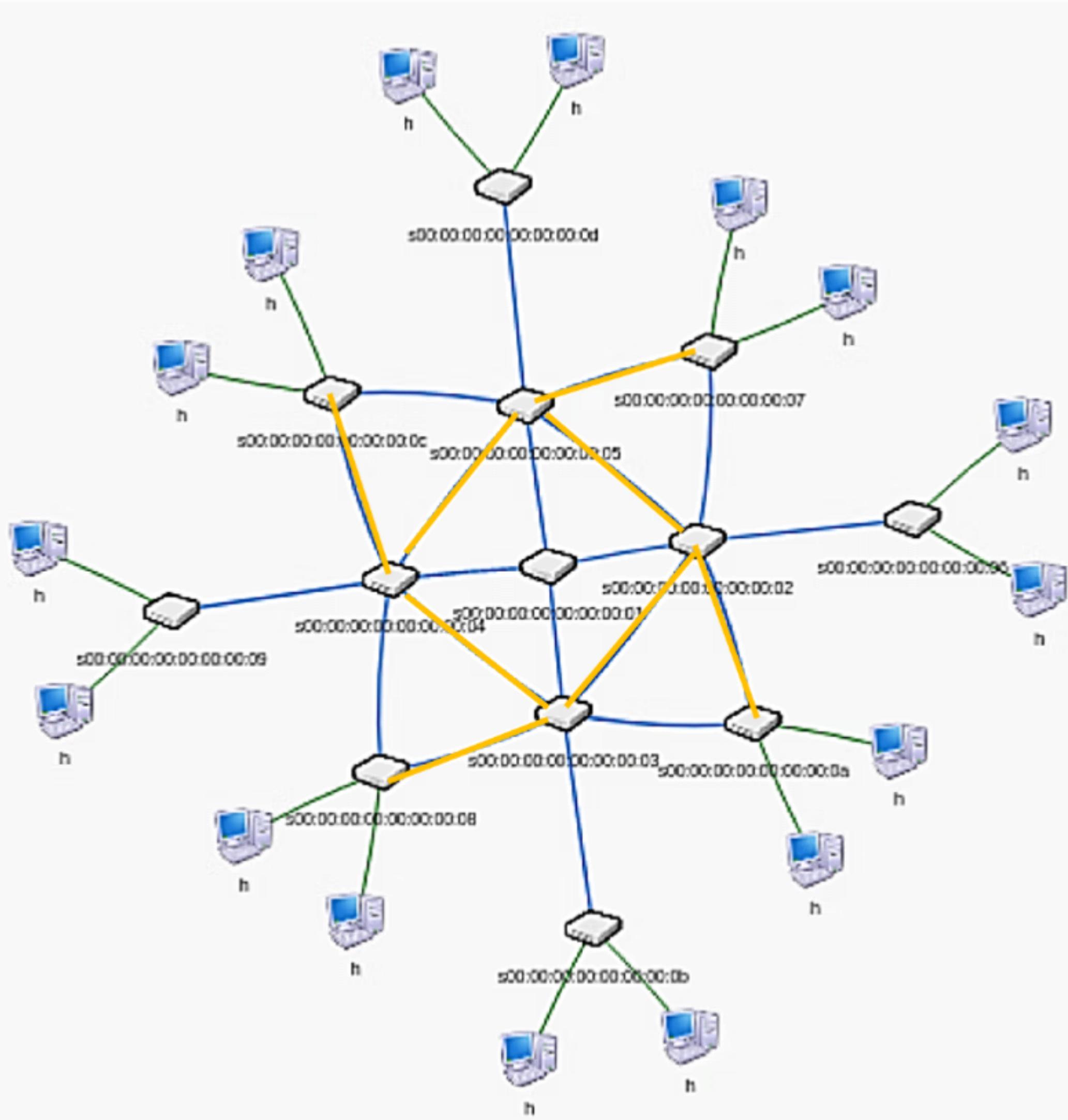


پروژه‌ی سوم شبکه

حدیث احمدیان ۹۶۲۲۶۱۳

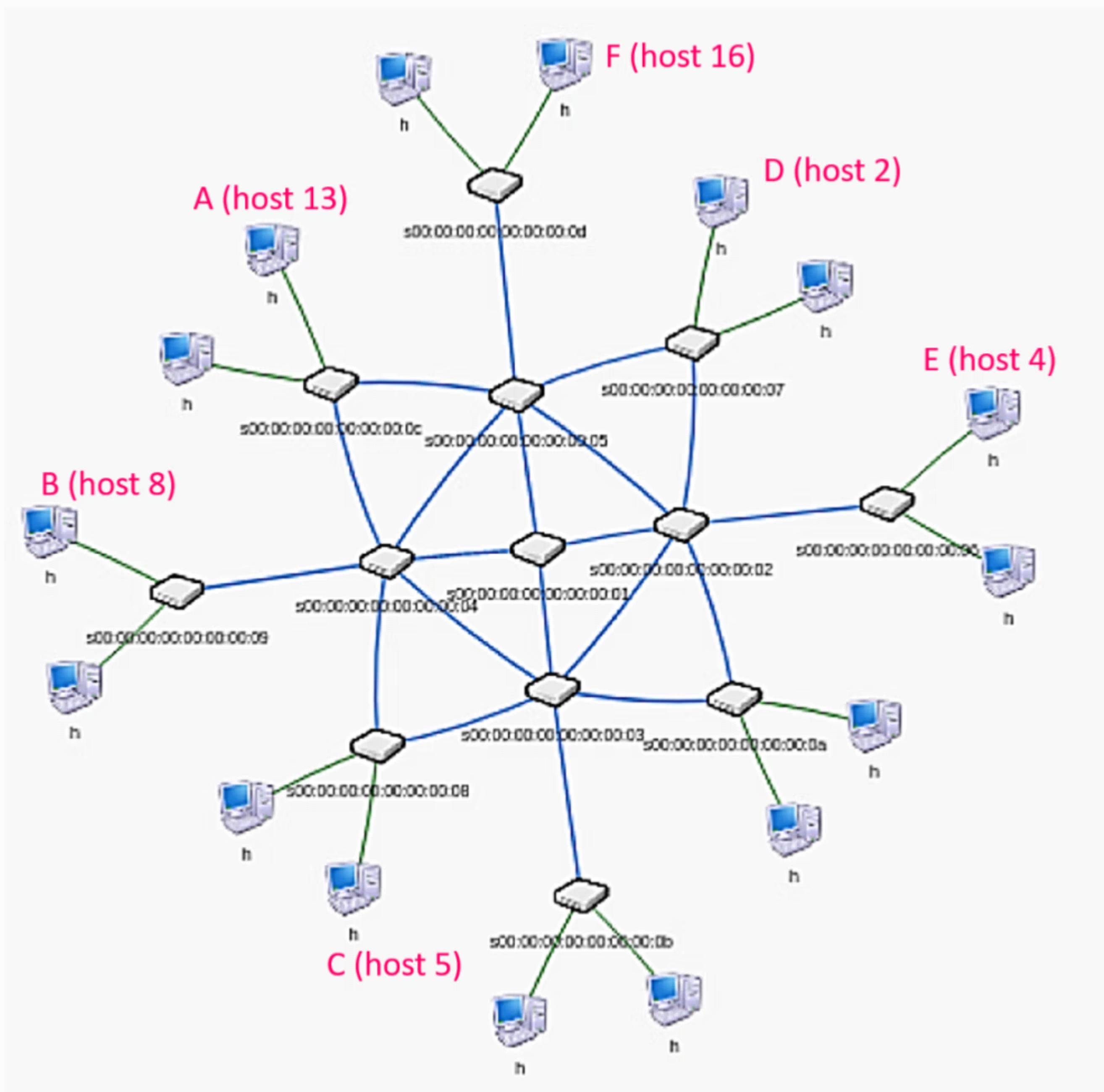
سوال ۱ :

(۱.۱) به دلیل اینکه در توپولوژی طراحی شده در پروژه یک امکان ایجاد تونل وجود نداشت، با افزودن چند یال به توپولوژی پروژه ۱ و تغییر آن، توپولوژی زیر را برای این پروژه استفاده کردم :

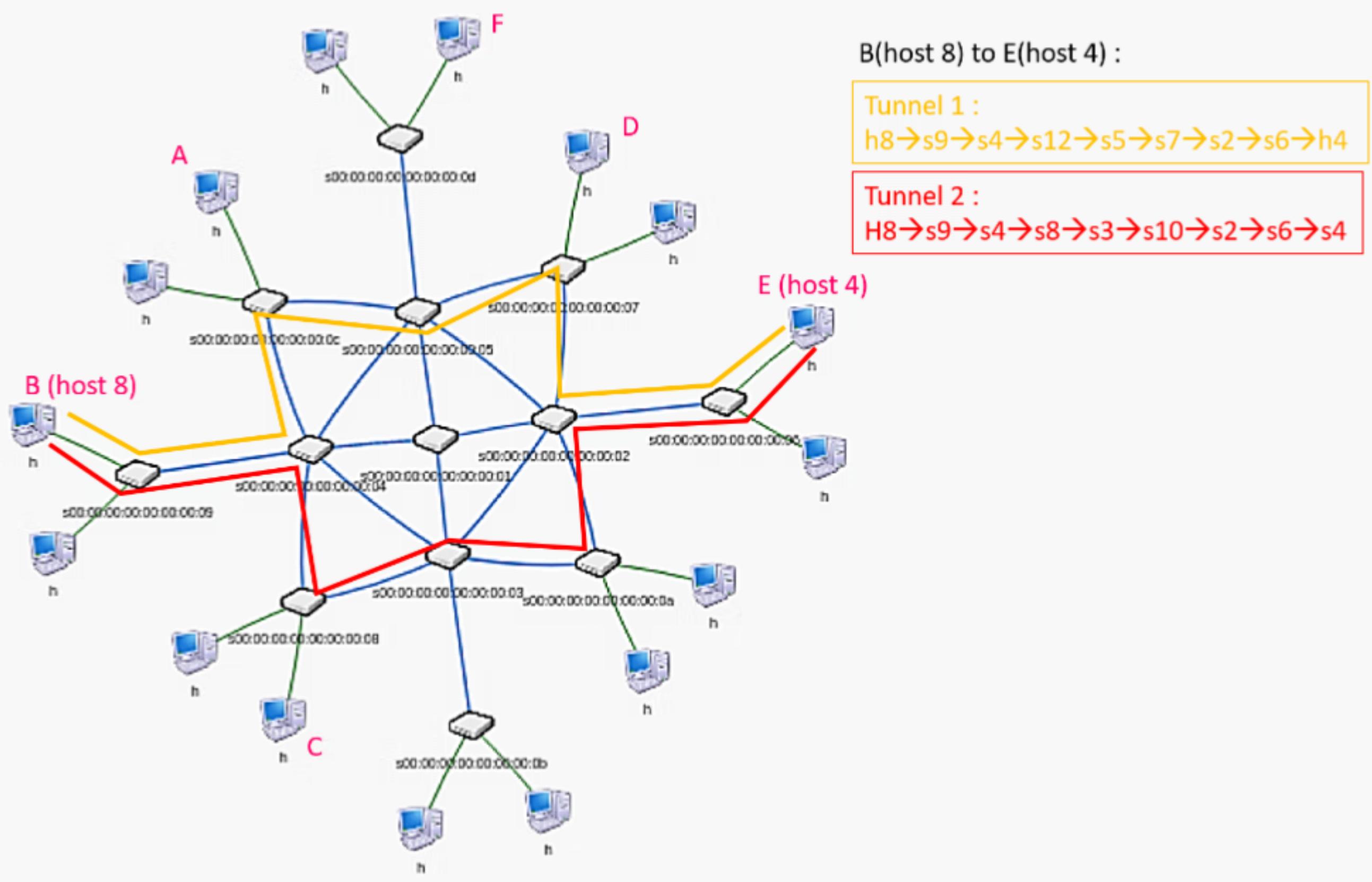
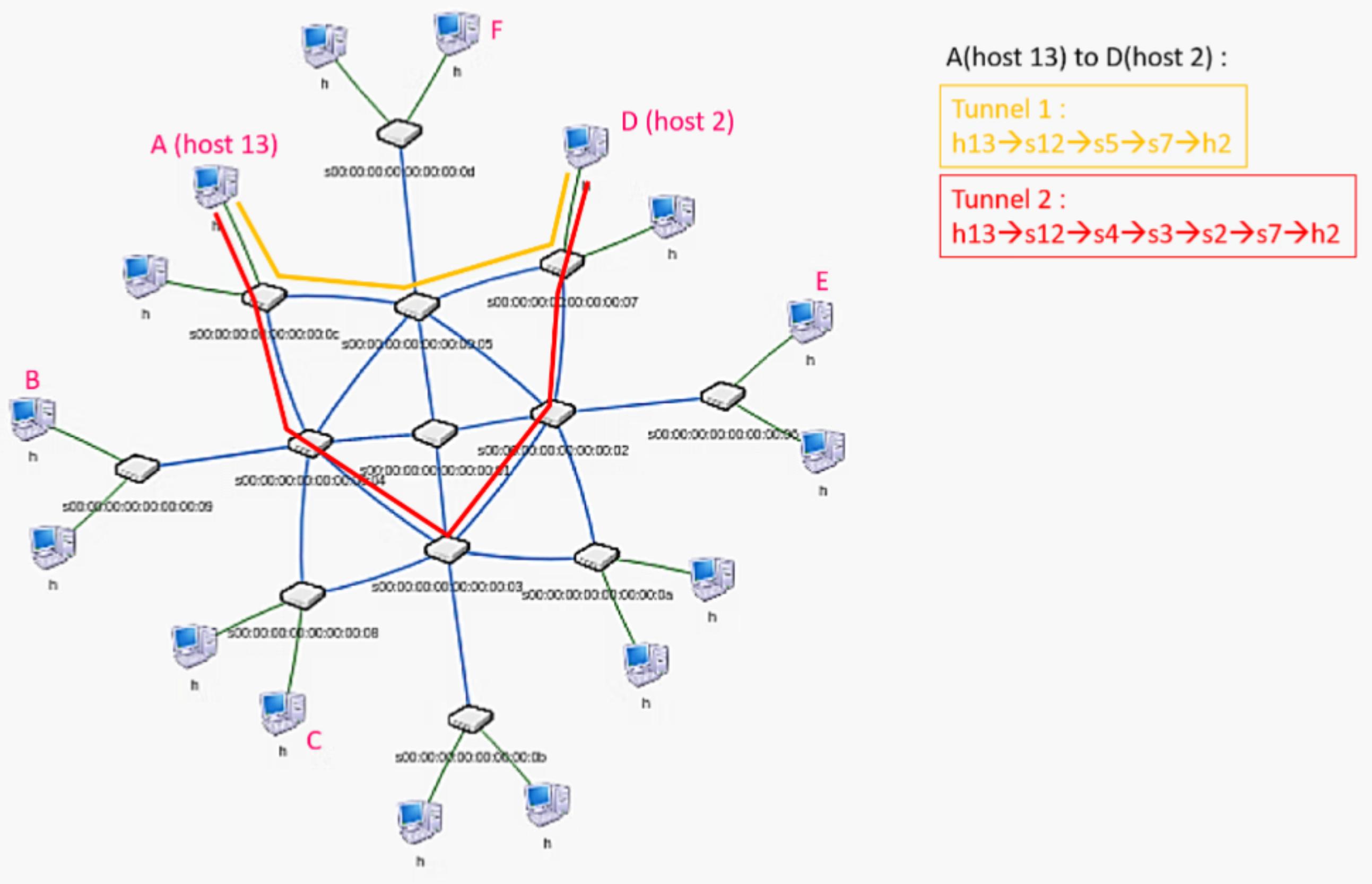


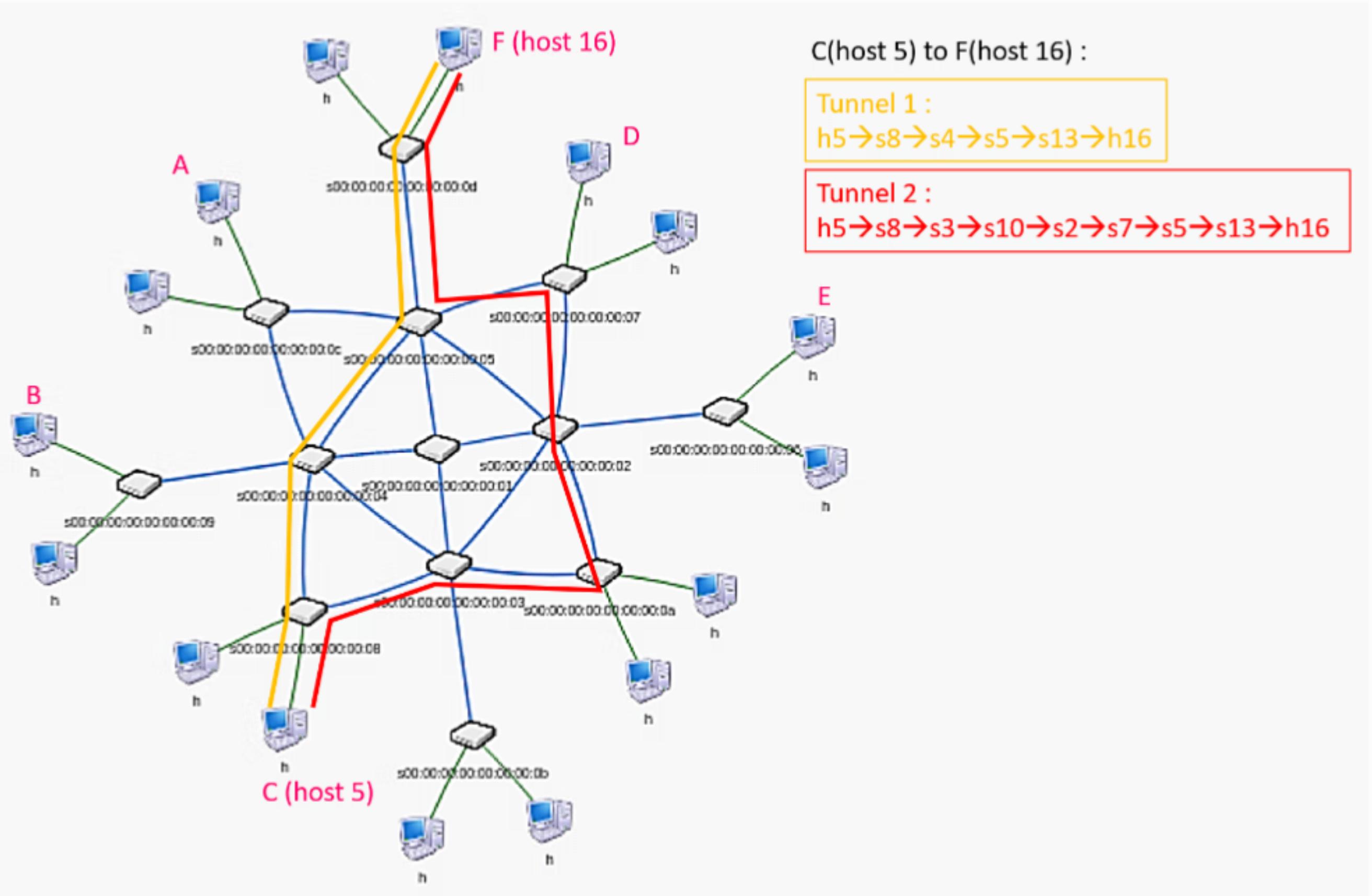
(یال های زرد به توپولوژی پروژه ۱ اضافه شده اند)
کد لازم برای ساخت این توپولوژی در فایل topo.py
ضمیمه شده است.

شش میزبان از توپولوژی به شکل زیر انتخاب شده اند:



تونل های تعریف شده :





همانطور که قابل مشاهده است بین هر مبدأ و مقصد دو تونل طراحی شده، در هنگام اجرا برای اینکه برای ping بین مبدأها و مقصد ها، مشخص کنیم کدام تونل مدنظر ماست، باید از فیلد TOS استفاده کنیم پس به طور کلی در کد، برای هر مبدأ و مقصد فیلد TOS برای تونل اول ۱ و برای تونل دوم ۲ تنظیم شده است. رویکرد ایجاد این تونل ها به این صورت است که در اولین روتر هدر mpls را در بسته ها میکنیم و با توجه به آیپی مبدأ و مقصد و TOS یک mpls lable برای آن در نظر میگیریم در بقیه مسیر

دیگر با توجه به این لیبل و پورت ورودی روتراست که مسیریابی میکنیم و دیگر آیپی مبدا و مقصد را چک نمیکنیم و لیبل بسته را تنظیم کرده و به پورت خروجی مناسب میفرستیم . و در آخرین روترا هم هدر mpls را pop میکنیم.

کد لازم برای ایجاد تونل ها در فایل `mpls.py` ضمیمه شده است.

دستورات لازم برای اجرا
اجرای کنترلر فلودلایت:

```
f@ubuntu:~/floodlight$ java -jar target/floodlight.jar
```

ساخت توپولوژی و تنظیم فلودلایت به عنوان کنترلر :

```
f@ubuntu:~/Desktop/prj3$ sudo mn --custom topo.py --topo mytopo --controller=remote,ip=127.0.0.1,port=6653
```

وارد کردن entry های لازم به سوییچ ها:

```
f@ubuntu:~/Desktop/prj3$ python mpls.py
```

دستور ping بین دو میزبان همراه با فیلد tos :

```
mininet> h5 ping h16 -Q 1
```

با استفاده از-Q و وارد کردن tos مورد نظر روبروی آن، بسته از تونل مورد نظر ما پینگ خواهد شد.

نحوه اجرا در ویدیو به طور کامل توضیح داده شده است و وجود ping و بررسی مسیریابی درست و عبور بسته ها از تونل های مورد نظر بین یک مبدأ و مقصد در ویدیو نمایش داده شده است.

لینک ویدیو :

<https://iutbox.iut.ac.ir/index.php/s/LZqDe7MG3YESa86>
برای سهولت در باز کردن لینک فایل **video_link.txt** ضمیمه شده و لینک در آن آورده شده است.

(۱.۲) در هر یک از فلوهایی که در سویچ ها وارد میکنیم یک فیلد به نام priority وجود دارد، اگر فلوهای مربوط به تونل اول را بالاتر از فلوهای مربوط به تونل دوم تنظیم کنیم، بین بسته های دو تونل در نود های مشترک در مسیر، به بسته های تونل اول اولویت بیشتری داده میشود.

سوال ۲ :

یکی از موفقیت آمیز ترین روش های توسعه‌ی نرم افزار، ماژولار است که در آن ادغام ماژول‌های به هم پیوسته یک برنامه‌ی بزرگ را تشکیل می‌دهد. OSGI یا Open Services Gateway Initiative یک سیستم ماژول داینامیک برای جاواست که یک معماری ماژولار برای توسعه‌ی برنامه‌ها را تعریف می‌کند. کنترلر ODL یا به اختصار openDayLight است نیز بر مبنای معماری OSGI ساخته شده است. ODL یک پروژه‌ی مشترک open source است. که تمرکز آن بر ساخت یک اکوسیستم multi-vendor و multi-project می‌باشد.

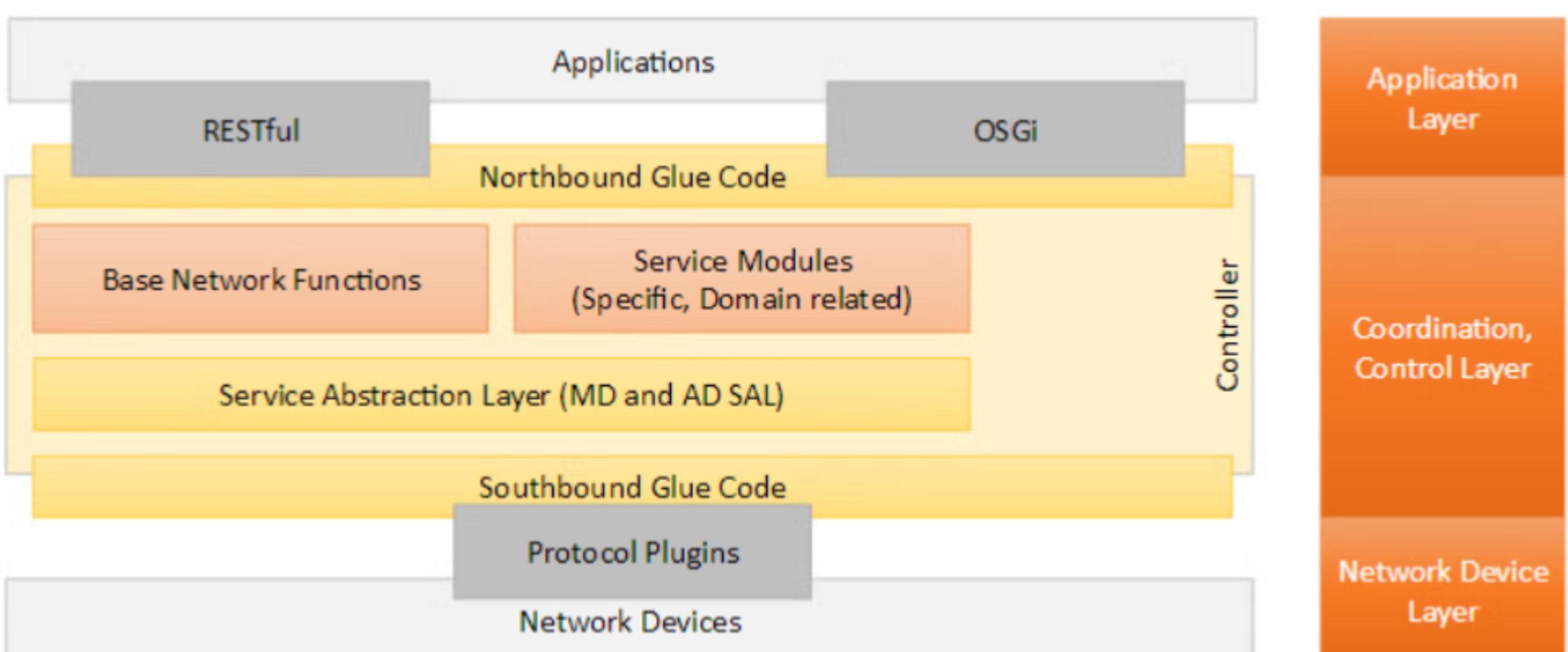
از ویژگی‌های ODL میتوان به قابلیت برنامه‌ریزی با پروتکل‌های northbound APIs و southbound programmable network service اپلیکیشنها و اشاره کرد. و هم چنین در مقایسه با سایر کنترلرها ODL کامل‌ترین documentation را دارد.

میتوان گفت که ODL تا کنون موفق ترین کنترلر open source بوده است و از آن به عنوان شکل دهنده‌ی آینده‌ی SDN یاد می‌شود چون به دلیل

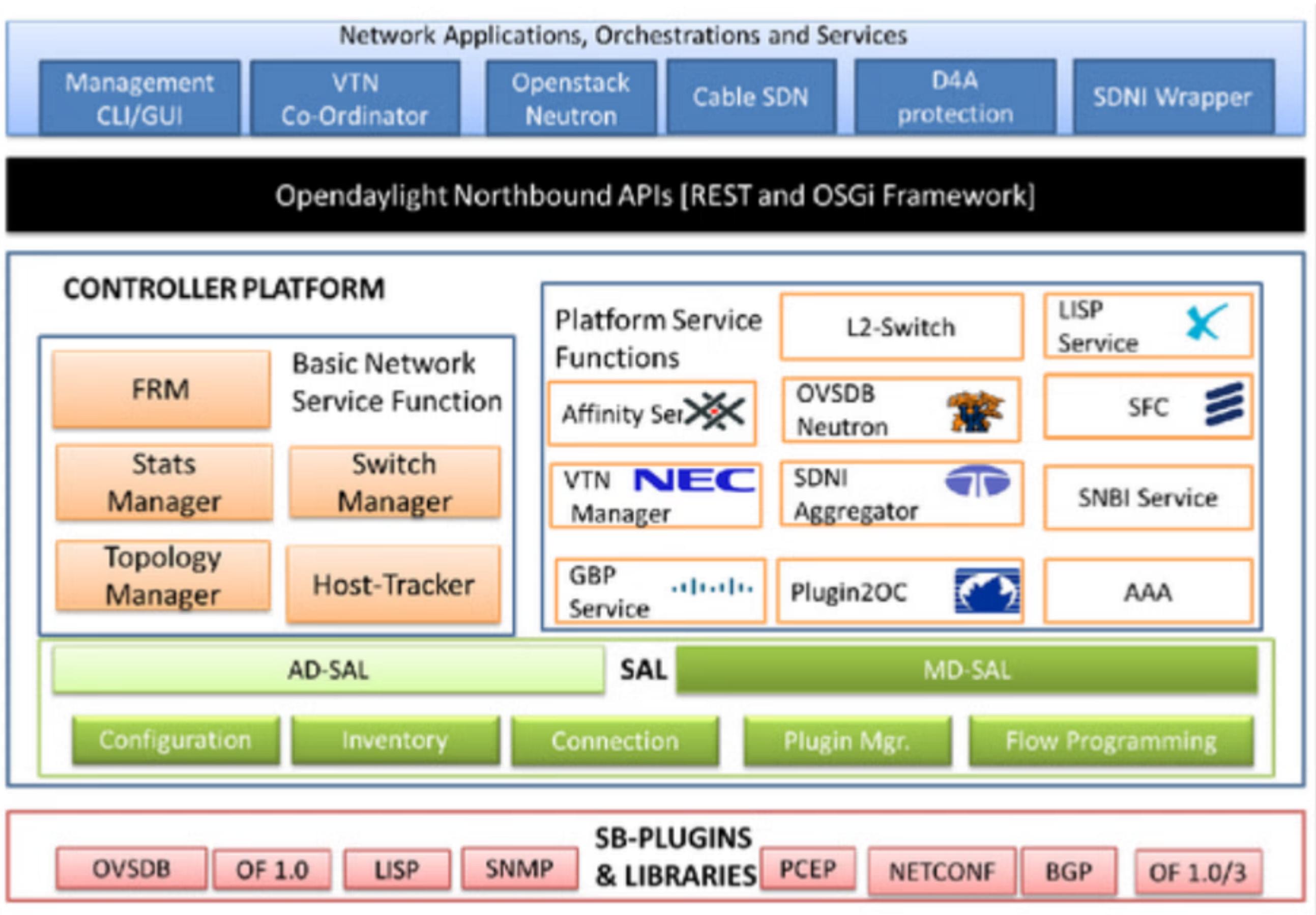
Open source بودن و تمرکز آن بر نوآوری این قابلیت را دارد که دولوپرهای سراسر جهان آن را بهبود بخشدند و به ارتقا و بهبود آن و حتی نوآوری در آن کمک کنند.

یکی از نکات قابل توجه در مورد ODL این است که توسط بنیاد لینوکس مدیریت میشود و در کمپانی هایی مانند Cisco ، IBM ، NEC استفاده میشود.

معماری خلاصه شده ODL



معماری با جزیات ODL



: ODL بخش های معماری

1. Southbound plugins and protocols forming the network device layer.
2. Service adaptation and network functions forming the coordination and control layer.
3. Northbound APIs and applications forming the application layer.

مُنْبَع:

<https://thenewstack.io/sdn-series-part-vi-opendaylight/>

