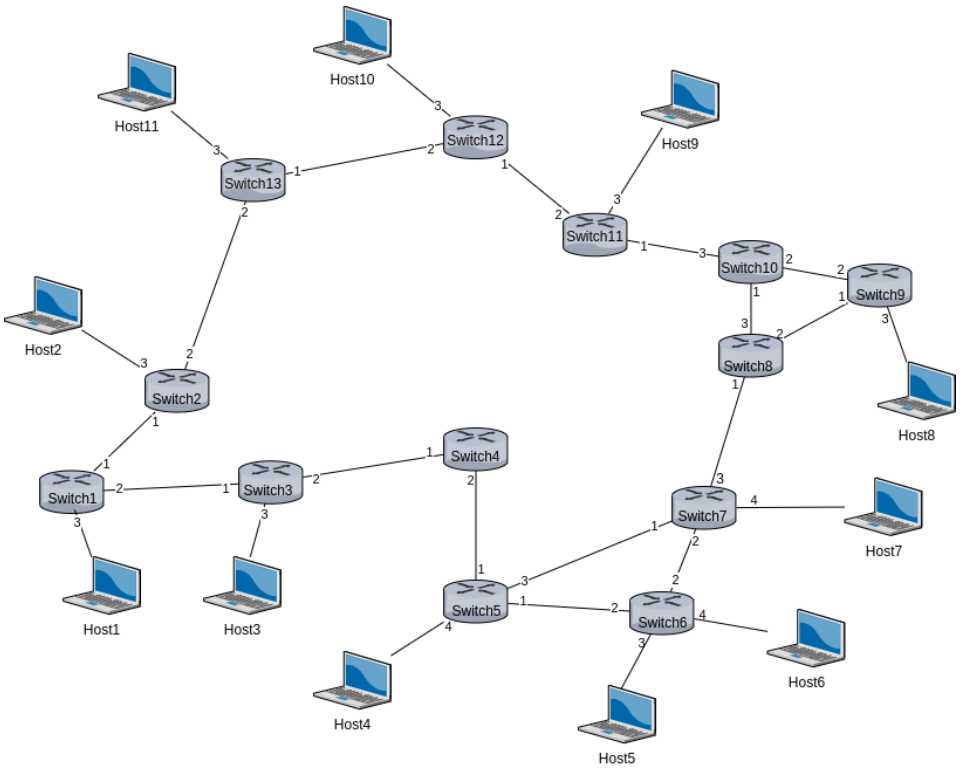
پروژه سوم

مریم سعیدمهر
شماره دانشجویی : ۹۶۲۹۳۷۳

۱ شبکه MPLS

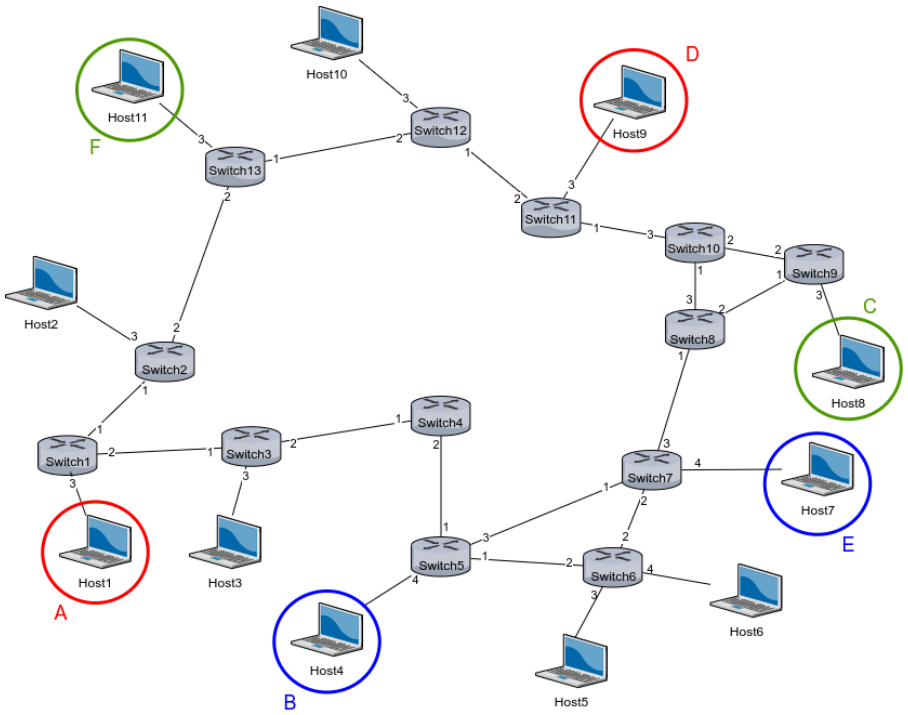
۱.۱ توپولوژی را که در پروژه یک ایجاد کردید در نظر بگیرید

توجه : شبکه‌ای که در پروژه اول ایجاد کردیم لاجرم درختی بوده زیرا کنترلر فلودلایت نداشتیم پس باتوجه به درختی بودن آن ، بین هر دو هاست فقط و فقط یک مسیر وجود داشته و لذا نمیتوان دو تونل با مسیرهای متفاوت تعریف کرد!
من در همان پروژه اول ، دو شبکه ایجاد کردم که اولی به دلیل داشتن دور مشکل داشت و ناچاراً شبکه دوم که درختی بود را طراحی و پیاده سازی کردم. لذا الان آن شبکه دارای دور را در اینجا استفاده میکنم. توپولوژی آن در شکل ۱ آمده است.



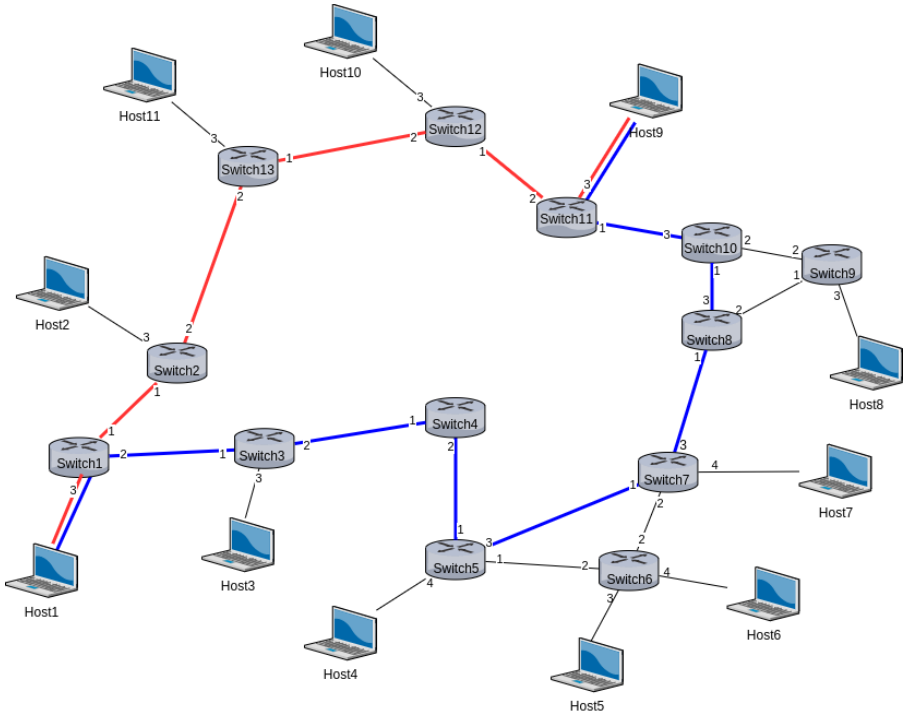
شکل ۱: توپولوژی شبکه مورد نظر

✓ مبدأهای A،B،C و مقصدهای D،E،F در شکل ۲ مشخص شده‌اند



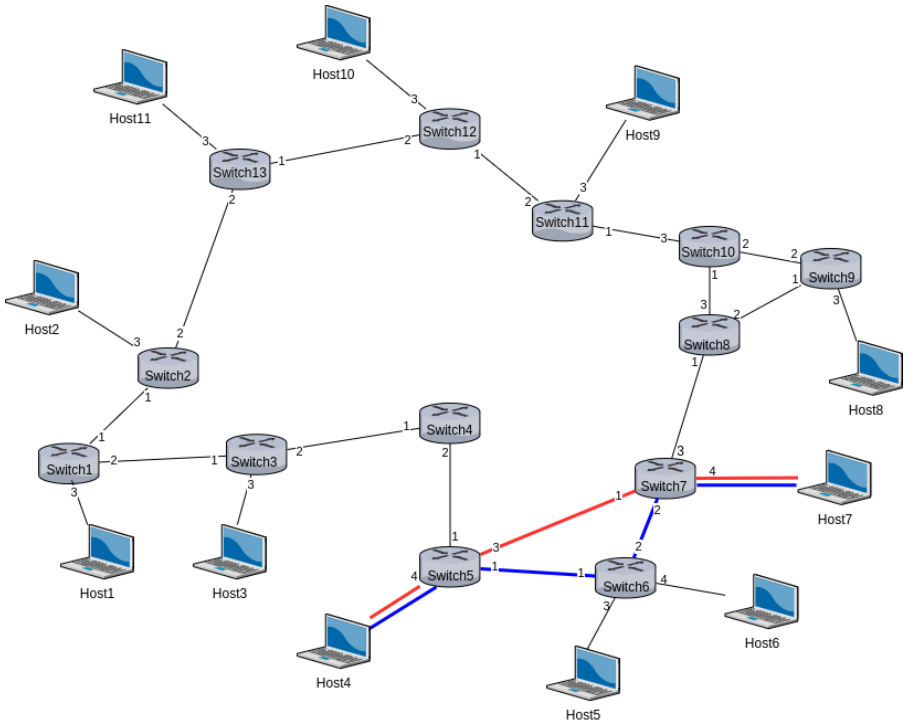
شکل ۲: مبدأها و مقصدها

✓ درشکل ۳ برای مبدا A و مقصد D دو تونل مشخص شده است.



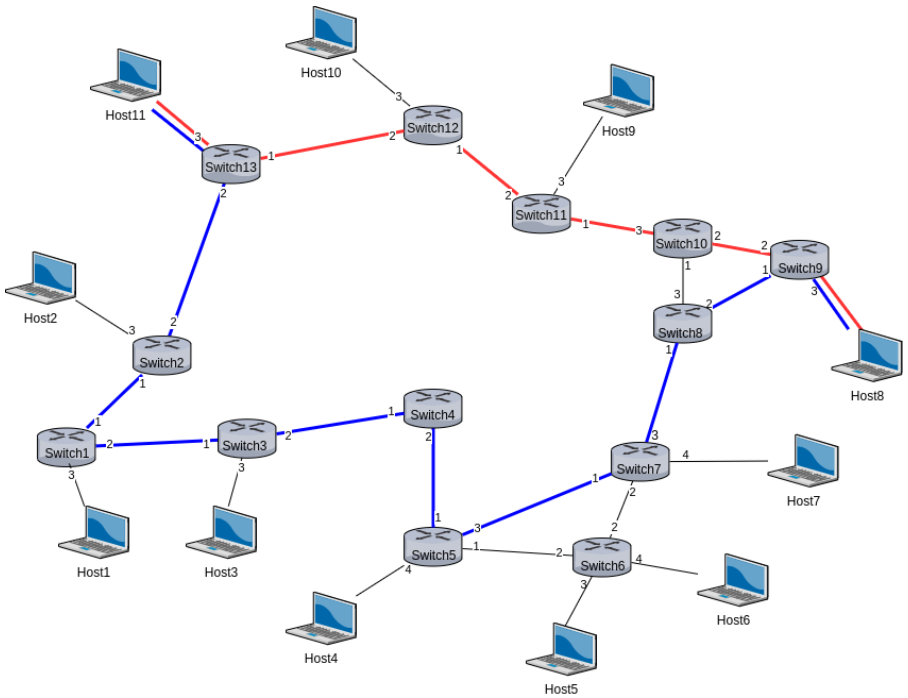
شکل ۳: دو تونل برای مبدا A و مقصد D

✓ درشکل ۴ برای مبدا B و مقصد E دو تونل مشخص شده است.



شکل ۴: دو تونل برای مبدا B و مقصد E

✓ درشکل ۵ برای مبدا C و مقصد F دو تونل مشخص شده است.



شکل ۵: دو تونل برای مبدا C و مقصد F

برای ایجاد entry های لازم از زبان پایتون استفاده شده است. کدها در پاسخنامه آپلود شده در سامانه ضمیمه شده است.

۱. فایل $Project1\_CustomTopo\_V1.py$ همان فایلی است که در پروژه اول نیز آپلود کرده ام و برای ایجاد توپولوژی مذکور است.
۲. فایل $entity.py$ برای ایجاد entry های لازم است.

۱.۱.۱ دستورات لازم برای اجرای کدها

۱. به کمک دستور شکل ۶ سرور کنترلر فلودلایت را در حالت run بگذارید.

```
~/floodlight / master
java -jar target/floodlight.jar
```

شکل ۶: فعال کردن کنترلر فلودلایت

۲. به کمک دستور شکل ۷ توپولوژی را در مینی‌نت ایجاد کرده و کنترلر فلودلایت را نیز به آن متصل کنید.

```
~
sudo mn --custom Project1_CustomTopo_V1.py --topo sample --controller=remote,ip=127.0.0.1,port=6653
```

شکل ۷: دستور ایجاد توپولوژی در مینی‌نت

۳. به کمک دستور شکل ۸ entry های لازم را به شبکه اضافه کنید

```
~
python entity.py
```

شکل ۸: اضافه کردن entry ها به شبکه

۴. در محیط مینی‌نت برای پینگ گرفتن باید از فلگ $-Q$ و سپس TOS مناسب استفاده کنیم.

۲.۱ اعمال اولویت بین دو تونل با یک مقصد

همانطور که در تصویر میبینید ، میتوان از فیلد priority برای این منظور استفاده کرد. مثلاً برای جریانی که از مبدأ A به مقصد D میرود و از تونل ۱ می‌آید در هر سوییچ مشترک مقدار priority را 32767 قرار دهیم و برای جزیان ورودی از تونل ۲ برای هر سوییچ مشترک از مسیر ، مقدار priority را کمتر از 32767 ست کنیم.

۲ کنترلر OpenDayLight(ODL)

کنترلر OpenDaylight نرم افزار JVM است و به شرط پشتیبانی از جاوا از هر سیستم عامل و سخت افزاری قابل اجرا است. کنترل کننده پیاده سازی مفهوم Network Defined Network (SDN) است و از ابزارهای زیر استفاده می کند:

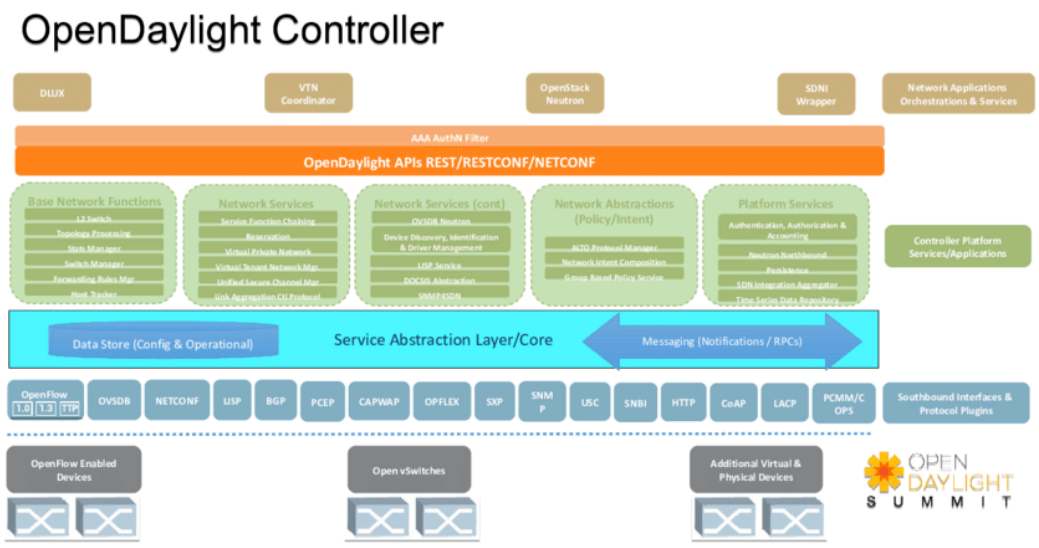
- Maven : کنترلر OpenDaylight از Maven برای اتوماسیون ساخت آسان تر استفاده می کند. Maven از [۱] pom.xml برای نوشتن وابستگی های بین بسته نرم افزاری و همچنین توصیف اینکه بسته های نرم افزاری بارگیری و شروع می شود استفاده می کند.
- OSGi : این چارچوب قسمت آخر OpenDaylight است زیرا امکان بارگذاری پویا بسته ها و بسته های پرونده های JAR را فراهم می کند و بسته های نرم افزاری را برای تبادل اطلاعات بهم متصل می کند.

[۱] $(Project\ Object\ Model)$

- رابط های JAVA : رابط های جاوا برای گوش دادن به رویدادها ، مشخصات و الگوهای شکل گیری استفاده می شوند. این روش اصلی است که در آن بسته های خاص عملکردهای برگشت تماس را برای رویدادها و همچنین نشان دادن آگاهی از حالت خاص اجرا می کنند.
- REST API : اینها API های شمال هستند مانند مدیر توپولوژی ، ردیاب میزبان ، برنامه ریز جریان ، مسیریابی ثابت و غیره.

این کنترل کننده API های باز شده به سمت شمال را که توسط برنامه ها استفاده می شود ، در معرض دید قرار می دهد. چارچوب OSGi و REST دو طرفه برای رابط های برنامه کاربردی شمال صحرا پشتیبانی می شود. چارچوب OSGi برای برنامه هایی که در همان فضای آدرس با کنترلر اجرا می شوند در حالی که REST API (مبتنی بر وب) برای برنامه هایی که در همان فضای آدرس (یا حتی همان سیستم) کنترل کننده اجرا نمی شوند ، استفاده می شود. منطق و الگوریتم های تجاری در برنامه ها قرار دارند. این برنامه ها از کنترل کننده برای جمع آوری هوش شبکه ، اجرای الگوریتم آن برای انجام تجزیه و تحلیل و سپس تنظیم قوانین جدید در سراسر شبکه استفاده می کنند. در جنوب ، چندین پروتکل به عنوان پلاگین پشتیبانی می شوند ، به عنوان مثال $OpenFlow\ 1.0$ ، $OpenFlow\ 1.3$ ، $BGP-LS$ و غیره. کنترلر OpenDaylight با یک پلاگین $OpenFlow\ 1.0$ $Southbound$ شروع می شود. سایر همکاران OpenDaylight شروع به افزودن کد کد کنترل می کنند. این ماژول ها به صورت پویا به یک لایه انتزاع خدمات (SAL) متصل می شوند.

SAL خدماتی را که ماژولهای شمالی آن نوشته شده است ، در معرض دید قرار می دهد. SAL چگونگی تحقق سرویس درخواستی را بدون توجه به پروتکل اساسی استفاده شده بین کنترل کننده و دستگاه های شبکه مشخص می کند. این باعث می شود که از برنامه های کاربردی محافظت شود زیرا OpenFlow و پروتکل های دیگر با گذشت زمان تکامل می یابند. برای اینکه کنترل کننده بتواند دستگاه ها را در دامنه خود کنترل کند ، باید در مورد دستگاه ها ، قابلیت های آنها ، قابلیت دسترسی و غیره اطلاعاتی کسب کند. این اطلاعات توسط Topology Manager ذخیره و مدیریت می شود. سایر مولفه ها مانند Tracker ARP handler، Manager Host، Device و Switch Manager در تولید پایگاه داده توپولوژی برای Topology Manager کمک می کنند.
معماری این کنترلر در شکل ۹ آمده است.



شکل ۹: معماری کنترلر ODL

۳ ویدیو

ویدیو توضیحات را در https://iutbox.iut.ac.ir/index.php/s/YGbGEaA۲kepfdyM مشاهده کنید.