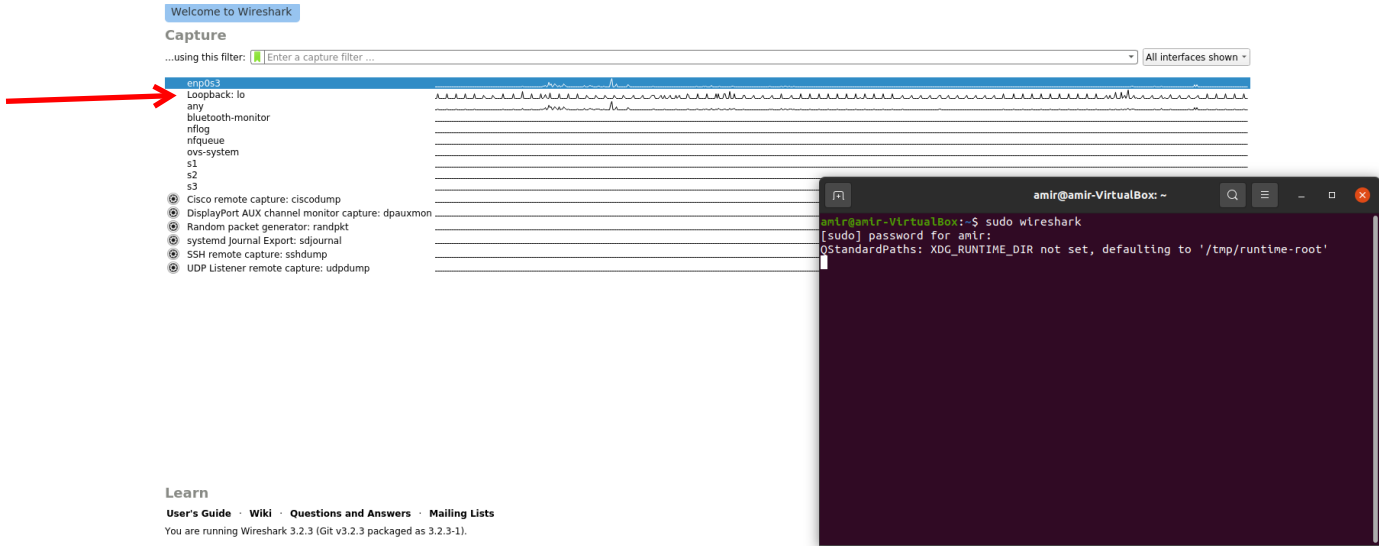


به نام خدا

امیررضا حسینی - ۹۸۲۰۳۶۳

شبکه ۲ - پروژه دوم

سؤال اول)



Welcome to Wireshark

Capture

...using this filter: All interfaces shown

enp0s3

Loopback: lo

any

bluetooth-monitor

nflog

nfqueue

ovs-system

s1

s2

s3

⊙ Cisco remote capture: ciscodump

⊙ DisplayPort AUX channel monitor capture: dpauxmon

⊙ Random packet generator: rndpkt

⊙ systemd Journal Export: systemd

⊙ SSH remote capture: sshdump

⊙ UDP Listener remote capture: udpdump

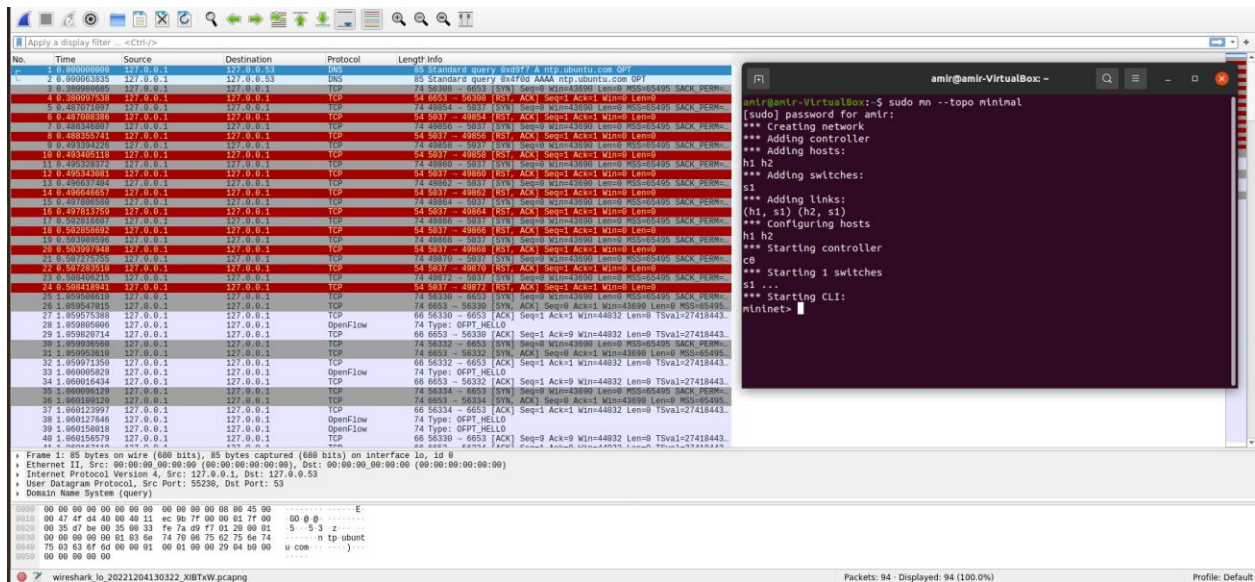
Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).

```
amir@amir-VirtualBox: ~  
amir@amir-VirtualBox:~$ sudo wireshark  
[sudo] password for amir:  
StandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

۱-۱) از زمان ایجاد توپولوژی پیغام‌های ردوبدل شده تنها از پروتکل TCP و OpenFlow استفاده کرده‌اند و همان‌طور که می‌دانیم پروتکل OpenFlow نیز خودش از TCP استفاده می‌کند.



Apply a display filter:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	DNS	80	Standard query 64977 & ntp.ubuntu.com OPT
2	0.000000000	127.0.0.1	127.0.0.1	DNS	80	Standard query 64977 AAAA ntp.ubuntu.com OPT
3	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
4	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
5	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
6	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
7	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
8	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
9	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
10	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
11	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
12	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
13	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
14	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
15	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
16	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
17	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
18	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
19	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
20	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
21	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
22	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
23	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
24	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
25	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
26	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
27	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
28	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
29	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
30	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
31	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
32	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
33	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
34	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
35	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
36	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
37	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
38	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0
39	0.000000000	127.0.0.1	127.0.0.1	TCP	74	65330 → 65332 [SYN] Seq=0 Win=0 Len=0 MSS=65495
40	0.000000000	127.0.0.1	127.0.0.1	TCP	64	65332 → 65330 [ACK] Seq=1 Ack=1 Win=0 Len=0

Frame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface lo, id 0

Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

User Datagram Protocol, Src Port: 55238, Dst Port: 53

Domin Name System (query)

```
amir@amir-VirtualBox: ~  
amir@amir-VirtualBox:~$ sudo mn --topo mininet  
[sudo] password for amir:  
*** 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
```

۲-۱) در قسمت info پیام‌های hello مشخص شده‌اند. به‌صورت OFTP_HELLO

The image displays a Wireshark packet capture of an OFTP network setup. The packet list on the left shows several OFTP_HELLO messages (e.g., 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65). The packet details pane on the right shows the structure of these messages, including fields like 'Seq', 'Ack', 'Win', 'Len', 'Type', and 'Features'. A red arrow points to the 'Seq' field in the 'OFTP_HELLO' packet details. On the right, a terminal window shows the configuration of a network topology using 'mininet', including creating a network, adding a controller, adding hosts (h1, h2), adding switches (s1), adding links, and starting the controller and switches.

۳-۱) روند آن بدین صورت است که ابتدا کنترلر برای بدست‌آوردن Data path ID سوئیچ در قالب پیغام feature request feature reply را ارسال می‌کند و در پاسخ نیز سوئیچ پیغام capabilities (در پیغام اول پورت هم از طرف کنترلر برای سوئیچ ارسال می‌شود).

The image displays a Wireshark packet capture showing the exchange of feature request and feature reply messages between a controller and switches. The packet list on the left shows several 'feature request' and 'feature reply' messages (e.g., 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100). The packet details pane on the right shows the structure of these messages, including fields like 'Seq', 'Ack', 'Win', 'Len', 'Type', and 'Features'. A red arrow points to the 'Seq' field in the 'feature request' packet details. On the right, a terminal window shows the configuration of a network topology using 'mininet', including creating a network, adding a controller, adding hosts (h1, h2), adding switches (s1), adding links, and starting the controller and switches.

دو پیغام feature request و feature reply به‌عنوان نمونه در تصویر فوق مشخص شده‌اند.

(۴-۱)

No.	Time	Source	Destination	Protocol	Length	Info
37	2.036429993	127.0.0.1	127.0.0.1	TCP	74	46794 → 5937 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 ...
38	2.036438661	127.0.0.1	127.0.0.1	TCP	54	5937 → 46794 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
39	2.036403094	127.0.0.1	127.0.0.1	TCP	74	46796 → 5937 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 ...
40	2.036410856	127.0.0.1	127.0.0.1	TCP	54	5937 → 46796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
41	2.040411650	127.0.0.1	127.0.0.1	TCP	74	46798 → 5937 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 ...
42	2.040472287	127.0.0.1	127.0.0.1	TCP	54	5937 → 46798 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
43	2.154374027	::	ff02::1:ffea:f65d	OpenFlow	162	Type: OFPT_PACKET_IN
44	2.156931897	127.0.0.1	127.0.0.1	OpenFlow	90	Type: OFPT_PACKET_OUT
45	2.156945653	127.0.0.1	127.0.0.1	TCP	66	56894 → 6653 [ACK] Seq=561 Ack=101 Win=44032 Len=0 TSval=373127...
46	2.624509908	::	ff02::1:ffd0:bfd9	OpenFlow	162	Type: OFPT_PACKET_IN
47	2.624775890	127.0.0.1	127.0.0.1	OpenFlow	90	Type: OFPT_PACKET_OUT
48	2.624805372	127.0.0.1	127.0.0.1	TCP	66	56894 → 6653 [ACK] Seq=657 Ack=125 Win=44032 Len=0 TSval=373244...
49	3.159808334	fe80::dc61:aaff:feea:f65d	ff02::16	OpenFlow	174	Type: OFPT_PACKET_IN
50	3.159412233	fe80::dc61:aaff:feea:f65d	ff02::2	OpenFlow	154	Type: OFPT_PACKET_IN
51	3.159622362	127.0.0.1	127.0.0.1	OpenFlow	90	Type: OFPT_PACKET_OUT
52	3.159643385	127.0.0.1	127.0.0.1	TCP	66	56894 → 6653 [ACK] Seq=853 Ack=149 Win=44032 Len=0 TSval=373378...
53	3.159677841	127.0.0.1	127.0.0.1	OpenFlow	90	Type: OFPT_PACKET_OUT
54	3.159681935	127.0.0.1	127.0.0.1	TCP	66	56894 → 6653 [ACK] Seq=853 Ack=173 Win=44032 Len=0 TSval=373378...
55	3.191571992	fe80::dc61:aaff:feea:f65d	ff02::16	OpenFlow	174	Type: OFPT_PACKET_IN
56	3.191713740	127.0.0.1	127.0.0.1	OpenFlow	90	Type: OFPT_PACKET_OUT
57	3.231245479	127.0.0.1	127.0.0.1	TCP	66	56894 → 6653 [ACK] Seq=961 Ack=197 Win=44032 Len=0 TSval=373396...
58	3.628711400	fe80::287e:4fff:fed0:bfd9	ff02::16	OpenFlow	174	Type: OFPT_PACKET_IN
59	3.628729654	fe80::287e:4fff:fed0:bfd9	ff02::2	OpenFlow	154	Type: OFPT_PACKET_IN
60	3.628900435	127.0.0.1	127.0.0.1	OpenFlow	90	Type: OFPT_PACKET_OUT

در اینجا نمونه‌ای از این پکت‌ها را مشاهده میکنیم که در قسمت info مشخص شده اند.

(۵-۱)

این بسته در هنگام reverse connection یا missing fellow control ارسال می‌شود.

در این پیغام‌ها عملاً کنترل به کنترلر سپرده می‌شود که یا در قسمت action این مورد ذکر می‌شود و یا اینکه هیچ match ای برای بسته وجود نداشته باشد. به عبارت دیگر، برای همه بسته‌هایی که ورودی جریان منطبق ندارند یا اگر بسته‌ای با یک ورودی با عمل ارسال به کنترل‌کننده مطابقت داشته باشد، یک پیام packet-in برای کنترل‌کننده ارسال می‌شود. اگر سوئیچ حافظه کافی برای بافر کردن بسته‌هایی که به کنترل‌کننده ارسال می‌شوند، داشته باشد، پیام packet-in حاوی بخشی از هدر بسته (به طور پیش‌فرض، 128 بایت) و یک شناسه بافر است که می‌تواند توسط کنترل‌کننده در زمان آماده‌سازی استفاده شود.

سوئیچ برای ارسال بسته سوئیچ‌هایی که از بافر داخلی پشتیبانی نمی‌کنند (یا فضای بافر داخلی آنها تمام شده است) باید بسته کامل را به عنوان بخشی از پیام به کنترل‌کننده ارسال کنند. در OpenFlow SDN، هنگامی که یک سوئیچ بسته‌ای را در یک پورت دریافت می‌کند، سعی می‌کند بسته را با یک ورودی جریان (Entry) در جدول جریان پیش‌فرض سوئیچ مطابقت دهد. اگر سوئیچ نتواند جریانی را که با بسته مطابقت دارد پیدا کند، به طور پیش‌فرض بسته را به عنوان یک بسته ورودی برای بررسی و پردازش دقیق‌تر به کنترل‌کننده ارسال می‌کند. کنترل‌کننده‌ها این پیام‌ها را برای تغییر وضعیت داخلی کنترل‌کننده‌ها و یا راه‌اندازی درج ورودیهای جریان و ارسال بسته‌ها به سوئیچ‌های دیگر پردازش می‌کنند.

۶-۱) بسته‌های ICMP تحت پروتکل OpenFlow از هاست ۱ با آیدی 10.0.0.1 به هاست ۲ با آیدی 10.0.0.2 فرستاده شده اند و در جواب نیز هاست ۲ برای هاست یک پاسخ را ارسال کرده است.

icmp							*** Starting 1 switches	
No.	Time	Source	Destination	Protocol	Length	Info	s1 ...	
0	0.805607285	10.0.0.1	10.0.0.2	OpenFlow	182	Type: OFPT_PACKET_IN	*** Starting CLI:	
11	0.806540565	10.0.0.2	10.0.0.1	OpenFlow	182	Type: OFPT_PACKET_IN	mininet> h1 ping h2	

در این قسمت نیز از h2 به h1 پینگ کردیم.

icmp							rtt min/avg/max/mdev = 0.038/0.199/1.978/0.468 ms	
No.	Time	Source	Destination	Protocol	Length	Info	mininet> h2 ping h1	
1	1.83808559	10.0.0.2	10.0.0.1	OpenFlow	182	Type: OFPT_PACKET_IN	PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.	
7	1.828264391	10.0.0.1	10.0.0.2	OpenFlow	182	Type: OFPT_PACKET_IN	64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=6.92 ms	
10	2.839041144	10.0.0.2	10.0.0.1	OpenFlow	182	Type: OFPT_PACKET_IN	64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.22 ms	

در این بسته محتوای قسمت OpenFlow > Internet Protocol Version 4 اشاره شده است که این بسته محتوی بسته‌ای با پروتکل icmp است و از آنجاکه جریان مناسبی برای آن یافته نشده که از طریق روترها هدایت شود با بسته OFPT_PACKET_IN به کنترلر ارسال شده است. همان‌طور که می‌دانیم بسته OPFT_PACKET_IN محتوی بسته‌ای است که نتوانستیم جریان مناسب را برای آن بیابیم.

Urgent pointer: 0	
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps	
[SEQ/ACK analysis]	
[Timestamps]	
TCP payload (116 bytes)	
[PDU Size: 116]	
▼ OpenFlow 1.0	
.000 0001 = Version: 1.0 (0x01)	
Type: OFPT_PACKET_IN (10)	
Length: 116	
Transaction ID: 0	
Buffer Id: 0x00000114	
Total length: 98	
In port: 2	
Reason: No matching flow (table-miss flow entry) (0)	
Pad: 00	
▶ Ethernet II, Src: 2a:7e:4f:d0:bf:d9 (2a:7e:4f:d0:bf:d9), Dst: de:61:aa:ea:f6:5d (de:61:aa:ea:f6:5d)	
▼ Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.1	
0100 = Version: 4	
.... 0101 = Header Length: 20 bytes (5)	
▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)	
Total Length: 84	
Identification: 0xe0c3 (57539)	
Flags: 0x4000, Don't fragment	
Time to live: 64	
Protocol: ICMP (1)	
Header checksum: 0x45e3 [validation disabled]	
[Header checksum status: Unverified]	
Source: 10.0.0.2	
Destination: 10.0.0.1	
▶ Internet Control Message Protocol	

سؤال دوم) از این دستور برای تست پهنای باند بین هاست ها استفاده می شود. به عنوان مثال من از Minimal topology استفاده کرده ام و در این سناریو دو هاست داریم، بنابراین پهنای باند را بین این دو هاست بررسی کرده است.

همین طور که در تصویر نیز مشخص است برای تست پهنای باند از کانکشن TCP استفاده کرده است. دستور iperf یک ابزار تست عملکرد شبکه است.

Iperf می تواند کیفیت پهنای باند TCP و UDP را آزمایش کند.

Iperf می تواند حداکثر پهنای باند TCP را با پارامترهای مختلف و ویژگی های UDP اندازه گیری کند.

iperf می تواند پهنای باند را گزارش کند، jitter را به تأخیر بیندازد و بسته را از دست بدهد. با استفاده از ویژگی iperf می توان از آن برای تست عملکرد برخی از دستگاه های شبکه مانند روتر، فایروال، سوئیچ و... استفاده کرد.

```
amir@amir-VirtualBox:~$ sudo mn --topo minimal
*** 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> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
f*** Results: ['63.7 Gbits/sec', '63.8 Gbits/sec']
mininet> h1 iperf -s -p 5566 -i &
mininet> h2 iperf -c 10.0.0.1 -p 5566 -t 15
-----
Client connecting to 10.0.0.1, TCP port 5566
TCP window size: 85.3 KByte (default)
-----
[ 3] local 10.0.0.2 port 55148 connected with 10.0.0.1 port 5566
[ ID] Interval      Transfer    Bandwidth
[ 3]  0.0-15.0 sec  112 GBytes  64.0 Gbits/sec
mininet>
```

Simple iperf TCP test between two (optionally specified) hosts Usage:
iperf node1 node2