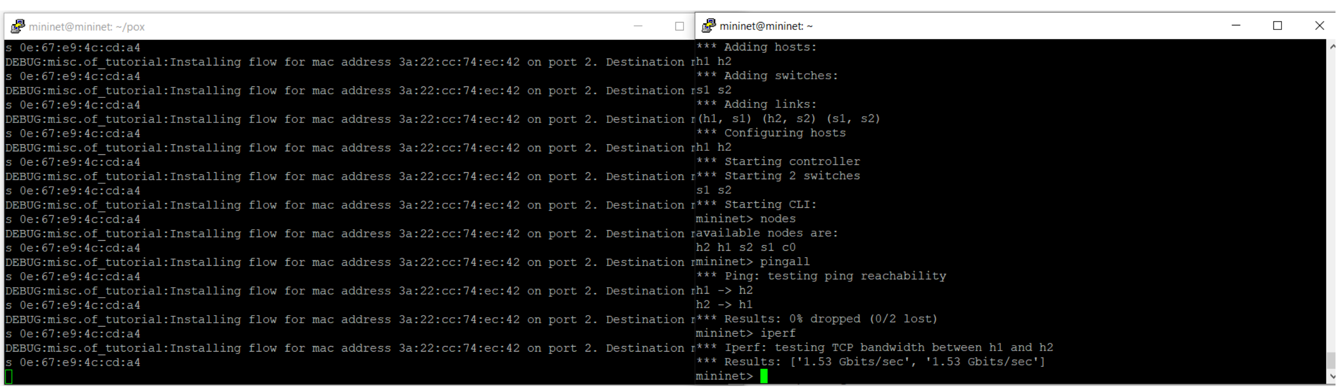
پشتیبانی از چند سوییچ:

ابتدا با دستور

$ sudo mn --topo linear --switch ovsk --controller remote

توپولوژی ای با دو host و دو switch می سازیم؛ که هر host به یک سوییچ متصل است و سوییچ ها نیز به یکدیگر متصل خواهند بود.

کنترلر ما این تست را نیز همانطور که در تصویر زیر مشاهده می شود، پشت سر می گذارد.



بخش هشتم:

ساخت firewall:

در این قسمت مانند قسمت پنجم ابتدا یک فایل مانند firewall.py را در مسیر pox/pox/misc/. ایجاد کرده و کد مورد نظر خود را در آن پیاده می کنیم.

قسمتی از کد در تصویر زیر آمده است:

حال مانند دو بخش قبل در دایرکتوری pox/. دستور

```
./pox.py log.level --DEBUG misc.firewall
```

را وارد می کنیم و در ترمینال mininet، دستور

```
$ sudo mn --topo single,3 --mac --controller remote --switch ovsk
```

را وارد می کنیم.

سپس با دستور xterm h1 h2، xterm مربوط به این دو host را بالا می آوریم.

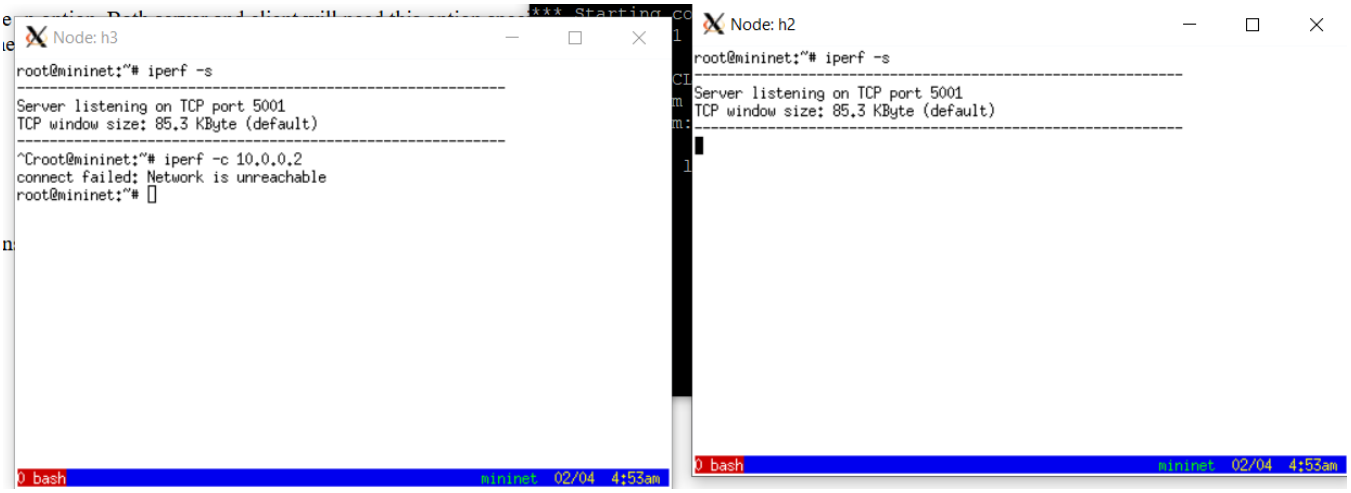
و دستور

```
$ iperf -s
```

را در h2 وارد می کنیم تا به عنوان یک سرور عمل کند و دستور

```
$ iperf -c 10.0.0.2
```

در xterm h3 وارد می کنیم و همانطور که در تصویر زیر مشاهده می کنید، firewall پیام های این host یعنی h3 را فیلتر می کند.



باسپاس از توجه شما