هو الحق

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

تمرین درس شبکه‌های کامپیوتری پیشرفته

برنامه‌نویسیِ شبکه‌های نرم‌افزار محور (SDN)

سیاوش خرسندی

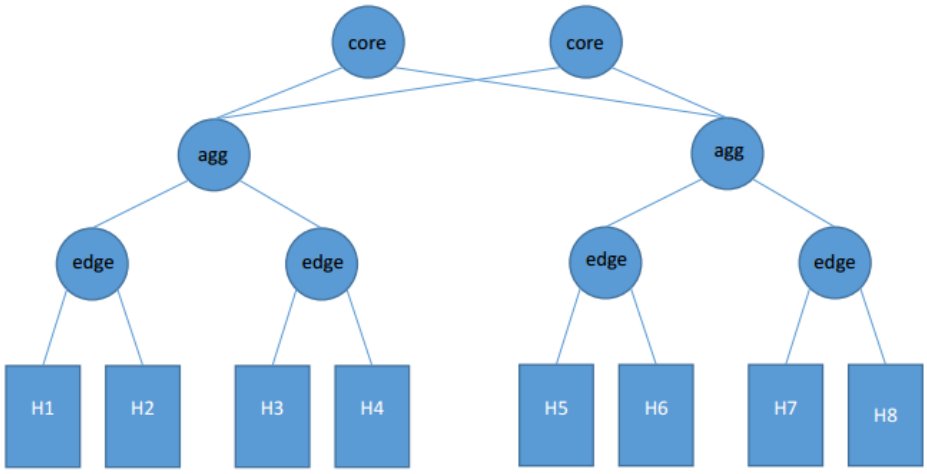
در بخش اول این تمرین از شما خواسته شده ابتدا توپولوژی FatTree که در مراکز داده مورد استفاده قرار می‌گیرد را با مینی‌نت پیاده‌سازی و اجرا کنید. در ادامه شاهد خواهید بود که به دلیل وجود حلقه در این شبکه، تست pingall با شکست مواجه خواهد شد. برای رفع این مشکل باید از ماژول‌های آماده در کنترلر POX با نام l2_learning و spanning_tree استفاده کنید که در ادامه نتیجه‌ی تست pingall موفقیت‌آمیز خواهد بود. پس از رفع این مشکل از شما خواسته شده تا خودتان ماژولی برای «کنترل دسترسی» در این کنترلر بنویسید که توضیحات آن را در ادامه ارائه شده است.

در بخش دوم تمرین که اختیاری نیز هست، از شما خواسته شده تا این بار از زبان برنامه‌نویسی سطح بالای Pyretic استفاده کرده و یک دیواره‌ی آتش برای شبکه بنویسید. لیست ماشین‌هایی که نمی‌توانند با یکدیگر در شبکه ارتباط داشته باشند به شما داده شده است و باید ارتباطات آن‌ها را با یکدیگر مسدود کنید.

I. بخش اول:

هدف از این تمرین آشنایی با ماژول‌های کنترلر پاکس و شبیه‌سازی یک محیط تصدیق است. یکی از مزیت‌های محیط نرم افزار محور اجرای برنامه‌های کاربری مختلف بر روی کنترلر شبکه است. به این صورت این برنامه‌های کاربردی به صورت مستقل از هم می‌توانند در کنترل شبکه همکاری کنند. در این تمرین شما با ماژول‌های ساخت درخت پوشا، مسیریابی لایه ۲ آشنا خواهید شد و از آن‌ها در کنار یک ماژول کنترل دسترسی که خودتان باید ایجاد کنید استفاده میکنید

در اولین گام این تمرین شما باید توپولوژی دارای حلقه به شکل زیر را با مینی‌نت ایجاد کنید.



در این توپولوژی، امکان استفاده از کنترلر پیش‌فرض وجود ندارد چرا که در آن برای برخورد با حلقه راهکاری اندیشیده نشده است. سعی کنید در این توپولوژی عمل ping را انجام دهید و مشاهدات خود را ثبت کنید.

در مرحله‌ی بعد شما باید کنترلر شبکه را به کنترلر remote تغییر دهید. سپس در هنگام راه‌اندازی کنترلر POX ماژول‌های درخت پوشا و مسیریابی لایه ۲ را آغاز کنید. در این مرحله شبکه شما باید کارکرد معمولی داشته باشد، یعنی بتوانید عمل ping را انجام دهید. کد زیر نحوه انتخاب کنترلر remote برای شبکه را نشان می‌دهد:

```
topo = DCTreeTopo(k)

net = Mininet(topo, link=TCLink, build=False,autoSetMacs = True)

net.addController(name='c0',controller=RemoteController,ip='127.0.0.1',port=6633)

net.start()
```

کد زیر نیز نحوه راه‌اندازی پاکس به همراه دو ماژول مذکور را نشان میدهد:

```
~/pox/pox.py openflow.spanning_tree forwarding.l2_learning
```

پس از راه‌اندازی شبکه‌ی دارای حلقه، حال باید ماژول کنترل دسترسی خود را ایجاد کنید. به این منظور فرض بر این است که ماشین شماره یک وظیفه‌ی تصدیق و شناسایی ماشین‌های داخل شبکه را برعهده دارد، به این ترتیب که هر ماشین باید پیش از اینکه پیامی در شبکه ارسال نماید، یک پیغام به ماشین یک ارسال و جوابی دریافت کند و فقط پس از دریافت پیغام از ماشین یک می‌تواند با سایر ماشین‌های داخل شبکه ارتباط برقرار کند. به این منظور شما باید لیستی از ماشین‌های تصدیق‌شده را در ماژول خود نگهداری کنید و هر زمان که ماشینی پیغامی از ماشین یک دریافت کرد، به لیست مورد نظر اضافه شود. آنگاه دو ماشین به شرطی می‌توانند مکاتبه داشته باشند که هر دو تصدیق‌شده باشند.

در طول پیاده‌سازی کنترل دسترسی، آدرس MAC ماشین‌ها را برابر آدرس IP آن‌ها قرار دهید. به این صورت ماشین با آدرس MAC زیر همان تصدیق‌کننده است:

۰۰:۰۰:۰۰:۰۰:۰۰:۰۱

شما باید در ماژول خود رخداد PacketIn را با پیاده‌سازی تابع __handle_PacketIn دریافت کنید و بر اساس منطق زیر عمل کنید:

۱. اگر آدرس MAC فرستنده همان تصدیق‌کننده است، آدرس MAC گیرنده را به لیست مورد نظر اضافه کنید و هیچ کار دیگری انجام ندهید.
۲. اگر آدرس MAC فرستنده و گیرنده در لیست موجود است و یا گیرنده تصدیق‌کننده است، هیچ قانونی را ثبت نمی‌کنید، چرا که مسیریاب لایه ۲، بسته را به مقصد خواهد رساند.
۳. اگر آدرس فرستنده و یا گیرنده در لیست وجود ندارد، قانونی برای حذف کردن بسته‌ی مورد نظر وضع کنید. مقادیر idle_timeout و hard_timeout را به ترتیب برابر ۱۴ و ۹۴ ثانیه قرار دهید. دقت داشته باشید که اولویت قانونی که وضع میکنید باید از اولویت قانونهای مسیریاب لایه ۲ باید بیشتر باشد که اجرای همزمان این دو ماژول نتیجه‌ی مورد نظر، یعنی رعایت کنترل دسترسی را به دنبال داشته باشد.

برای پیاده‌سازی این ماژول، کار خود را با آشنا شدن با فایل l2_learning در مسیر ~/pox/pox/forwarding شروع کنید. می‌توانید منطق مورد نظر را در تابع __handle_PacketIn آن پیاده‌سازی کنید. اگر نام ماژول کنترل دسترسی را AccessCtrl.py و آن را در پوشه ~/pox/pox/forwarding قرار دهید، حال می‌توانید پاکس را به همراه ماژول خود به شکل زیر راه‌اندازی کنید.

~/pox/pox.py openflow.spanning_tree forwarding.l2_learning forwarding.AccessCtrl

گزارش باید حاوی مطالب زیر باشد:

۱. مشاهدات خود در بخش انجام عمل ping در شبکه دارای حلقه و کنترلر پیش‌فرض را بنویسید.
۲. ابتدا شبکه را راه‌اندازی کنید و فقط دو ماژول درخت پوشا و مسیریاب لایه ۲ را راه‌اندازی کنید. با اجرای عمل pingall اتصال تمام ماشین‌ها به یکدیگر را نشان دهید. از این آزمایش اسکرین‌شات تهیه کنید و در گزارش خود قرار دهید.
۳. توضیح دهید که حفظ اولویت قانون‌های کنترل دسترسی نسبت به مسیریاب چگونه پیاده‌سازی شده است.
۴. کنترلر را متوقف کنید و این بار آن را با سه ماژول درخت پوشا، مسیریاب و کنترل دسترسی خودتان اجرا کنید. اکنون عمل ping را برای ماشین‌های ۲ و ۳ انجام دهید. این دو ماشین در این مرحله نباید بتوانند یکدیگر را ببینند. سپس این عمل را برای ماشین‌های ۲ و ۱ انجام دهید که این عمل باید اجرا شود. اکنون دوباره ping بین ۲ و ۳ را تکرار کنید که در این مرحله باز هم نباید بتوانند یکدیگر را ببینند. سپس ping بین ۳ و ۱ و در نهایت بین ۲ و ۳ را اجرا کنید. از مراحل این آزمایش اسکرین‌شات تهیه کنید.

شما همچنین باید کد بخش توپولوژی شبکه را در قالب فایلی به نام LoopTopo.py و کنترل دسترسی را در فایلی به نام AccessCtrl.py به همراه گزارش ارسال کنید. تمام خروجی‌ها را در فایل فشرده‌ای قرار دهید و اسم آن را همان شماره دانشجویی خود قرار دهید. تمپلیت برای ایجاد برنامه‌ی توپولوژی شبکه و مازول کنترل دسترسی نیز به همراه گزارش تمرین در اختیار شما قرار گرفته است.

I. بخش دوم (اختیاری):

هدف از این تمرین آشنایی با زبان Pyretic است. در تمرین قبلی با APIهایی که کنترلر POX برای کنترل شبکه در اختیار شما قرار می‌دهد آشنا شدید. از اشکالات روش پیشین ورود به جزئیات بیش از اندازه برای پیاده‌سازی ماژول‌های مورد نظر و همچنین کم بودن قابلیت حمل ماژول‌های ایجاد شده است. در زبان Pyretic به راحتی می توان دستورات سطح بالا و بدون نیاز به اطلاع از جزئیات پیاده‌سازی را استفاده کرد.

در زبان Pyretic برای کنترل شبکه هر ماژول تابعی به نام main دارد که باید یک Policy را بازگرداند. Policy تکه کد یا ماژول یا ترکیبی از چند ماژول دیگر است که مشخص می‌کند شبکه در مقابل دریافت یک بسته چطور باید رفتار کند. یک Policy یک بسته را می‌گیرد و می‌تواند صفر یا بیشتر بسته به عنوان خروجی ایجاد کند. مثلاً قطعه کد زیر یک hub را پیاده‌سازی می‌کند:

```
from pyretic.lib.corelib import*

def main:()

    return flood()
```

در این قسمت شما ماژولی ایجاد خواهید کرد که باید بر روی هر توپولوژی به درستی کار کند. ماژول شما در ابتدای شروع به کار خود باید فایلی به نام blocked_hosts.csv را بخواند. فرمت این فایل به شکل زیر است:

<line num>, <host 1 mac>, <host 2 mac>

جفت آدرس‌های MAC که در هر خط این فایل قرار دارند اجازه ارتباط با همدیگر را ندارند. شما باید در تابع main یک policy ایجاد کنید که در آن مشخص شود چه هاست‌هایی نمی‌توانند با همدیگر ارتباط داشته باشند (از match و عملگر | استفاده کنید). سپس با استفاده از امکانات زبان Pyretic این policy را بر عکس کنید (~) و به همراه ماژول mac_learner بازگردانید. به طور مثال اگر Policy نهایی شما allowed_hosts نام داشته باشد، در انتهای تابع main عبارت زیر باید برگردانده شود:

allowed_host >> mac_learner

این ماژول را پس از نوشتن، در پوشه‌ی پایرتیک و در پوشه‌ی ماژول‌ها ذخیره کنید. در ادامه باید یک توپولوژی ساده و دلخواه ایجاد کنید. پیشنهاد ما ایجاد یک توپولوژی تک سوئیچی با دستور زیر است:

sudo mn —topo single,5 —controller remote —mac

به‌عنوان مثال در فایل بلاک‌شده‌ها در نظر بگیرید که هاست‌های ۲ و ۴ و همچنین ۳ و ۵ مجوز ارتباط ندارند. عمل pingall را انجام داده و در یک اسکرین‌شات گزارش کنید که این دو جفت هاست نتوانسته‌اند یکدیگر را ping کنند. تمپلیت برای این ماژول نیز به همراه گزارش تمرین، در اختیار شما قرار گرفته است.

موفق باشید.