

بسمه تعالی
دانشکده ی مهندسی برق و کامپیوتر
دانشگاه صنعتی اصفهان

شبکه‌های کامپیوتری ۲ – نیمسال اول ۱۳۹۹-۱۴۰۰
گزارش پروژه شماره یک – تحویل سه‌شنبه ۱۳۹۹/۸/۶

مریم سعیدمهر – ش.د. : ۹۶۲۹۳۷۳

برای ایجاد توپولوژی در Mininet دو راه داریم :

1. استفاده از توپولوژی‌های آماده در Mininet .

I. توپولوژی Single

II. توپولوژی Linear

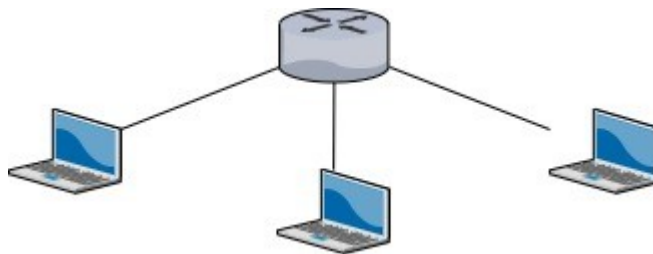
III. توپولوژی Tree

2. استفاده از کتابخانه‌های موجود و برنامه نویسی با زبان پایتون .

معرفی توپولوژی‌های آماده Mininet :

I -1 : توپولوژی Single :

در این توپولوژی یک سوئیچ خواهیم داشت با هر تعداد هاستی که بخواهیم. شبکه به این صورت خواهد بود که تمام هاست ها به تنها سوئیچ موجود متصل می‌شوند. مثلاً توپولوژی single,3 به صورت زیر است :

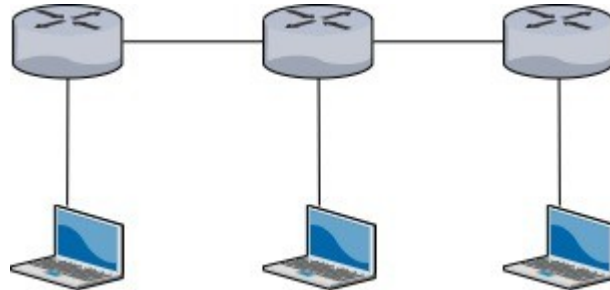


پیاده سازی توپولوژی single,3 در Mininet :

```
sudo mn --topo single,3
*** 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> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

1- II : توپولوژی Linear :

در این توپولوژی به تعداد مساوی سوئیچ و هاست داریم که هر هاست به صورت اختصاصی به یک سوئیچ متصل است و تمام سوئیچ‌ها به صورت خطی به یکدیگر متصل هستند. مثلاً توپولوژی Linear,3 به صورت زیر است :

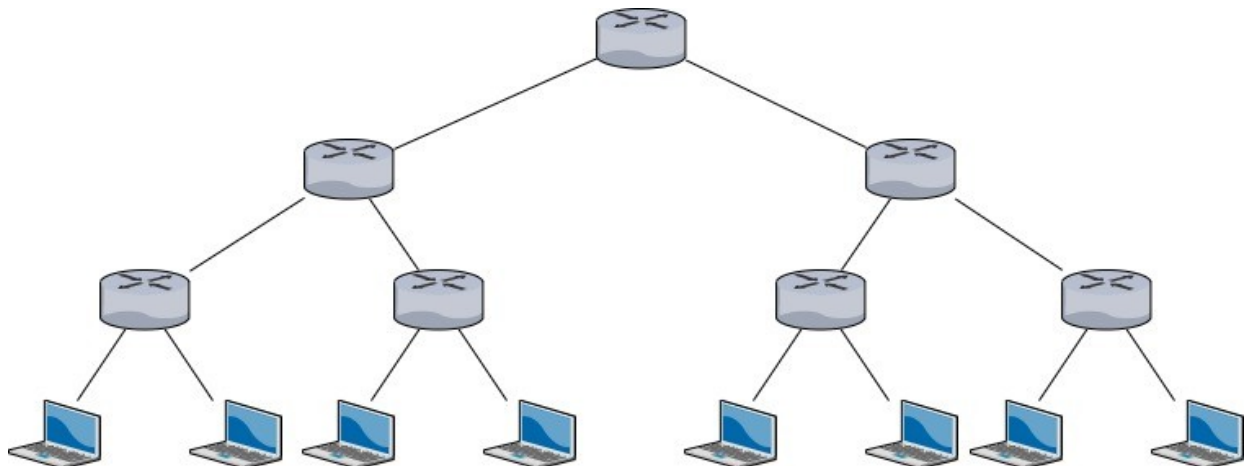


پیاده سازی توپولوژی Linear,3 در Mininet :

```
sudo mn --topo linear,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> |
```

1- III : توپولوژی Tree :

در این توپولوژی ابتدا یک درخت دودویی از سوئیچ‌ها می‌سازیم و سپس به سوئیچ‌های انتهایی (برگ‌های درخت) هاست‌ها را متصل می‌کنیم. مثلاً اگر یک توپولوژی Tree با عمق ۳ بسازیم (در ساختن این توپولوژی در Mininet باید بعد از مشخص کردن نوع توپولوژی ، عمق درخت را مشخص کنیم!) ، ۷ سوئیچ و ۸ هاست خواهیم داشت. مثلاً توپولوژی Tree,3 به صورت زیر است :



تعداد سوئیچ‌ها در درخت در سطح (L) برابر است با 2^{L-1} . (سطح اول شامل ۱ نود است)

تعداد سوئیچ‌ها در درخت با عمق N برابر است با : $\sum_{i=1}^N 2^{i-1} = 2^N - 1$

تعداد هاست‌ها در درخت با عمق N برابر است با : 2^N

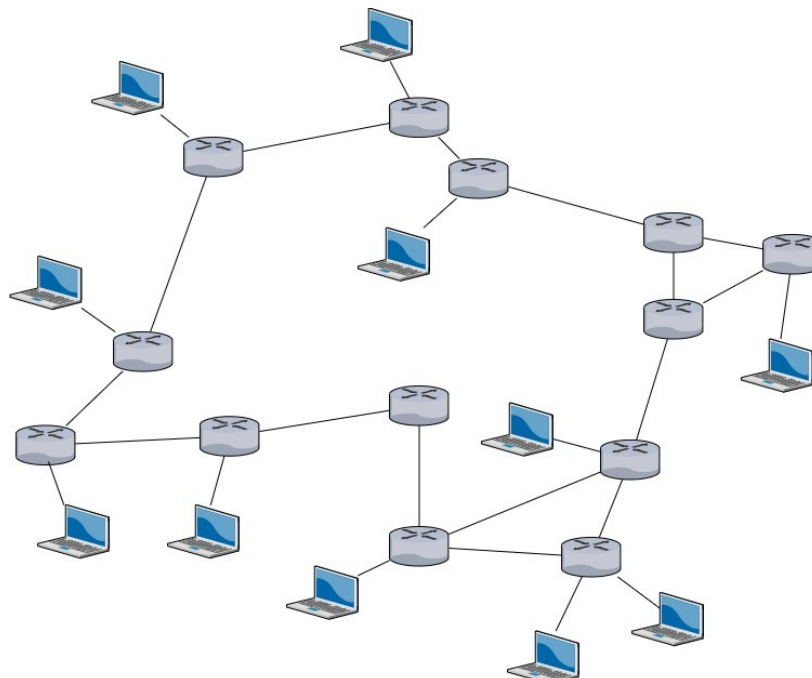
پیاده سازی توپولوژی Tree,3 در Mininet :

```

sudo mn -topo tree,3
[sudo] password for maryam:
*** 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> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>

```

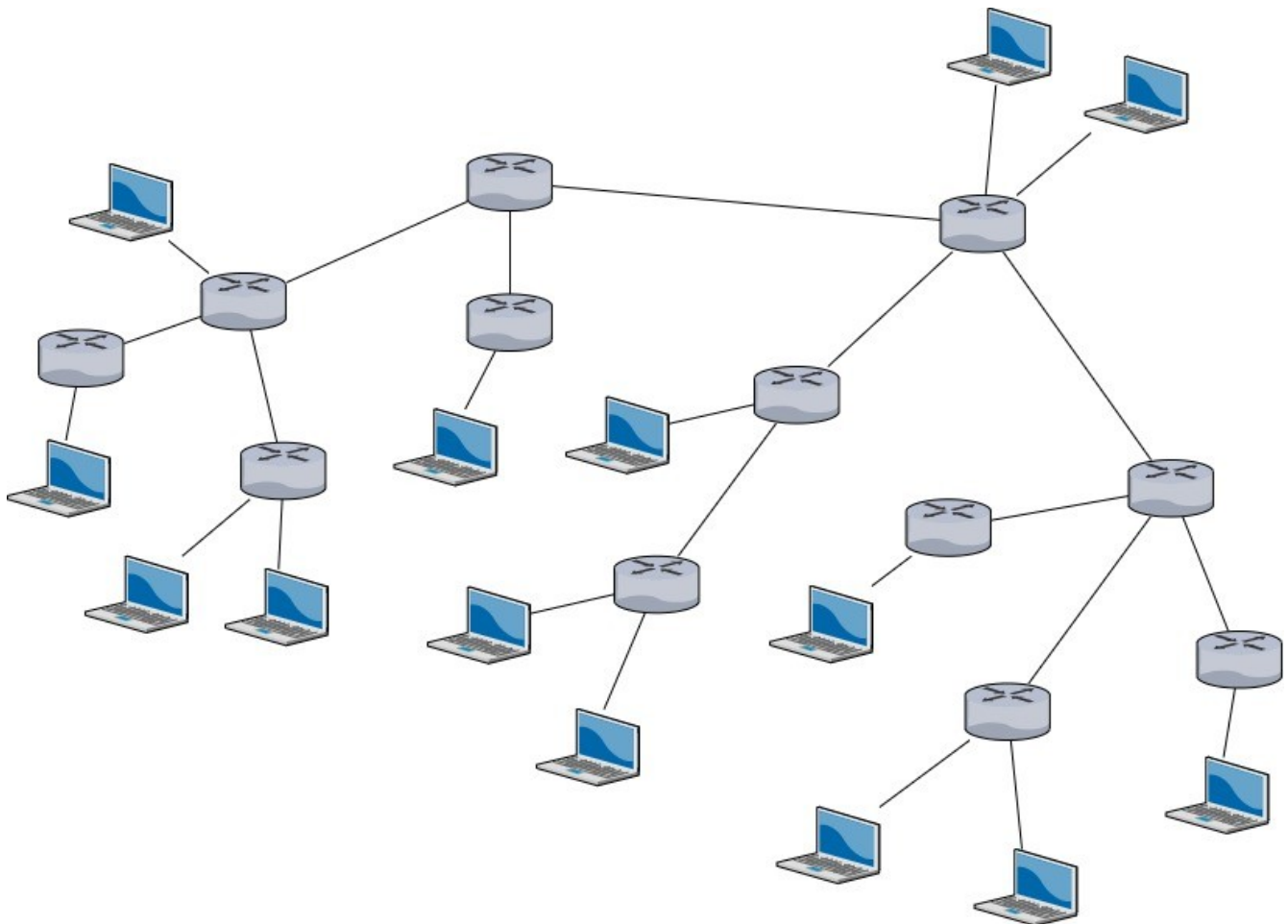
همان طور که در ابتدای گزارش هم مطرح شد ، دو روش برای ایجاد توپولوژی در Mininet داریم که در این قسمت به روش دوم (استفاده از کتابخانه‌های موجود و برنامه‌نویسی با زبان پایتون) می‌پردازیم.
من در ابتدا یک شبکه به فرم زیر طراحی کردم :



اما موقعی که pingall می‌گرفتم تمام پکت‌ها داخل شبکه دراپ می‌شدند.

```
sudo mn --custom Project1_CustomTopo_v1.py --topo sample
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (h7, s7) (h8, s8) (h9, s9) (h10, s10) (h11, s11)
(s1, s2) (s1, s3) (s3, s4) (s4, s5) (s5, s6) (s6, s7) (s7, s8) (s8, s9) (s8, s10) (s9, s10) (s10, s11) (s11, s12) (s12, s13) (s13, s2)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Starting controller
c0
*** Starting 13 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X X X X X
h2 -> X X X X X X X X X
h3 -> X X X X X X X X X
h4 -> X X X X X X X X X
h5 -> X X X X X X X X X
h6 -> X X X X X X X X X
h7 -> X X X X X X X X X
h8 -> X X X X X X X X X
h9 -> X X X X X X X X X
h10 -> X X X X X X X X X
h11 -> X X X X X X X X X
*** Results: 100% dropped (0/110 received)
mininet>
```

که با بررسی و تغییر مدام شبکه طراحی شده ام ، به این نتیجه رسیدم که کنترلر دیفالت Mininet وقتی داخل شبکه لوپ وجود داشته باشه ، نمی‌تونه درست مسیریابی کنه و توی حلقه‌ها گیر می‌کنه. (کد نوشته شده در فایل Project1_CustomTopo_v1.py موجود است) پس یه شبکه جدید بدون حلقه طراحی کردم ، به صورت زیر :



کد نوشته برای راه اندازی این شبکه در فایل پاسخنامه آورده شده است. (فایل Project1_CustomTopo_v2.py)

```
sudo mn --custom Project1_CustomTopo_v2.py --topo sample

*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12
*** Adding links:
(h1, s1) (h2, s1) (h3, s3) (h4, s5) (h5, s6) (h6, s7) (h7, s7) (h8, s8) (h9, s9) (h10, s9) (h11, s10) (h12, s11) (h13, s11) (h14, s12) (s1, s2) (s1, s3) (s1, s4) (s2, s5) (s2, s6) (s3, s7) (s4, s8) (s4, s9) (s4, s10) (s5, s11) (s5, s12)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14
*** Starting controller
c0
*** Starting 12 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 ...
*** Starting CLI:
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)
mininet>
```

کد نوشته شده برای ایجاد شبکه فوق را در ادامه می‌آورم :

```
from mininet.topo import Topo

class CustomTopo( Topo ):
    def __init__( self ):
        Topo.__init__( self )
        #hosts definition
        Host1 = self.addHost( 'h1' )
        Host2 = self.addHost( 'h2' )
        Host3 = self.addHost( 'h3' )
        Host4 = self.addHost( 'h4' )
        Host5 = self.addHost( 'h5' )
        Host6 = self.addHost( 'h6' )
        Host7 = self.addHost( 'h7' )
        Host8 = self.addHost( 'h8' )
        Host9 = self.addHost( 'h9' )
        Host10 = self.addHost( 'h10' )
        Host11 = self.addHost( 'h11' )
        Host12 = self.addHost( 'h12' )
        Host13 = self.addHost( 'h13' )
        Host14 = self.addHost( 'h14' )
```



```

#switches definition
Switch1 = self.addSwitch( 's1' )
Switch2 = self.addSwitch( 's2' )
Switch3 = self.addSwitch( 's3' )
Switch4 = self.addSwitch( 's4' )
Switch5 = self.addSwitch( 's5' )
Switch6 = self.addSwitch( 's6' )
Switch7 = self.addSwitch( 's7' )
Switch8 = self.addSwitch( 's8' )
Switch9 = self.addSwitch( 's9' )
Switch10 = self.addSwitch( 's10' )
Switch11 = self.addSwitch( 's11' )
Switch12 = self.addSwitch( 's12' )

#links definition
self.addLink( Switch1, Switch2 )
self.addLink( Switch1, Switch3 )
self.addLink( Switch1, Switch4 )
self.addLink( Switch2, Switch5 )
self.addLink( Switch2, Switch6 )
self.addLink( Switch3, Switch7 )
self.addLink( Switch4, Switch8 )
self.addLink( Switch4, Switch9 )
self.addLink( Switch4, Switch10 )
self.addLink( Switch5, Switch11 )
self.addLink( Switch5, Switch12 )

self.addLink( Host1, Switch1 )
self.addLink( Host2, Switch1 )
self.addLink( Host3, Switch3 )
self.addLink( Host4, Switch5 )
self.addLink( Host5, Switch6 )
self.addLink( Host6, Switch7 )
self.addLink( Host7, Switch7 )
self.addLink( Host8, Switch8 )
self.addLink( Host9, Switch9 )
self.addLink( Host10, Switch9 )
self.addLink( Host11, Switch10 )
self.addLink( Host12, Switch11 )
self.addLink( Host13, Switch11 )
self.addLink( Host14, Switch12 )

```

```
topos = { 'sample' : ( lambda: CustomTopo() ) }
```

* توجه برای اجرای آن در سیستم خود کافیت دستور زیر را در ترمینال وارد کنید :

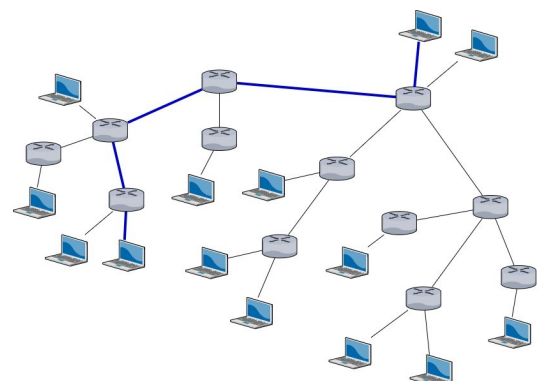
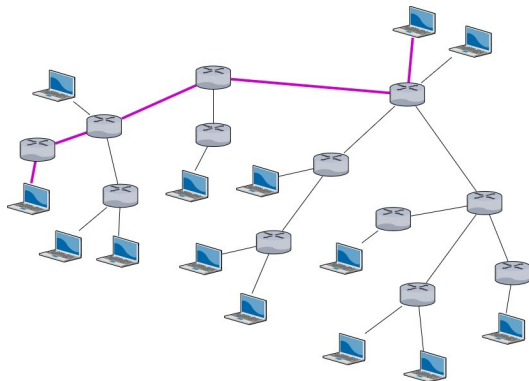
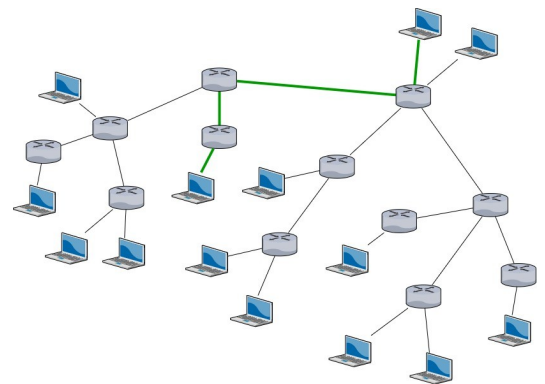
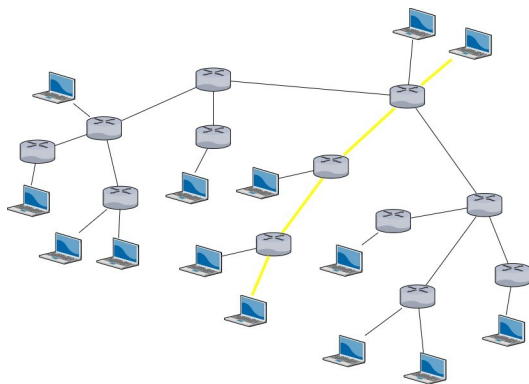
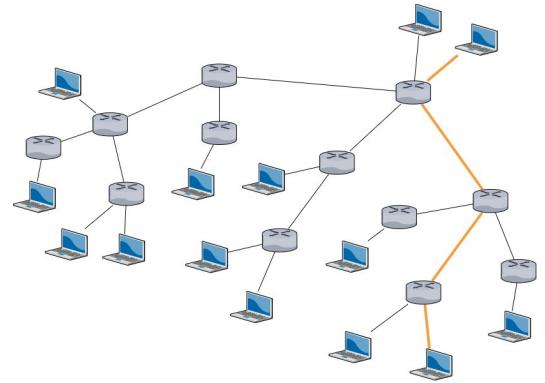
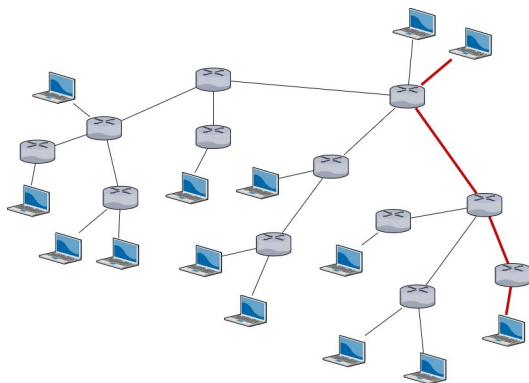
```
~$ sudo mn --custom Project1_CustomTopo_v2.py --topo sample
```

کد واقعا توضیح خاصی نداره ، فقط در همین حد که با استفاده از تابع `addHost()` که تابعی برگرفته از کتابخانه `mininet.topo` است که هاست اضافه می‌کند. تابع `addSwitch()` هم سوئیچ اضافه می‌کند و در نهایت با استفاده از تابع `addLink()` لینک های ارتباطی بین هاست ها و سوئیچ ها را برقرار می‌کنیم. در آخر با استفاده از

قطعه کد `topos = { 'sample' : (lambda: CustomTopo()) }`

شبکه طراحی شده را با نامی مثلا در اینجا "sample" تثبیت می‌کنیم تا بتوانیم آن را به عنوان یک Custom Topology به Mininet معرفی کرد.

در ادامه ۶ مسیر بین هاست های مختلف را آورده ام (با رنگ های مختلف مشخص شده اند) :



در بخش دوم گزارش پروژه اول ، کمی هم درباره **کنترلر Floodlight** تحقیق کردم که نتایج آن به شرح زیر است :

* تاریخچه و کلیات :

کنترلر Floodlight یک کنترلر برای پروتکل OpenFlow است که به زبان جاوا نوشته شده است و تحت لیسانس Apache 2.0 منتشر شده و در حال حاضر در OpenFlowHub که مجمعی از developer های پروژه های OpenFlow است در دسترس می باشد. برای اولین بار در دانشگاه استنفورد و UC Berkeley مطرح شد و اکنون توسط Open source developer ها در حال توسعه است.

کنترلر Floodlight در ابتدا توسط Big Switch بعنوان بخشی از پروژه OpenDaylight ارائه شد ، اما در ژوئن 2013 ، Big Switch ، گویا متناقض با Cisco بود ، کنار رفت در حالی که Floodlight Controller هنوز open-source است ، ولی دیگر با پروژه OpenDaylight درگیر نیست.

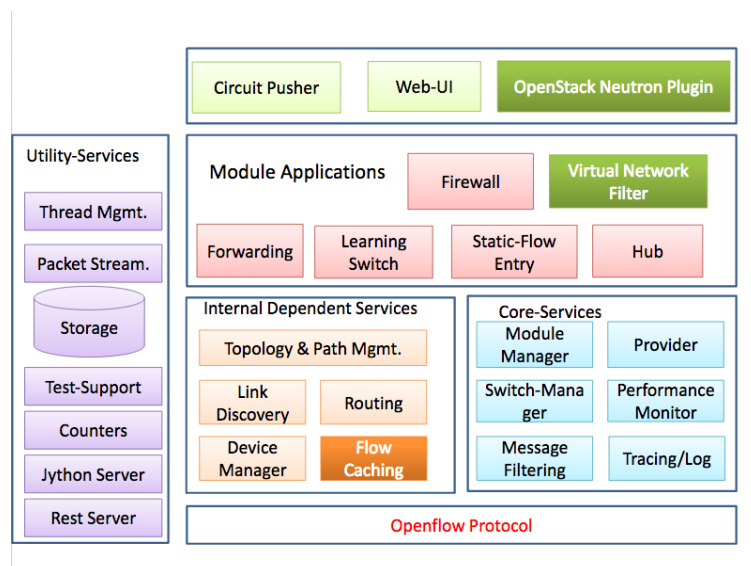
Floodlight Controller می تواند برای توسعه دهندگان سودمند باشد زیرا به آنها امکان سازگاری آسان نرم افزار و توسعه برنامه ها را می دهد. همچنین دارای REST API است که برنامه نویسی رابط با محصول را آسان تر می کند. به علاوه وب سایت Floodlight نمونه های کدگذاری را ارائه می دهد که به توسعه دهندگان در ساخت محصول کمک می کند.

با استفاده از هر دو کلید فیزیکی و مجازی سازگار با Floodlight Controller ، OpenFlow می تواند در محیط های مختلفی کار کند و می تواند با آنچه از قبل مشاغل در اختیار دارند همگام شود. همچنین می تواند از شبکه هایی پشتیبانی کند که گروهی از سوئیچ های سازگار با OpenFlow از طریق سوئیچ های سنتی و غیر OpenFlow به هم متصل می شوند.

کنترلر Floodlight با OpenStack سازگار است ، مجموعه ای از ابزارهای نرم افزاری که به ساخت و مدیریت سیستم عامل های رایانش ابری برای ابرهای عمومی و خصوصی کمک می کند. Floodlight را می توان به عنوان پشتیبان شبکه OpenStack با استفاده از یک پلاگین Neutron که یک مدل شبکه به عنوان سرویس را با API REST که Floodlight ارائه می دهد ، در معرض نمایش قرار داد.

* معماری :

همانطور که در تصویر نیز پیداست ، معماری این کنترلر ماژولار است.



* منبع : لینک