سوال اول)

هر دستور چه کار می کند:

sudo mn --topo single,3 --mac --switch ovsk:

یک توپولوژی single با سه هاست و یک سوییچ ایجاد می کند. که شکل توپولوژی به صورت زیر است:

```
c0
|
s1
/ | \
h1 h2 h3
```

همچنین –mac به هر interface یک mac address اختصاص می دهد و ovsk می گوید که ما میخواهیم open vswitch kernel module را در بحث پیاده سازی سوئیچ داشته باشیم.

sudo mn --topo single,3 --mac --controller remote -x:

یک توپولوژی single با سه هاست و یک سوییچ ایجاد می کند. که شکل توپولوژی مانند دستور قبل است.

هنگام زدن دستور 5 ترمینال متناظر با هر یک از node های شبکه popup شد. همچنین --controller remote عنوان می دارد که ما می خواهیم remote controller از openflow باشد و -x نیز همان ترمینال های گفته شده را نشان می دهد.

sudo mn --topo tree,3 --mac --arp

یک توپولوژی درخت است که شکل آن در زیر رسم شده است:

همچنین –mac به هر interface یک mac address می دهد و –arp به صورت automatic و static جدول arp را در نود ها configure می کند که برای تحلیل و شبیه سازی های که این نیاز را دارند ضروری است.

sudo mn --topo linear --controller=remote,ip=127.0.0.1,port=6633

دارای توپولوژی خطی است که شکل آن در زیر رسم شده است:

```
h1 - s1 - s2 - h2
```

همچنین remote controller بر روی ip 127.0.0.1 و port 6633 ست شده است.

سوال دوم)

خروجی دستور مطابق شکل زیر است:

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
```

توجیه خروجی:

پس از اجرای دستور ارور زیر را میگیرم که فکر میکنم دلیل این خروجی این باشد:

```
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
```

در واقع اتصال به controller موفقیت آمیز نبوده است.

سوال سوم)
شماره دانشجویی: 9900283
بخش اول) توپولوژی انتخابی شکل اول.
بخش دوم) تست پینگ چند میزبان مختلف به صورت رندوم:

```
mininet> h3 ping h7
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=9.46 ms
64 bytes from 10.0.0.7: icmp_seq=2 ttl=64 time=1.04 ms
64 bytes from 10.0.0.7: icmp_seq=3 ttl=64 time=0.129 ms
64 bytes from 10.0.0.7: icmp_seq=4 ttl=64 time=0.115 ms
```

```
mininet> h2 ping h6
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
64 bytes from 10.0.0.6: icmp_seq=1 ttl=64 time=11.7 ms
64 bytes from 10.0.0.6: icmp_seq=2 ttl=64 time=1.06 ms
64 bytes from 10.0.0.6: icmp_seq=3 ttl=64 time=0.134 ms
64 bytes from 10.0.0.6: icmp_seq=4 ttl=64 time=0.135 ms
64 bytes from 10.0.0.6: icmp_seq=5 ttl=64 time=0.130 ms
```

بخش سوم) دلیل اینکه در اولین تلاش تاخیر پینگ بیشتر است اجرای پروتکل ARP می باشد تا هاست بتواند با استفاده از ip مک آدرس مورد نظر را پیدا کند.
بخش چهارم)

Dump: node name, interface name, IP address, MAC address, and other relevant configuration information for each node and interface in the Mininet network

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=2029>
<Host h2: h2-eth0:10.0.0.2 pid=2031>
<Host h3: h3-eth0:10.0.0.3 pid=2033>
<Host h4: h4-eth0:10.0.0.4 pid=2035>
<Host h5: h5-eth0:10.0.0.5 pid=2037>
<Host h6: h6-eth0:10.0.0.6 pid=2039>
<Host h7: h7-eth0:10.0.0.7 pid=2041>
<Host h8: h8-eth0:10.0.0.8 pid=2043>
<Host h9: h9-eth0:10.0.0.9 pid=2045>
<Host h10: h10-eth0:10.0.0.10 pid=2047>
<Host h11: h11-eth0:10.0.0.11 pid=2049>
<Host h12: h12-eth0:10.0.0.12 pid=2051>
<Host h13: h13-eth0:10.0.0.13 pid=2053>
<Host h14: h14-eth0:10.0.0.14 pid=2055>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=2060>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None pid=2063>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=2066>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-eth4:None,s4-eth5:None pid=2069>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None,s5-eth3:None pid=2072>
<OVSSwitch s6: lo:127.0.0.1,s6-eth1:None,s6-eth2:None,s6-eth3:None,s6-eth4:None pid=2075>
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None,s7-eth4:None pid=2078>
<OVSSwitch s8: lo:127.0.0.1,s8-eth1:None,s8-eth2:None pid=2081>
<Controller c0: 127.0.0.1:6653 pid=2022>
```

Nodes: the nodes in the mininet network

```
available nodes are:
c0 h1 h10 h11 h12 h13 h14 h2 h3 h4 h5 h6 h7 h8 h9 s1 s2 s3 s4 s5 s6 s7 s8
```

Pingall:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
*** Results: 0% dropped (182/182 received)
```

Net: includes links and interfaces.

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s2-eth1
h4 h4-eth0:s2-eth2
h5 h5-eth0:s3-eth1
h6 h6-eth0:s4-eth1
h7 h7-eth0:s4-eth2
h8 h8-eth0:s4-eth3
h9 h9-eth0:s5-eth1
h10 h10-eth0:s6-eth1
h11 h11-eth0:s6-eth2
h12 h12-eth0:s7-eth1
h13 h13-eth0:s7-eth2
h14 h14-eth0:s8-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:s2-eth3
s2 lo:  s2-eth1:h3-eth0 s2-eth2:h4-eth0 s2-eth3:s1-eth3 s2-eth4:s3-eth2
s3 lo:  s3-eth1:h5-eth0 s3-eth2:s2-eth4 s3-eth3:s4-eth4
s4 lo:  s4-eth1:h6-eth0 s4-eth2:h7-eth0 s4-eth3:h8-eth0 s4-eth4:s3-eth3 s4-eth5:s5-eth2
s5 lo:  s5-eth1:h9-eth0 s5-eth2:s4-eth5 s5-eth3:s6-eth3
s6 lo:  s6-eth1:h10-eth0 s6-eth2:h11-eth0 s6-eth3:s5-eth3 s6-eth4:s7-eth3
s7 lo:  s7-eth1:h12-eth0 s7-eth2:h13-eth0 s7-eth3:s6-eth4 s7-eth4:s8-eth2
s8 lo:  s8-eth1:h14-eth0 s8-eth2:s7-eth4
c0
```