بسمه تعالی

دانشکده ی مهندسی برق و کامپیوتر

دانشگاه صنعتی اصفهان

شبکه‌های کامپیوتری۲ — نیمسال اول ۱۴۰۰-۱۳۹۹

گزارش پروژه شماره یک — تحویل سه‌شنبه ۱۳۹۹/۸/۶

مریم سعیدمهر — ش.د. : ۹۶۲۹۳۷۳

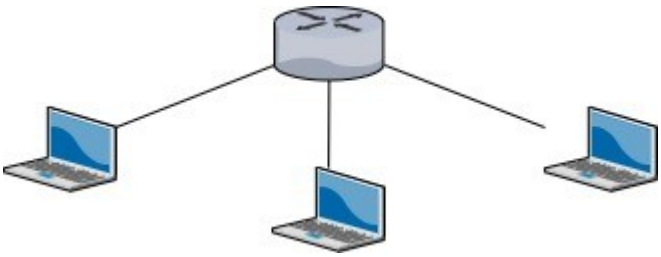
برای ایجاد توپولوژی در Mininet دو راه داریم :

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 .

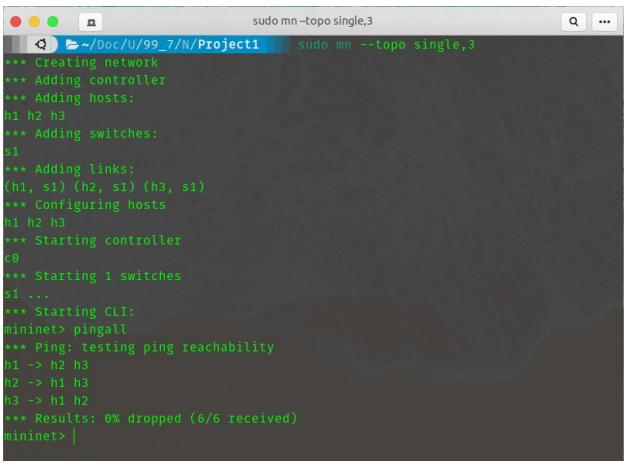
معرفی توپولوژی‌های آماده Mininet :

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 :

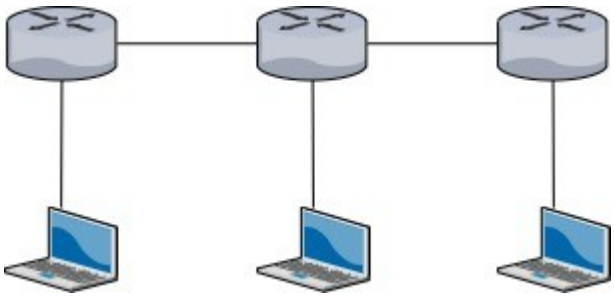


پیاده سازی توپولوژی single,3 در Mininet :

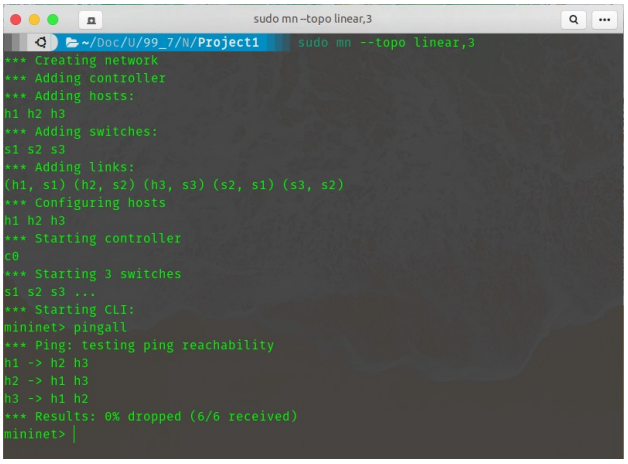


II -1 : توپولوژی Linear :

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 :

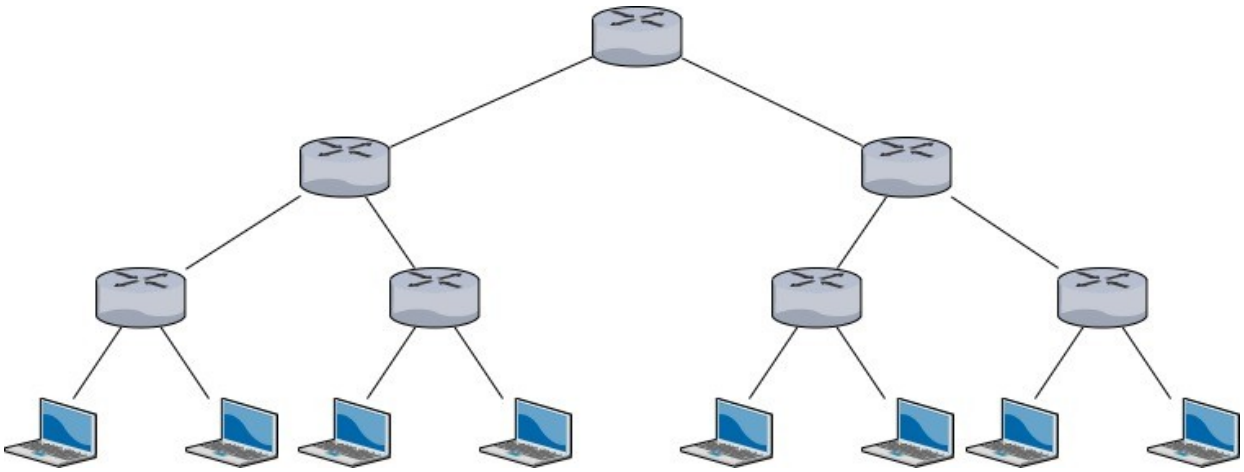


پیاده سازی توپولوژی Linear,3 در Mininet :



III -1 : توپولوژی Tree :

در این توپولوژی ابتدا یک درخت دودویی از سوئیچ‌ها می‌سازیم و سپس به سوئیچ‌های انتهایی(برگ‌های درخت) هاست ها را متصل می‌کنیم. مثلا 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 :

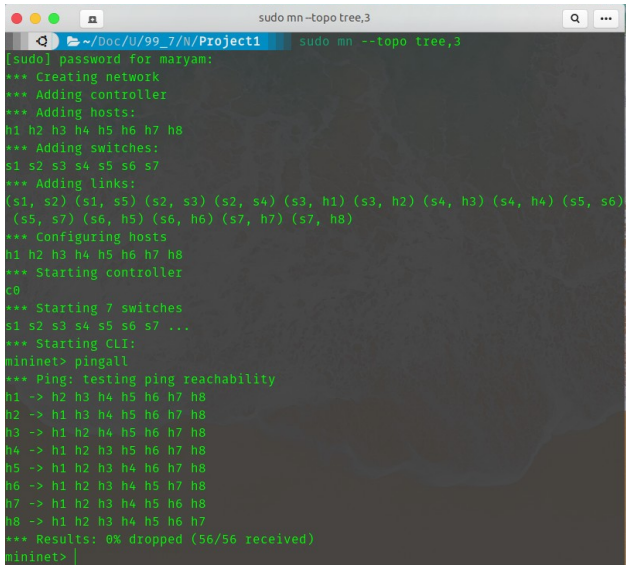


تعداد سوئیچ‌ها در درخت در سطح(L) برابر است با 2^{L-1} . (سطح اول شامل ۱ نود است)

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

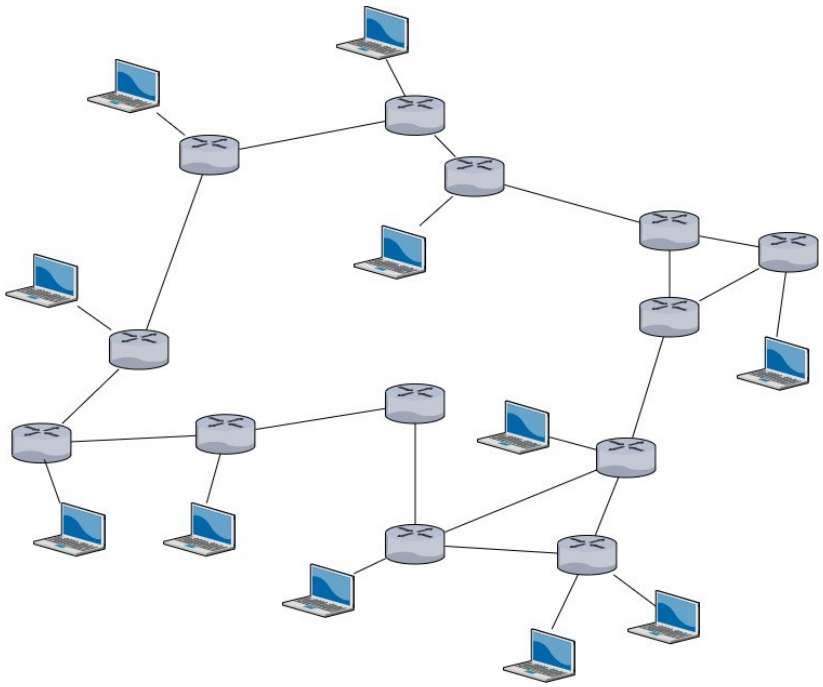
تعداد هاست‌ها در درخت با عمق N برابر است با : 2^N

پیاده سازی توپولوژی Tree,3 در Mininet :

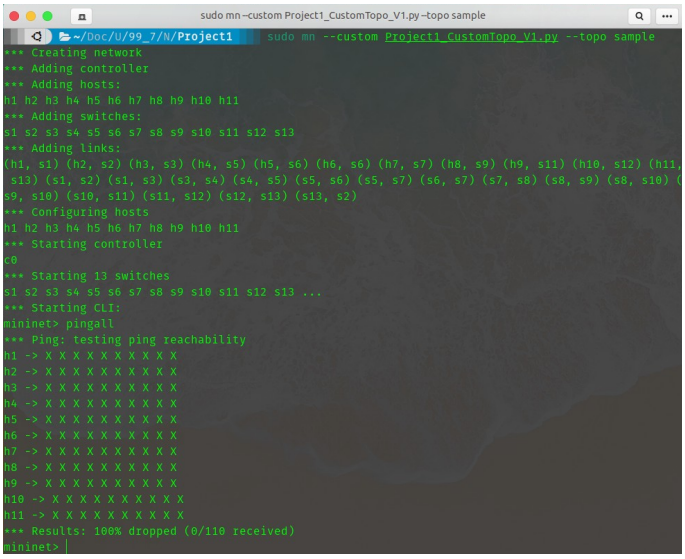


همان طور که در ابتدای گزارش هم مطرح شد ، دو روش برای ایجاد توپولوژی در Mininet داریم که در این قسمت به روش دوم (استفاده از کتابخانه‌های موجود و برنامه‌نویسی با زبان پایتون) می‌پردازیم.

من در ابتدا یک شبکه به فرم زیر طراحی کردم :

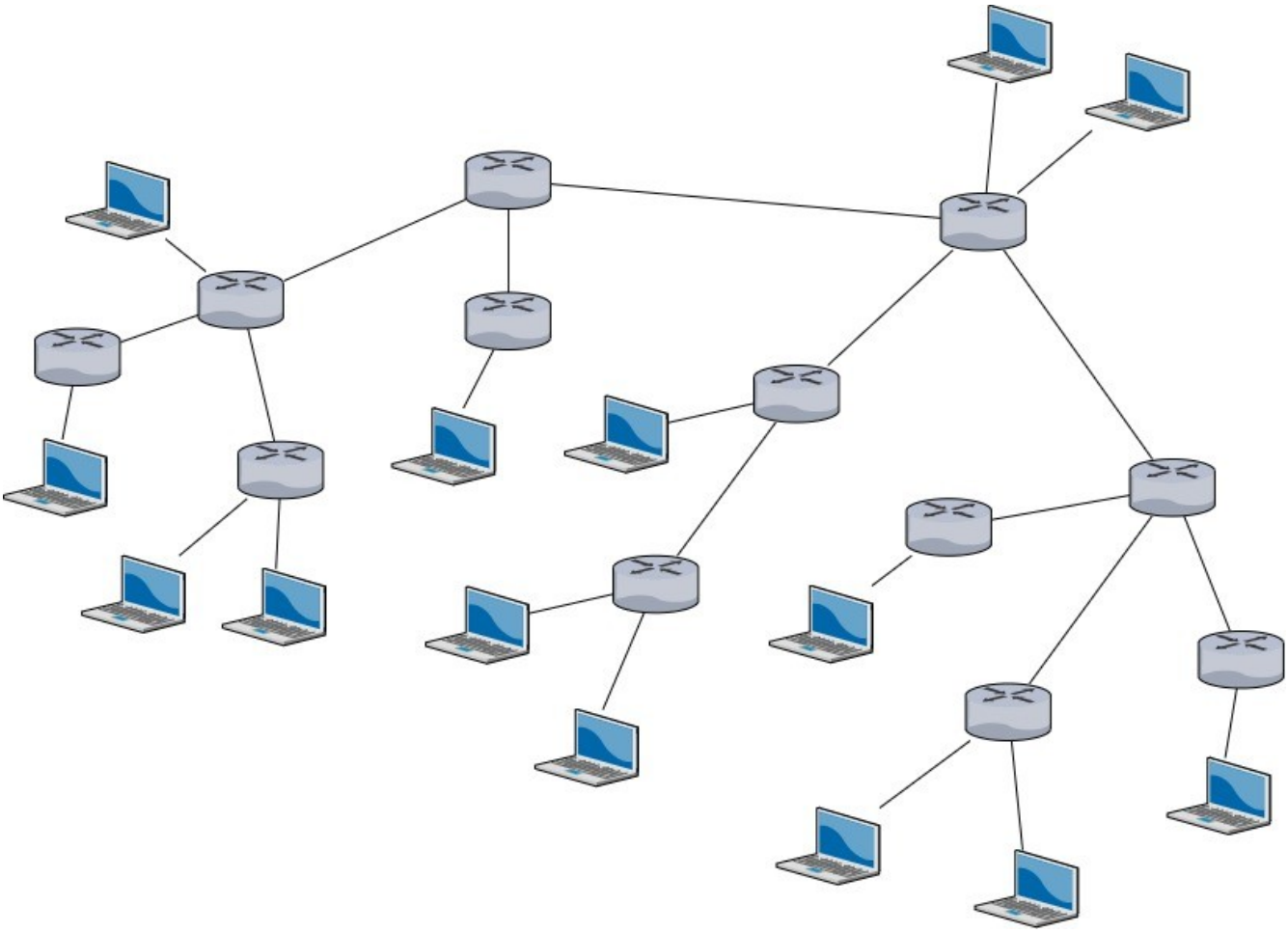


اما موقعی که pingall میگرفتم تمام پکت‌ها داخل شبکه دراپ می‌شدند.

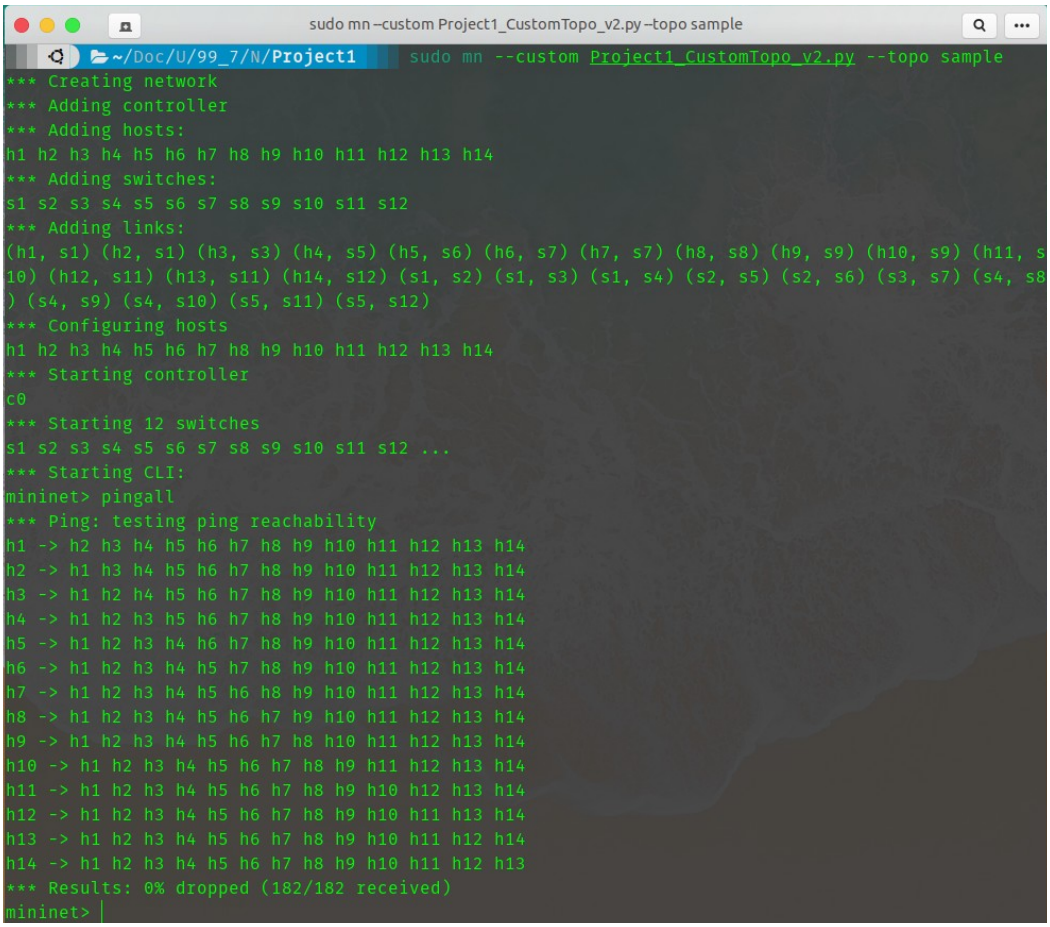


که با بررسی و تغییر مدام شبکه طراحی شده ام ، به این نتیجه رسیدم که کنترلر دیفالت Mininet وقتی داخل شبکه لوپ وجود داشته باشه ، نمی‌تونه درست مسیریابی کنه و توی حلقه‌ها گیر می‌کنه. (کد نوشته شده در فایل Project1_CustomTopo_v1.py موجود است)

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)



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 کافیست دستور زیر را در ترمینال وارد کنید :

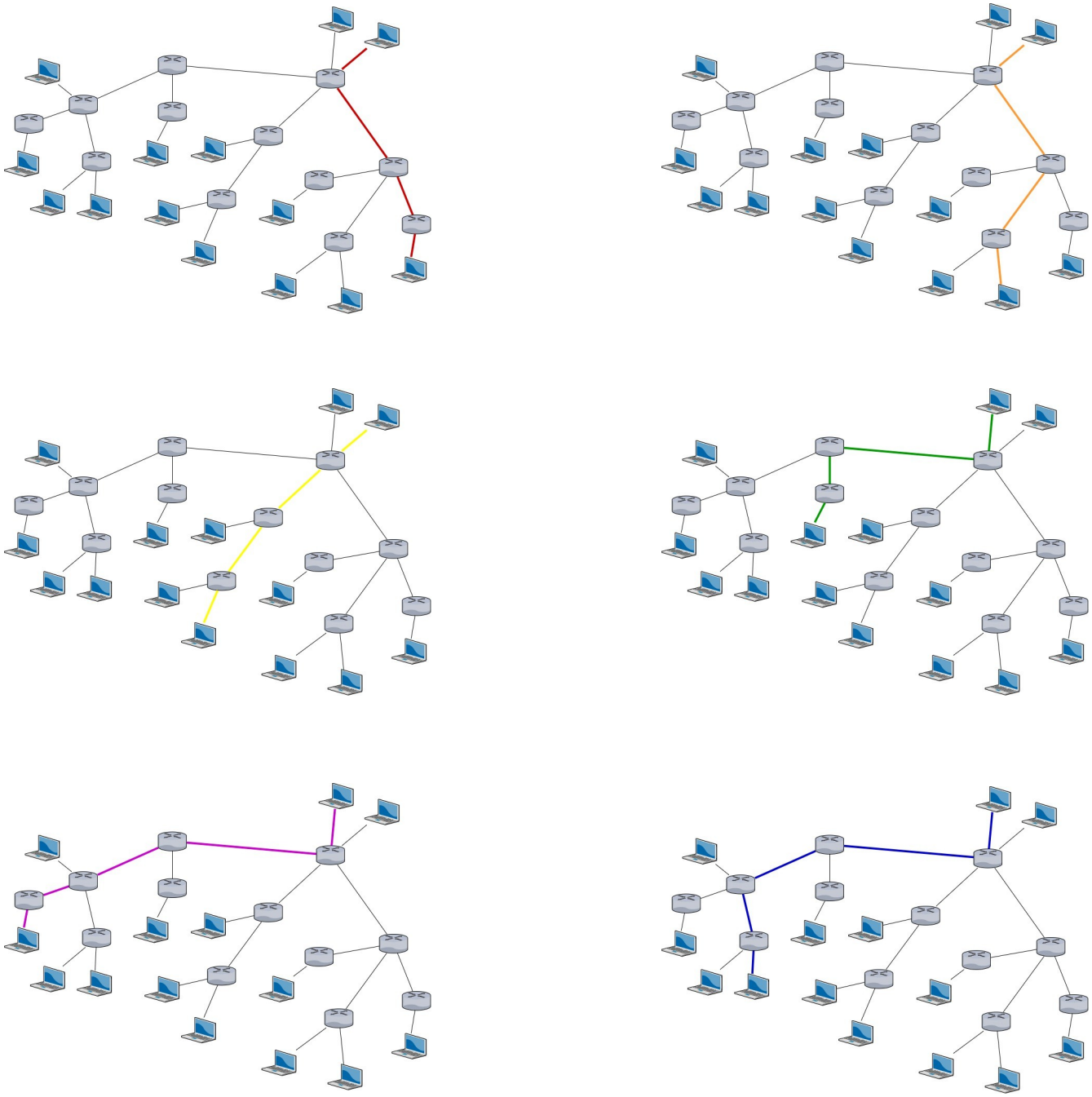
```
~$ sudo mn --custom Project1_CustomTopo_v2.py --topo sample
```

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 هاست‌ها و سوئیچ‌ها 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 هاست‌های 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 پروژه‌های OpenFlow است در دسترس می‌باشد. 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 که گویا متناقض با Cisco بود ، کنار رفت در حالی که Floodlight Controller هنوز open-source است ، ولی دیگر با پروژه OpenDaylight درگیر نیست.

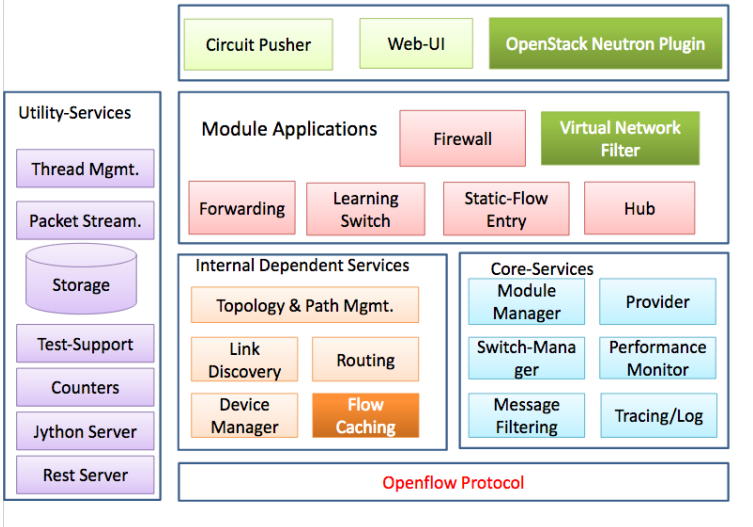
Floodlight Controller می تواند برای توسعه دهندگان سودمند باشد زیرا به آنها امکان سازگاری آسان نرم افزار و توسعه برنامه ها را می دهد. همچنین دارای REST API است که برنامه نویسی رابط با محصول را آسان تر می‌کند. به علاوه وب سایت Floodlight نمونه های کدگذاری را ارائه می دهد که به توسعه دهندگان در ساخت محصول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

با استفاده از هر دو کلید فیزیکی و مجازی سازگار با OpenFlow ، Floodlight Controller می تواند در محیط‌های 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