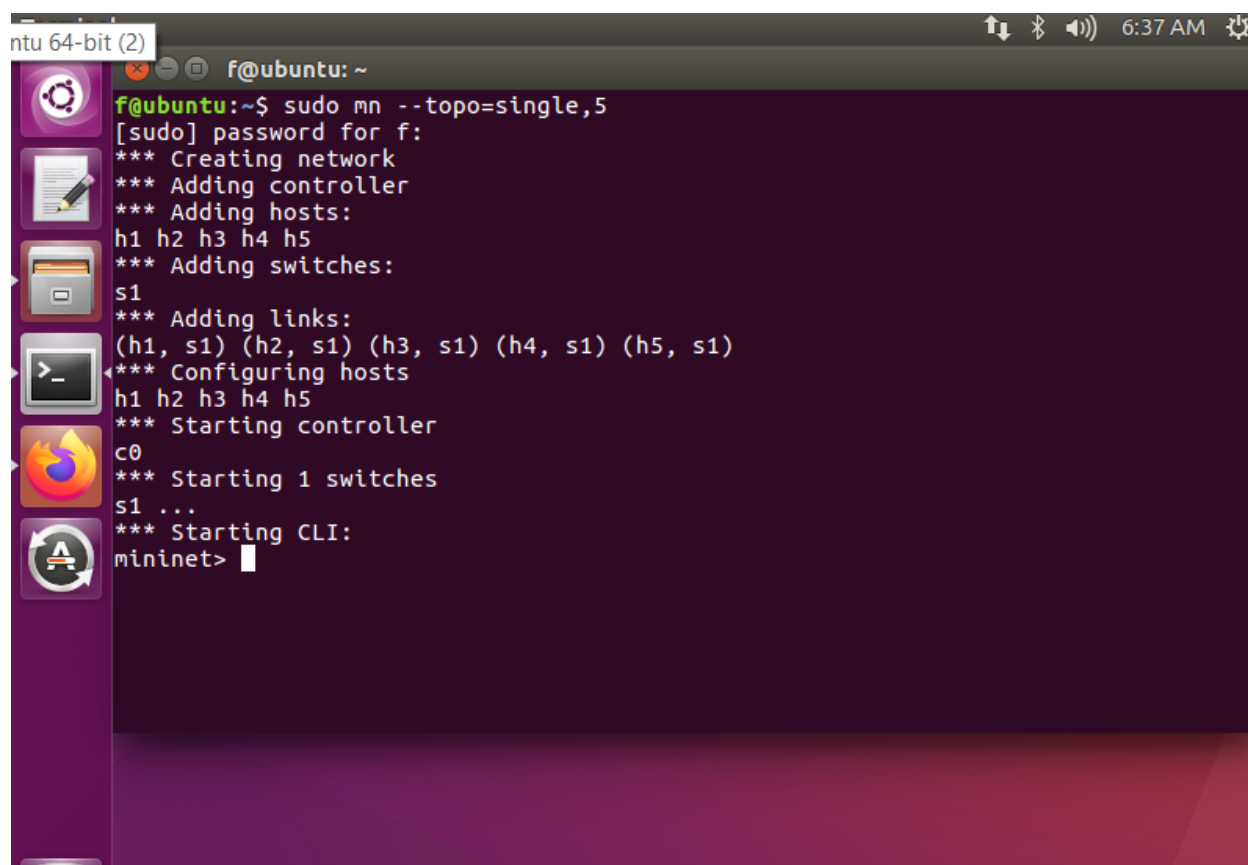


(دو)

توپولوژی single : در این حالت فقط یک سویچ داریم و تعدادی میزبان که همگی به آن سویچ متصل هستند. تعداد میزبان ها را به عنوان پارامتر ورودی در هنگام ساخت توپولوژی تعیین میکنیم :

با دستور `sudo mn --topo=single,n` توپولوژی single ای با n میزبان ساخته می شود.

مثال:



```
ntu 64-bit (2) f@ubuntu: ~
f@ubuntu:~$ sudo mn --topo=single,5
[sudo] password for f:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

توپولوژی linear: توپولوژی خطی به این گونه است که n سویچ و n میزبان دارد به طوری که هر میزبان به یکی از سویچ ها متصل میشود و هر سویچ به شکل خطی به سویچ بعدی خود متصل میشود. در هنگام ساخت این توپولوژی n به صورت پارامتر ورودی داده میشود.

با دستور `sudo mn --topo=linear,n` توپولوژی single ای با n سویچ و n میزبان ساخته می شود.

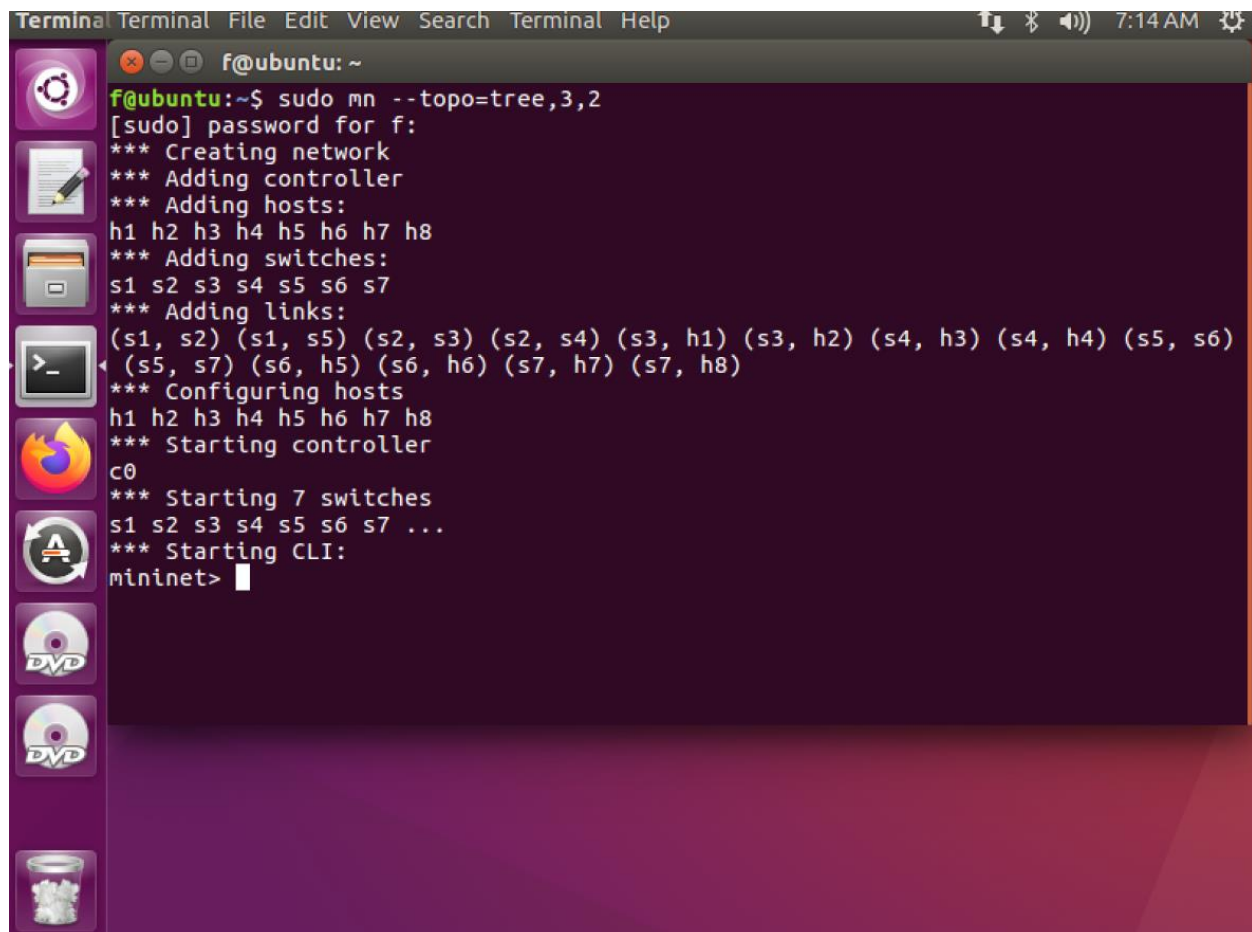
مثال:

```
Terminal Terminal File Edit View Search Terminal Help
f@ubuntu: ~
f@ubuntu:~$ sudo mn --topo=linear,7
[sudo] password for f:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (h7, s7) (s2, s1) (s3, s2)
(s4, s3) (s5, s4) (s6, s5) (s7, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet>
```

توپولوژی tree : توپولوژی درختی به این گونه است که m لایه سویچ دارد، در لایه ی اول فقط یک سویچ هست، در لایه ی بعدی n سویچ به هر سویچ لایه ی قبل متصل میشوند یعنی در هر لایه نسبت به لایه ی قبل n برابر سویچ داریم به این صورت در لایه ی نهایی n^m سویچ داریم و به هر کدام از سویچ های لایه ی آخر هم یک میزبان متصل میشود یعنی n^m میزبان در این توپولوژی وجود دارد. پارامتر های m, n نیز در هنگام ساخت توپولوژی به عنوان ورودی داده میشوند.

با دستور `sudo mn --topo=tree,3,2` توپولوژی tree ای با m لایه سویچ و n^m میزبان ساخته می شود.

مثال:

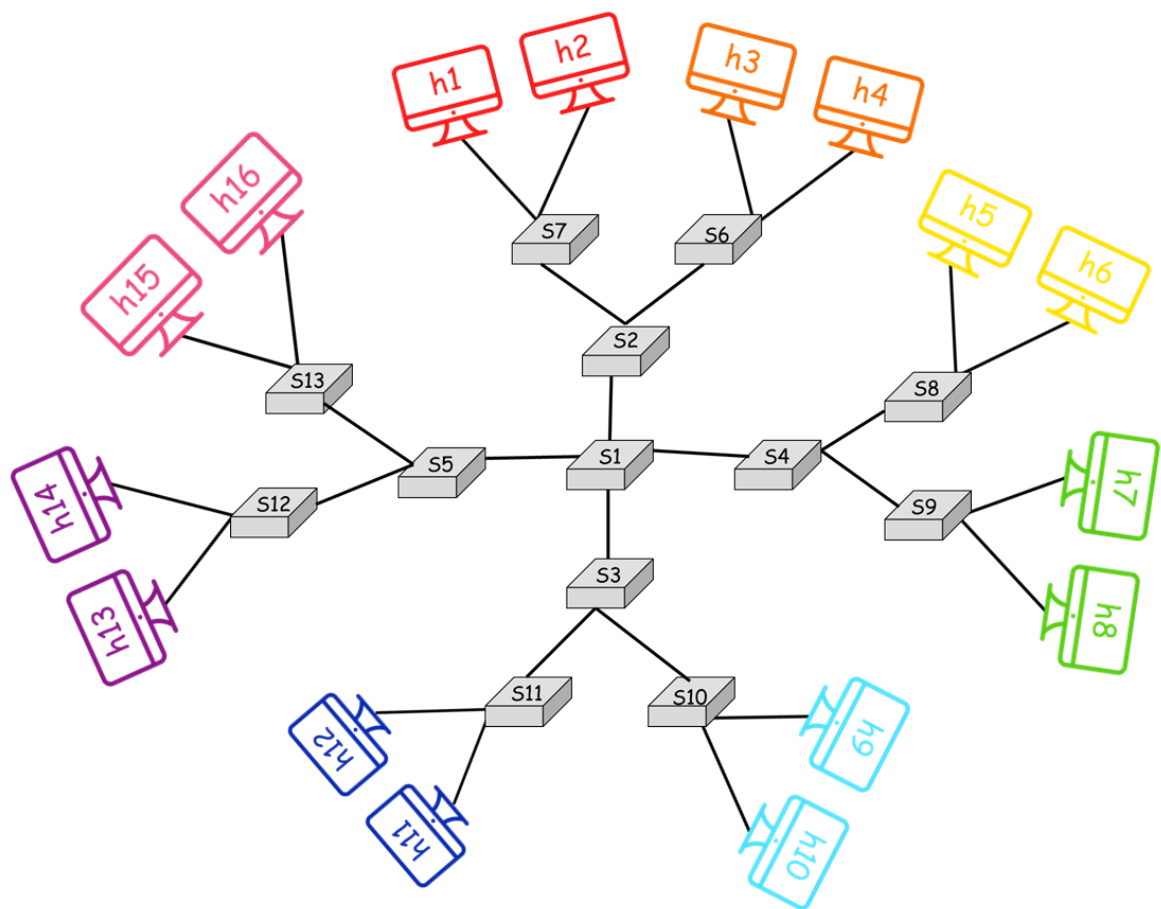


```

Terminal Terminal File Edit View Search Terminal Help
f@ubuntu: ~
f@ubuntu:~$ sudo mn --topo=tree,3,2
[sudo] password for f:
*** Creating network
*** Adding controller
*** 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>

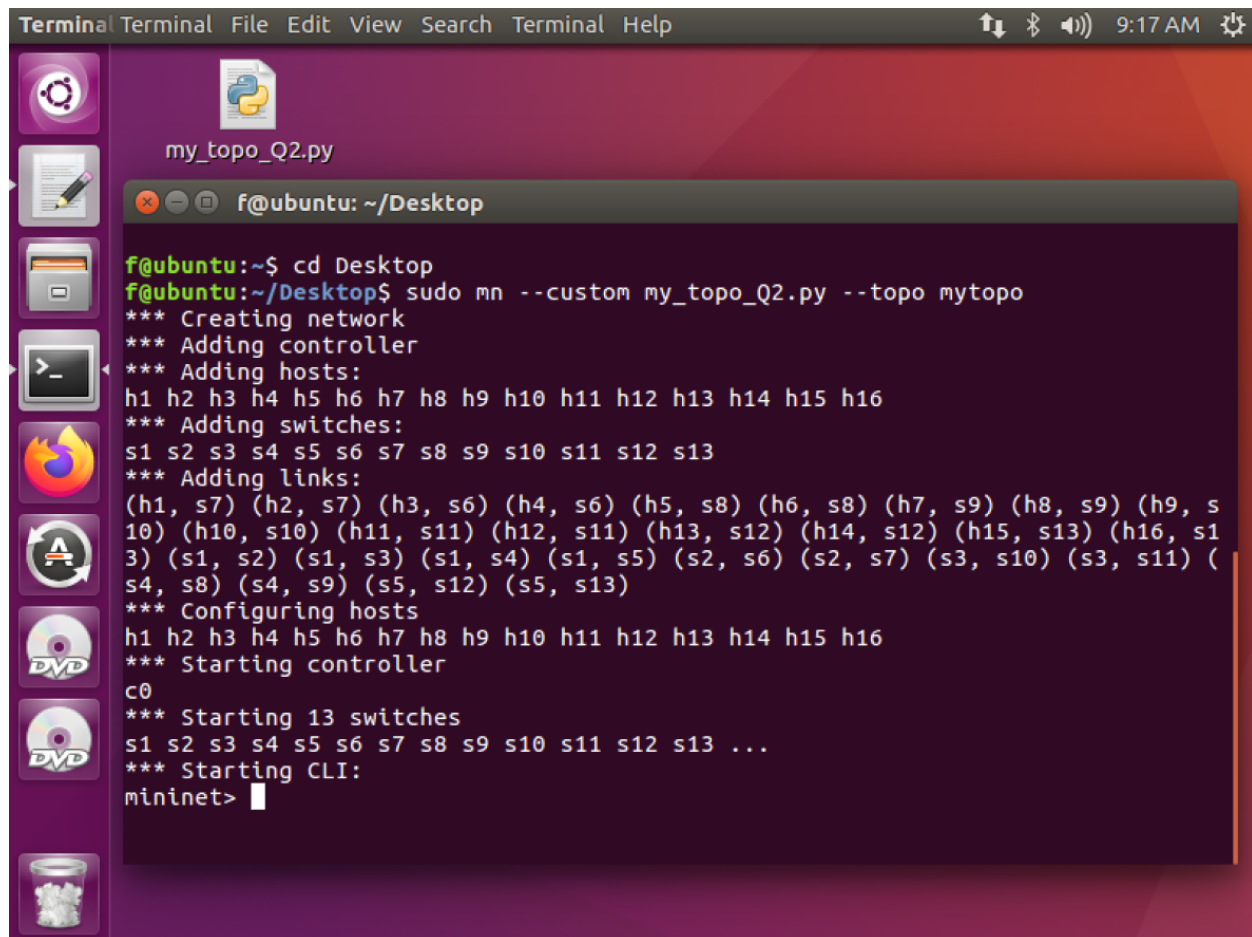
```

سه) توپولوژی طراحی شده به شکل زیر است :



کد مورد نظر با نام my_topo_Q۲.py ضمیمه شده است.

با دستور اجرای `sudo mn --custom my_topo_Q2.py --topo mytopo` کد اجرا شده و توپولوژی ایجاد میشود:



The screenshot shows a terminal window titled "Terminal" with a menu bar (Terminal, File, Edit, View, Search, Terminal, Help) and system status icons (network, volume, 9:17 AM). The desktop background is Ubuntu's default purple. A file icon for "my_topo_Q2.py" is on the desktop. The terminal window title is "f@ubuntu: ~/Desktop". The command executed is `cd Desktop` followed by `sudo mn --custom my_topo_Q2.py --topo mytopo`. The output shows the creation of a network with 16 hosts (h1-h16), 13 switches (s1-s13), and a controller (c0), along with the configuration of links between them. The prompt "mininet>" is visible at the end.

```
f@ubuntu:~/Desktop
f@ubuntu:~$ cd Desktop
f@ubuntu:~/Desktop$ sudo mn --custom my_topo_Q2.py --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13
*** Adding links:
(h1, s7) (h2, s7) (h3, s6) (h4, s6) (h5, s8) (h6, s8) (h7, s9) (h8, s9) (h9, s
10) (h10, s10) (h11, s11) (h12, s11) (h13, s12) (h14, s12) (h15, s13) (h16, s1
3) (s1, s2) (s1, s3) (s1, s4) (s1, s5) (s2, s6) (s2, s7) (s3, s10) (s3, s11) (
s4, s8) (s4, s9) (s5, s12) (s5, s13)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Starting controller
c0
*** Starting 13 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 ...
*** Starting CLI:
mininet>
```


امتحان ping بین چند میزبان :

میزبان ۱ و ۵ :

```
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=23.9 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=1.03 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.204 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.051 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.832 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.198 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=0.177 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=0.167 ms
64 bytes from 10.0.0.5: icmp_seq=9 ttl=64 time=0.176 ms
64 bytes from 10.0.0.5: icmp_seq=10 ttl=64 time=0.169 ms
64 bytes from 10.0.0.5: icmp_seq=11 ttl=64 time=0.120 ms
64 bytes from 10.0.0.5: icmp_seq=12 ttl=64 time=0.111 ms
64 bytes from 10.0.0.5: icmp_seq=13 ttl=64 time=0.193 ms
64 bytes from 10.0.0.5: icmp_seq=14 ttl=64 time=0.264 ms
64 bytes from 10.0.0.5: icmp_seq=15 ttl=64 time=0.235 ms
64 bytes from 10.0.0.5: icmp_seq=16 ttl=64 time=0.163 ms
```

میزبان ۱۳ و ۱۴ :

```
mininet> h13 ping h14
PING 10.0.0.14 (10.0.0.14) 56(84) bytes of data.
64 bytes from 10.0.0.14: icmp_seq=1 ttl=64 time=6.84 ms
64 bytes from 10.0.0.14: icmp_seq=2 ttl=64 time=0.680 ms
64 bytes from 10.0.0.14: icmp_seq=3 ttl=64 time=0.124 ms
64 bytes from 10.0.0.14: icmp_seq=4 ttl=64 time=0.175 ms
64 bytes from 10.0.0.14: icmp_seq=5 ttl=64 time=0.110 ms
64 bytes from 10.0.0.14: icmp_seq=6 ttl=64 time=0.140 ms
64 bytes from 10.0.0.14: icmp_seq=7 ttl=64 time=0.079 ms
64 bytes from 10.0.0.14: icmp_seq=8 ttl=64 time=0.135 ms
64 bytes from 10.0.0.14: icmp_seq=9 ttl=64 time=0.136 ms
64 bytes from 10.0.0.14: icmp_seq=10 ttl=64 time=0.147 ms
64 bytes from 10.0.0.14: icmp_seq=11 ttl=64 time=0.132 ms
64 bytes from 10.0.0.14: icmp_seq=12 ttl=64 time=0.135 ms
64 bytes from 10.0.0.14: icmp_seq=13 ttl=64 time=0.146 ms
64 bytes from 10.0.0.14: icmp_seq=14 ttl=64 time=0.136 ms
64 bytes from 10.0.0.14: icmp_seq=15 ttl=64 time=0.134 ms
64 bytes from 10.0.0.14: icmp_seq=16 ttl=64 time=0.134 ms
64 bytes from 10.0.0.14: icmp_seq=17 ttl=64 time=0.134 ms
64 bytes from 10.0.0.14: icmp_seq=18 ttl=64 time=0.127 ms
```

```
mininet> h16 ping h9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=64 time=16.7 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=64 time=4.45 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 10.0.0.9: icmp_seq=4 ttl=64 time=0.274 ms
64 bytes from 10.0.0.9: icmp_seq=5 ttl=64 time=0.190 ms
64 bytes from 10.0.0.9: icmp_seq=6 ttl=64 time=0.176 ms
64 bytes from 10.0.0.9: icmp_seq=7 ttl=64 time=0.178 ms
64 bytes from 10.0.0.9: icmp_seq=8 ttl=64 time=0.163 ms
64 bytes from 10.0.0.9: icmp_seq=9 ttl=64 time=0.049 ms
64 bytes from 10.0.0.9: icmp_seq=10 ttl=64 time=0.094 ms
64 bytes from 10.0.0.9: icmp_seq=11 ttl=64 time=0.114 ms
64 bytes from 10.0.0.9: icmp_seq=12 ttl=64 time=0.067 ms
64 bytes from 10.0.0.9: icmp_seq=13 ttl=64 time=0.174 ms
```

شش مسیر با حداکثر طول و حداقل تلاقی :

h1 → S7 → S2 → S1 → S3 → S10 → h9
h5 → S8 → S4 → S1 → S5 → S13 → h16
h11 → S11 → S3 → S1 → S5 → S12 → h13
h4 → S6 → S2 → S1 → S4 → S9 → h8
h2 → S7 → S2 → S1 → S5 → S13 → h15
H6 → S8 → S4 → S1 → S3 → S10 → h10

چهار Floodlight یک کنترل کننده OpenFlow با ویژگی‌های زیر است:

Enterprise-class

Apache-licensed

Java-based

طراحی کنترل کننده Floodlight با کارایی بالا بوده و در شبکه‌های با تعداد مؤلفه بالا به خوبی مقیاس پذیر است. کنترل کننده Floodlight مبتنی بر کنترل کننده دیگری تحت عنوان Beacon می باشد. زبان برنامه نویسی جاوا بدین دلیل برای آن انتخاب شده که از توازن مناسبی میان کارایی و کاربرپسند بودن برخوردار است. همچنین پرتابل هم می باشد، یعنی این که روی انواع مختلف سیستم عامل قابل اجرا است. علاوه بر این، Beacon و نیز Floodlight دارای واسط برنامه نویسی کاربری خوب و ساده ای است که به همراه برنامه های کاربردی مفیدی عرضه می شوند، از جمله:

Device Manager : دستگاه هایی که در شبکه دیده شده اند را ردیابی می کند. این ردیابی شامل مواردی از قبیل اطلاعات آدرس آن ها، آخرین تاریخ رؤیت آن ها، و آخرین سوئیچ و پورتی که در آن رؤیت شده اند می باشد.

Topology : لینک های مابین سوئیچ های OpenFlow را کشف می کند.

Routing : کوتاه ترین مسیریابی لایه ۲ را میان دستگاه های شبکه فراهم می کند.

Web : یک واسط کاربری تحت وب فراهم می کند.

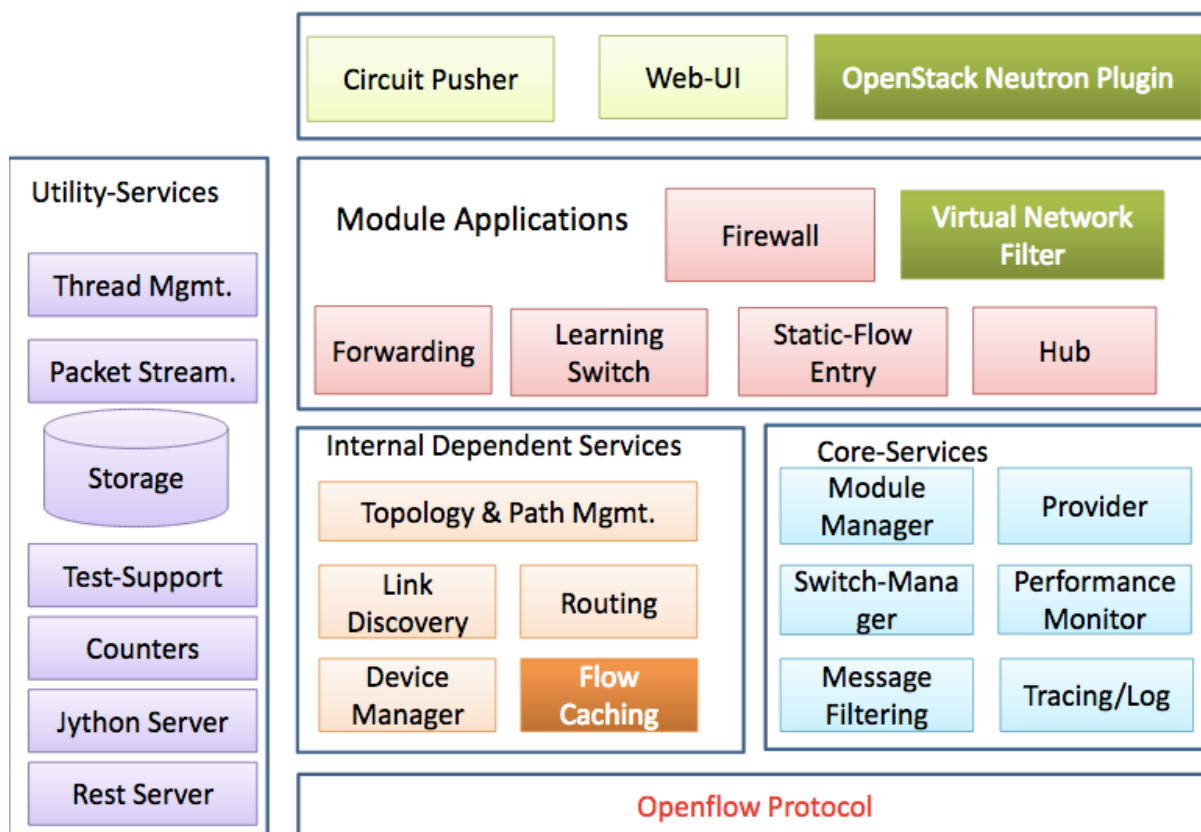
کنترلر فلاادلایت تنها یک کنترلر اوپن فلو نیست. بلکه یک کنترلر با مجموعه ای از برنامه های کاربردی است که بر روی آن پیاده سازی شده اند.

کنترلر فلاادلایت شامل مجموعه ای از قابلیت ها برای کنترل و مدیریت شبکه های اوپن فلو می باشد و برنامه های کاربردی آن ویژگی های مورد نیاز کاربر را برآورده می کند.

از دیگر مزایا و قابلیت ها :

- امکان نوشتن برنامه برای کنترلر به زبان دلخواه
- انعطاف پذیری در انتقال برنامه کاربردی از یک VM به VM دیگر بدون جابجایی کنترلر
- امکان توسعه راحت تر برنامه های کاربردی و مهاجرت از یک کنترلر به کنترلر دیگر در آینده

معماری foodlight به شکل ماژولار است و در شکل زیر به طور کامل تمامی بخش ها و سرویس های آن نشان داده شده است :



منابع استفاده شده برای سوال ۴ (برای اطلاعات بیشتر):

<https://sdncentral.ir/-api-۹b۸٪d۷٪a۸٪d۹٪۸۸٪d۹٪۸۶٪d۷٪a۸٪/%d۰۲/۰۶/۲۰۱۷>

<http://nooran.com/blog/entry/various/software-defined-network-sdn>

<https://thenewstack.io/sdn-series-part-v-floodlight>