پروژه ۲ شبکه ۲ _ سارا سلطانی _ ۹۸۲۳۰۳۳

سوال ۱: وایرشارک به مدیران شبکه این امکان را می‌دهد که ترافیک شبکه را به صورت بلادرنگ مانیتور کنند و به محض وقوع هر گونه مشکل یا تهدید، اقدامات لازم را انجام دهند. همچنین وایرشارک می‌تواند بسته‌های شبکه را ضبط کرده و آنها را تجزیه و تحلیل کند .این ابزار می‌تواند اطلاعات دقیقی از هر بسته شامل منبع، مقصد، پروتکل‌های استفاده شده و محتوای بسته‌ها را نمایش دهد. این اطلاعات می‌تواند به شناسایی مشکلات عملکردی مانند تنگناها، تاخیرها و از دست رفتن بسته‌ها کمک کند.
علاوه بر آن وایرشارک می‌تواند الگوهای غیرعادی در ترافیک شبکه را شناسایی کرده و به مدیران شبکه هشدار دهد.
وایرشارک می‌تواند به شناسایی مشکلات مرتبط با تنظیمات نادرست پروتکل‌ها، نسخه‌های ناسازگار، و خطاهای پروتکل کمک کند.

۱_۲: پروتکل های open flow , tcp

```
^Cacer@ubuntu:~$ ryu-manager ryu.app.simple_switch_13
loading app ryu.app.simple_switch_13
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch_13 of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 1 00:00:00:00:00:01 33:33:ff:00:00:01 1
packet in 1 00:00:00:00:00:02 33:33:ff:00:00:02 2
packet in 1 00:00:00:00:00:02 33:33:00:00:00:16 2
packet in 1 00:00:00:00:00:01 33:33:00:00:00:16 1
packet in 1 00:00:00:00:00:03 33:33:00:00:00:16 3
packet in 1 00:00:00:00:00:03 33:33:ff:00:00:03 3
packet in 1 00:00:00:00:00:01 33:33:00:00:00:16 1
packet in 1 00:00:00:00:00:01 33:33:00:00:00:02 1
packet in 1 00:00:00:00:00:02 33:33:00:00:00:16 2
packet in 1 00:00:00:00:00:02 33:33:00:00:00:02 2
packet in 1 00:00:00:00:00:03 33:33:00:00:00:16 3
packet in 1 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 1 00:00:00:00:00:03 33:33:00:00:00:16 3
packet in 1 00:00:00:00:00:01 33:33:00:00:00:16 1
packet in 1 00:00:00:00:00:02 33:33:00:00:00:16 2
packet in 1 00:00:00:00:00:01 33:33:00:00:00:02 1
```

```
acer@ubuntu:~$ sudo mn --topo=single,3 --mac --controller=remote,ip=127.0.0.1,port=6633 --switch=ovsk,protocols=OpenFlow13
*** 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>
```

No.	Time	Source	Destination	Protocol	Length	Info
138	63.440443646	127.0.0.1	127.0.0.1	TCP	74	34544 → 5037 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
139	63.440447259	127.0.0.1	127.0.0.1	TCP	54	5037 → 34544 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
140	63.441664016	127.0.0.1	127.0.0.1	TCP	74	34546 → 5037 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
141	63.441668485	127.0.0.1	127.0.0.1	TCP	54	5037 → 34546 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
142	63.479461033	127.0.0.1	127.0.0.1	TCP	66	37472 → 6633 [ACK] Seq=529 Ack=239 Win=44032 Len=0 TSval=2964…
143	63.672069543	127.0.0.1	127.0.0.1	OpenFl…	194	Type: OFPT_PACKET_IN
144	63.672728771	127.0.0.1	127.0.0.1	OpenFl…	192	Type: OFPT_PACKET_OUT
145	63.672734283	127.0.0.1	127.0.0.1	TCP	66	37472 → 6633 [ACK] Seq=657 Ack=365 Win=44032 Len=0 TSval=2964…
146	63.895635398	127.0.0.1	127.0.0.1	OpenFl…	218	Type: OFPT_PACKET_IN
147	63.896355289	127.0.0.1	127.0.0.1	OpenFl…	216	Type: OFPT_PACKET_OUT
148	63.896360506	127.0.0.1	127.0.0.1	TCP	66	37472 → 6633 [ACK] Seq=809 Ack=515 Win=44032 Len=0 TSval=2964…
149	63.959946989	127.0.0.1	127.0.0.1	OpenFl…	218	Type: OFPT_PACKET_IN
150	63.960646051	127.0.0.1	127.0.0.1	OpenFl…	216	Type: OFPT_PACKET_OUT
151	63.960651555	127.0.0.1	127.0.0.1	TCP	66	37472 → 6633 [ACK] Seq=961 Ack=665 Win=44032 Len=0 TSval=2964…
152	64.119765489	127.0.0.1	127.0.0.1	OpenFl…	218	Type: OFPT_PACKET_IN
153	64.120450123	127.0.0.1	127.0.0.1	OpenFl…	216	Type: OFPT_PACKET_OUT
154	64.120455583	127.0.0.1	127.0.0.1	TCP	66	37472 → 6633 [ACK] Seq=1113 Ack=815 Win=44032 Len=0 TSval=296…
155	64.183873880	127.0.0.1	127.0.0.1	OpenFl…	194	Type: OFPT_PACKET_IN
156	64.184612947	127.0.0.1	127.0.0.1	OpenFl…	192	Type: OFPT_PACKET_OUT

:۲_۲

No.	Time	Source	Destination	Protocol	Length	Info
10	4.004202557	127.0.0.1	127.0.0.1	TCP	54	6633 → 38470 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	5.006130084	127.0.0.1	127.0.0.1	TCP	74	38480 → 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
12	5.006159556	127.0.0.1	127.0.0.1	TCP	54	6633 → 38480 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	6.009448316	127.0.0.1	127.0.0.1	TCP	74	38482 → 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
14	6.009477588	127.0.0.1	127.0.0.1	TCP	54	6633 → 38482 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	7.009649525	127.0.0.1	127.0.0.1	TCP	74	55378 → 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
16	7.009699851	127.0.0.1	127.0.0.1	TCP	74	6633 → 55378 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495…
17	7.009709855	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=1 Ack=1 Win=44032 Len=0 TSval=29660700…
18	7.009736408	127.0.0.1	127.0.0.1	OpenFl…	82	Type: OFPT_HELLO
19	7.009738762	127.0.0.1	127.0.0.1	TCP	66	6633 → 55378 [ACK] Seq=1 Ack=17 Win=44032 Len=0 TSval=2966070…
20	7.010479047	127.0.0.1	127.0.0.1	OpenFl…	74	Type: OFPT_HELLO
21	7.010482368	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=17 Ack=9 Win=44032 Len=0 TSval=2966070…
22	7.010493005	127.0.0.1	127.0.0.1	OpenFl…	74	Type: OFPT_FEATURES_REQUEST
23	7.010495135	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=17 Ack=17 Win=44032 Len=0 TSval=296607…
24	7.510860020	127.0.0.1	127.0.0.1	OpenFl…	98	Type: OFPT_FEATURES_REPLY
25	7.511475022	127.0.0.1	127.0.0.1	OpenFl…	82	Type: OFPT_MULTIPART_REQUEST, OFPMP_PORT_DESC
26	7.511480331	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=49 Ack=33 Win=44032 Len=0 TSval=296607…
27	7.511493849	127.0.0.1	127.0.0.1	OpenFl…	146	Type: OFPT_FLOW_MOD
28	7.511495674	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=49 Ack=113 Win=44032 Len=0 TSval=29660…

۳_۲: در ابتدا کنترلر یک بسته به نام features_request برای سوییچ میفرستد تا DPID و قابلیت ها و ویژگی های سوییچ را درخواست کند. این پیغام شامل هدر openflow است و بدنه آن خالی است. در پاسخ به کنترلر سوییچ یک پیغام features_reply میفرستد که شامل تمامی اطلاعات درخواستی کنترلر یعنی DPIDو ویژگی ها و قابلیت های سوییچ است. به طور کلی این پیغام ها رد و بدل شده تا کنترلر تمامی اطلاعات سوییچ را بدست آورده و براساس نیاز های شبکه آن را پیکر بندی کند.

DPID در واقع یک شناسه منحصر به فرد برای هر سوییچ است که کنترلر بوسیله این شناسه این سوییچ را در نظر گرفته و میشناسد.

No.	Time	Source	Destination	Protocol	Length	Info
10	4.004202557	127.0.0.1	127.0.0.1	TCP	54	6633 → 38470 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	5.006130084	127.0.0.1	127.0.0.1	TCP	74	38480 → 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
12	5.006159556	127.0.0.1	127.0.0.1	TCP	54	6633 → 38480 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	6.009448316	127.0.0.1	127.0.0.1	TCP	74	38482 → 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
14	6.009477588	127.0.0.1	127.0.0.1	TCP	54	6633 → 38482 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	7.009649525	127.0.0.1	127.0.0.1	TCP	74	55378 → 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
16	7.009699851	127.0.0.1	127.0.0.1	TCP	74	6633 → 55378 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495…
17	7.009709855	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=1 Ack=1 Win=44032 Len=0 TSval=29660700…
18	7.009736408	127.0.0.1	127.0.0.1	OpenFl…	82	Type: OFPT_HELLO
19	7.009738762	127.0.0.1	127.0.0.1	TCP	66	6633 → 55378 [ACK] Seq=1 Ack=17 Win=44032 Len=0 TSval=2966070…
20	7.010479047	127.0.0.1	127.0.0.1	OpenFl…	74	Type: OFPT_HELLO
21	7.010482368	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=17 Ack=9 Win=44032 Len=0 TSval=2966070…
22	7.010493005	127.0.0.1	127.0.0.1	OpenFl…	74	Type: OFPT_FEATURES_REQUEST
23	7.010495135	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=17 Ack=17 Win=44032 Len=0 TSval=296607…
24	7.510860020	127.0.0.1	127.0.0.1	OpenFl…	98	Type: OFPT_FEATURES_REPLY
25	7.511475022	127.0.0.1	127.0.0.1	OpenFl…	82	Type: OFPT_MULTIPART_REQUEST, OFPMP_PORT_DESC
26	7.511480331	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=49 Ack=33 Win=44032 Len=0 TSval=296607…
27	7.511493849	127.0.0.1	127.0.0.1	OpenFl…	146	Type: OFPT_FLOW_MOD
28	7.511495674	127.0.0.1	127.0.0.1	TCP	66	55378 → 6633 [ACK] Seq=49 Ack=113 Win=44032 Len=0 TSval=29660…

٤_٢:

No.	Time	Source	Destination	Protocol	Length	Info
121	31.091892059	127.0.0.1	127.0.0.1	TCP	54	5037 → 36700 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
122	31.092988141	127.0.0.1	127.0.0.1	TCP	74	36710 → 5037 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
123	31.092992811	127.0.0.1	127.0.0.1	TCP	54	5037 → 36710 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
124	31.094382861	127.0.0.1	127.0.0.1	TCP	74	36724 → 5037 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=…
125	31.094409960	127.0.0.1	127.0.0.1	TCP	54	5037 → 36724 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
126	31.124173479	127.0.0.1	127.0.0.1	TCP	66	6633 → 59964 [ACK] Seq=113 Ack=401 Win=44032 Len=0 TSval=2966…
127	31.148513773	127.0.0.1	127.0.0.1	OpenFl…	218	Type: OFPT_PACKET_IN
128	31.148524521	127.0.0.1	127.0.0.1	TCP	66	6633 → 59964 [ACK] Seq=113 Ack=553 Win=44032 Len=0 TSval=2966…
129	31.149364039	127.0.0.1	127.0.0.1	OpenFl…	216	Type: OFPT_PACKET_OUT
130	31.192205964	127.0.0.1	127.0.0.1	TCP	66	59964 → 6633 [ACK] Seq=553 Ack=263 Win=44032 Len=0 TSval=2966…
131	31.496617945	127.0.0.1	127.0.0.1	OpenFl…	218	Type: OFPT_PACKET_IN
132	31.497346083	127.0.0.1	127.0.0.1	OpenFl…	216	Type: OFPT_PACKET_OUT
133	31.497351146	127.0.0.1	127.0.0.1	TCP	66	59964 → 6633 [ACK] Seq=705 Ack=413 Win=44032 Len=0 TSval=2966…
134	31.592373556	127.0.0.1	127.0.0.1	OpenFl…	194	Type: OFPT_PACKET_IN
135	31.592391415	127.0.0.1	127.0.0.1	OpenFl…	194	Type: OFPT_PACKET_IN
136	31.592710779	127.0.0.1	127.0.0.1	TCP	66	6633 → 59964 [ACK] Seq=413 Ack=961 Win=44032 Len=0 TSval=2966…
137	31.593288191	127.0.0.1	127.0.0.1	OpenFl…	192	Type: OFPT_PACKET_OUT
138	31.593291644	127.0.0.1	127.0.0.1	TCP	66	59964 → 6633 [ACK] Seq=961 Ack=539 Win=44032 Len=0 TSval=2966…
139	31.593305011	127.0.0.1	127.0.0.1	OpenFl…	192	Type: OFPT_PACKET_OUT

٥_٢: یکی از این حالات table miss entry است و زمانی رخ میدهد که بسته به سوییچ می‌رسد و هیچ قانون تطابقی برای آن در جدول جریان (Flow Table) سوییچ وجود ندارد، سوییچ بسته را به کنترلر ارسال می‌کند که همان packet_in است. این حالت به کنترلر اجازه می‌دهد تا در مورد چگونگی رسیدگی به بسته تصمیم‌گیری کند .این فرآیند به کنترلر امکان می‌دهد که قوانین جدید را ایجاد کرده و به سوییچ ارسال کند تا برای بسته‌های مشابه در آینده استفاده شوند. در واقع نیاز است کنترلر فلویی برای این بسته در نظر گرفته و در جدول انتری سوییچ قرار گیرد.

حالت دیگری که میتواند رخ دهد output to controller (action type) است. در برخی موارد، ممکن است قوانین جریان خاصی در جدول سوییچ وجود داشته باشد که مشخص میکند تا بسته‌های مچ شده با آن به کنترلر ارسال شوند. این دستورالعمل معمولاً برای بسته‌هایی استفاده می‌شود که نیاز به پردازش خاص توسط کنترلر دارند، مانند بسته‌هایARP ، DHCP، یا هر نوع بسته‌ای که نیاز به تصمیم‌گیری پیچیده‌تر یا مدیریت مرکزی دارد. یا اگر نیاز به ایجاد یک قانون جدید در جدول مسیریابی برای آن بسته باشد این بسته به کنترلر ارسال میشود.

٦_٢: از پروتکل openflow استفاده شده است. بسته های packet_in نشان دهنده درخواست برای مک آدرس هاست دوم از طرف هاست اول هستند و در بسته های packet_out پاسخ این درخواست ها قرار دارد که از طریق پاسخ هاست دوم به کنترلر به دست آمده است. در نتیجه این فرآیند مک آدرس هاست دوم به سوییچ و هاست اول انتقال داده میشود. در تمام این مراحل کنترلر به عنوان یک واسط تحت پروتکل openflow عمل میکند. در واقع وقتی سوییچ یک بسته پینگ دریافت میکند و نمیداند با آن چکار کند و چگونه برخورد کند سوییچ پیام های packet_in ایجاد میکند.

icmp

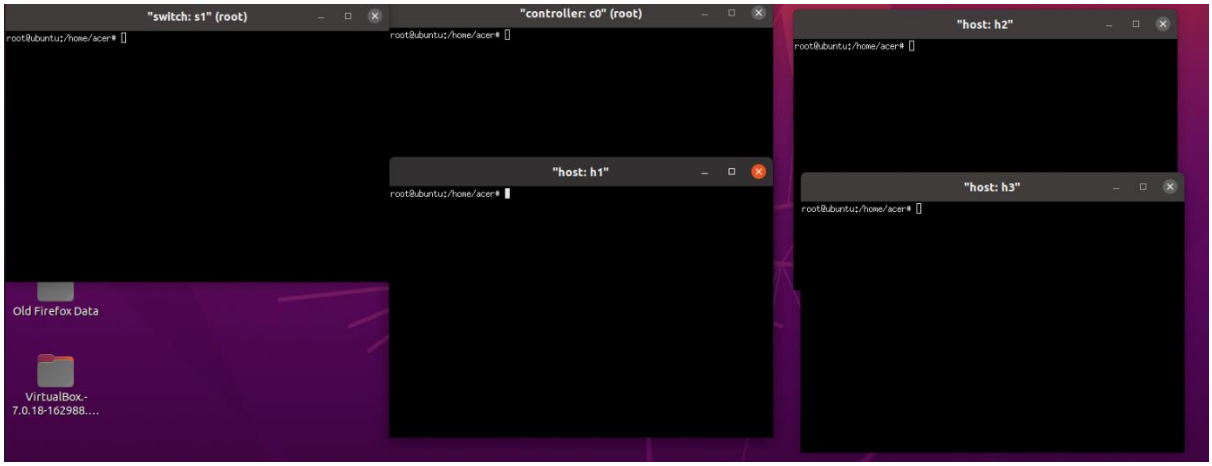
No.	Time	Source	Destination	Protocol	Length	Info
751	44.788077449	10.0.0.1	10.0.0.2	OpenFlow	182	Type: OFPT_PACKET_IN
753	44.788113596	10.0.0.1	10.0.0.2	OpenFlow	188	Type: OFPT_PACKET_OUT
755	44.788582774	10.0.0.2	10.0.0.1	OpenFlow	182	Type: OFPT_PACKET_IN
757	44.788618517	10.0.0.2	10.0.0.1	OpenFlow	188	Type: OFPT_PACKET_OUT

سوال ۳: N=۳ است.

```
acer@ubuntu:~$ sudo mn --topo=single,3 --mac --switch ovsk --controller=remote
*** Creating network
*** Adding controller
Connecting to remote controller at 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>
```

این دستور یک شبکه در mininet میسازد که شامل یک سوییچ و سه هاست متصل به آن است. از طرف دیگر mac– یک آدرس مک یکتا به هر هاست اختصاص میدهد و سپس آپشن switch ovsk– مشخص میکند که mininetاز یک سوییچ open Vswitch که در سطح کرنل است استفاده میکند. و در آخر نیاز یک ریموت کنترلر برای کنترل مرکزی شبکه ساخته شده در نظر گرفته شده است.

```
acer@ubuntu:~$ sudo mn --topo=single,3 --controller=remote -x
[sudo] password for acer:
*** Creating network
*** Adding controller
Connecting to remote controller at 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
*** Running terms on :0
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```



این دستور نیز مانند دستور قبلی یک شبکه سینگل با سه هاست در mininet ایجاد میکند. همچنین آپشن controller–remote نیز بک کنترلر ریموت مرکزی برای کنترل کل شبکه ایجاد کرده. آپشن x- نیز بک xterm برای هر هاست و سوییچ و کنترلر باز میکند تا بتوان با استفاده از آن به صورت اختصاصی تر پکت های دریافتی یا ارسالی برای هر کدام را مشاهده کرد.

ای

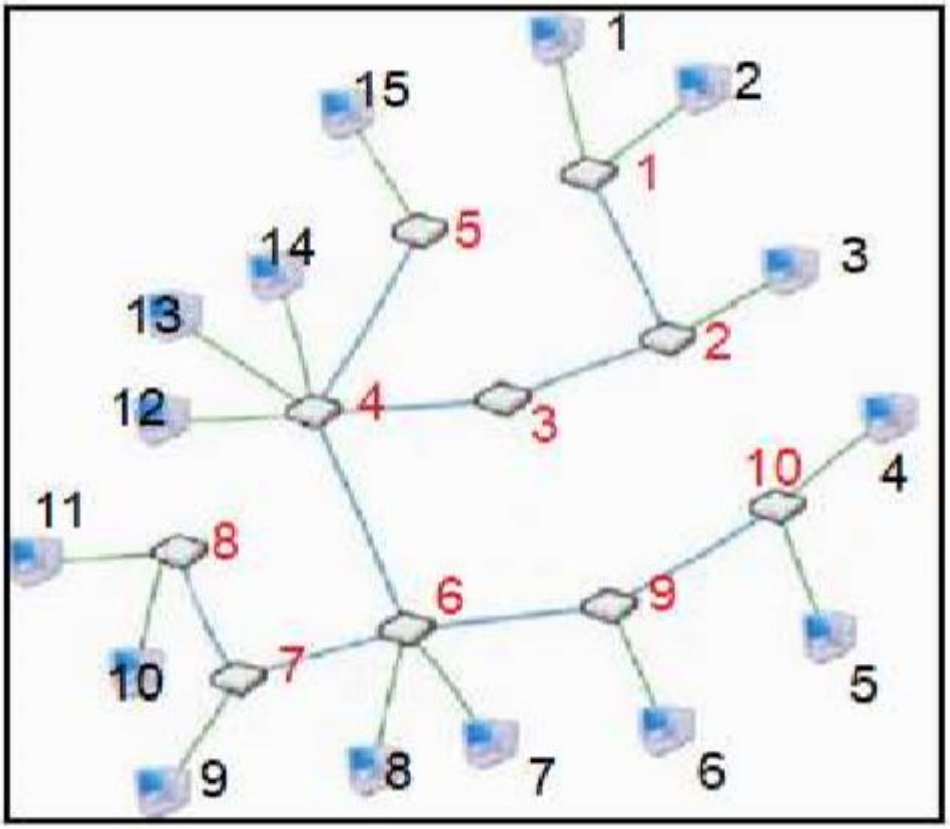
```
acer@ubuntu:~$ sudo mn --topo=tree,3 --mac --arp --controller=remote
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet>
```

این دستور نیز یک شبکه درختی شامل ۳ سطح ایجاد میکند که سوییچ ها به صورت سلسه مراتبی به هاست ها متصل بوده.
آپشن mac– همانطور که قبا ذکر شد یک ادرس مک یکتا برای هر هاست در نظر گرفته. آپشن arp– نیز جدول arpهر هاست را با ورودی های یکدیگر پر میکند. مثلا اگر دو هاست h۲ , h۱را داشته باشیم و از این آپشن استفاده کنیم به هاست h۲ مک آدرسی که در جدول arpهاست h۱ است را میدهد و برعکس.

```
acer@ubuntu:~$ sudo mn --topo linear --controller=remote,ip=127.0.0.1,port=6633
[sudo] password for acer:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s2) (s2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
```

این دستور یک شبکه با توپولوژی خطی ایجاد میکند که هر هاست آن به یک سوییچ مجزا متصل است. ابتدا یک کنترلر ریموت مانند قبل مشخص شده و سپس آیپی ادرس و شماره پورت کنترلر مورد نظر را به کنترلر اختصاص میدهد که در اینجا ما ادرس ایپی لوکال ۱۲۷,۰,۰,۱ و شماره پورت ٦٦٣٣را که مربوط به کنترلر ryu است در نظر گرفتیم.

۱_٤: کد پایتون براساس شماره گذاری های زیر نوشته شده است:




```
acer@ubuntu:~$ sudo mn --custom ./Q4.py --topo mytopo --controller=remote
[sudo] password for acer:
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Adding switches:
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
*** Adding links:
(S1, S2) (S2, S3) (S3, S4) (S4, S5) (S4, S6) (S6, S7) (S6, S9) (S7, S8) (S9, S10) (h1, S1) (h2, S1) (h3, S2) (h4, S10) (h5, S10) (h6, S9) (h7, S6) (h8, S6) (h9, S7) (h10, S8) (h11,
S8) (h12, S4) (h13, S4) (h14, S4) (h15, S5)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Starting controller
c0
*** Starting 10 switches
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 ...
*** Starting CLI:
mininet>
```

همانطور که در شکل نیز مشخص است ۱۰ سوییچ و ۱۵ هاست داریم.

٢_٤:

```
mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=0.428 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.234 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.240 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.069 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.202 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.165 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=0.224 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=0.081 ms
64 bytes from 10.0.0.5: icmp_seq=9 ttl=64 time=0.084 ms
64 bytes from 10.0.0.5: icmp_seq=10 ttl=64 time=0.074 ms
64 bytes from 10.0.0.5: icmp_seq=11 ttl=64 time=0.289 ms
c64 bytes from 10.0.0.5: icmp_seq=12 ttl=64 time=0.115 ms
64 bytes from 10.0.0.5: icmp_seq=13 ttl=64 time=0.236 ms
64 bytes from 10.0.0.5: icmp_seq=14 ttl=64 time=0.210 ms
^C
--- 10.0.0.5 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13263ms
rtt min/avg/max/mdev = 0.069/0.189/0.428/0.097 ms
mininet>
```

```
mininet> h13 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=40.2 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.855 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.217 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.163 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.172 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=0.094 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=0.201 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=0.198 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=0.205 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=0.173 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=0.200 ms
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=0.291 ms
^C
--- 10.0.0.4 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11216ms
rtt min/avg/max/mdev = 0.094/3.580/40.202/11.043 ms
mininet>
```

```
mininet> h11 ping 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=52.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.930 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.256 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.092 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.333 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.112 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.235 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.132 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.203 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.215 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.204 ms
^C
--- 10.0.0.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10185ms
rtt min/avg/max/mdev = 0.092/5.058/52.933/15.140 ms
mininet>
```

۳_٤: به دلیل cold start latency زمان پینگ برای اولین تلاش بزرگتر از بقیه تلاش هاست. علت اصلی این مشکل این است که هنوز جداول arp اپدیت نشده اند. پس در اولین گام آپدیت کردن این جداول باید اتفاق بیفتد که خود کمی زمانبر است.

٤_٤: نتیجه dump

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=4229>
<Host h2: h2-eth0:10.0.0.2 pid=4231>
<Host h3: h3-eth0:10.0.0.3 pid=4233>
<Host h4: h4-eth0:10.0.0.4 pid=4235>
<Host h5: h5-eth0:10.0.0.5 pid=4237>
<Host h6: h6-eth0:10.0.0.6 pid=4239>
<Host h7: h7-eth0:10.0.0.7 pid=4241>
<Host h8: h8-eth0:10.0.0.8 pid=4243>
<Host h9: h9-eth0:10.0.0.9 pid=4245>
<Host h10: h10-eth0:10.0.0.10 pid=4247>
<Host h11: h11-eth0:10.0.0.11 pid=4249>
<Host h12: h12-eth0:10.0.0.12 pid=4251>
<Host h13: h13-eth0:10.0.0.13 pid=4253>
<Host h14: h14-eth0:10.0.0.14 pid=4255>
<Host h15: h15-eth0:10.0.0.15 pid=4257>
<OVSSwitch S1: lo:127.0.0.1,S1-eth1:None,S1-eth2:None,S1-eth3:None pid=4262>
<OVSSwitch S2: lo:127.0.0.1,S2-eth1:None,S2-eth2:None,S2-eth3:None pid=4265>
<OVSSwitch S3: lo:127.0.0.1,S3-eth1:None,S3-eth2:None pid=4268>
<OVSSwitch S4: lo:127.0.0.1,S4-eth1:None,S4-eth2:None,S4-eth3:None,S4-eth4:None,S4-eth5:None,S4-eth6:None pid=4271>
<OVSSwitch S5: lo:127.0.0.1,S5-eth1:None,S5-eth2:None pid=4274>
<OVSSwitch S6: lo:127.0.0.1,S6-eth1:None,S6-eth2:None,S6-eth3:None,S6-eth4:None,S6-eth5:None pid=4277>
```

```
<OVSSwitch S7: lo:127.0.0.1,S7-eth1:None,S7-eth2:None,S7-eth3:None pid=4280>
<OVSSwitch S8: lo:127.0.0.1,S8-eth1:None,S8-eth2:None,S8-eth3:None pid=4283>
<OVSSwitch S9: lo:127.0.0.1,S9-eth1:None,S9-eth2:None,S9-eth3:None pid=4286>
<OVSSwitch S10: lo:127.0.0.1,S10-eth1:None,S10-eth2:None,S10-eth3:None pid=4289>
<RemoteController c0: 127.0.0.1:6653 pid=4223>
mininet>
```

نتیجه nodes

```
mininet> nodes
available nodes are:
S1 S10 S2 S3 S4 S5 S6 S7 S8 S9 c0 h1 h10 h11 h12 h13 h14 h15 h2 h3 h4 h5 h6 h7 h8 h9
mininet>
```

نتیجه pingall

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14
*** Results: 0% dropped (210/210 received)
mininet>
```

از تمامی هاست ها میتوان به یکدیگر بسته ارسال کرد و امکان پینگ وجود دارد.