پروژه درس شبکه های کامپیوتری

توضیحات مینی نت و ریو

دانشگاه تهران

۱۳۹۹

فهرست مطالب

۱

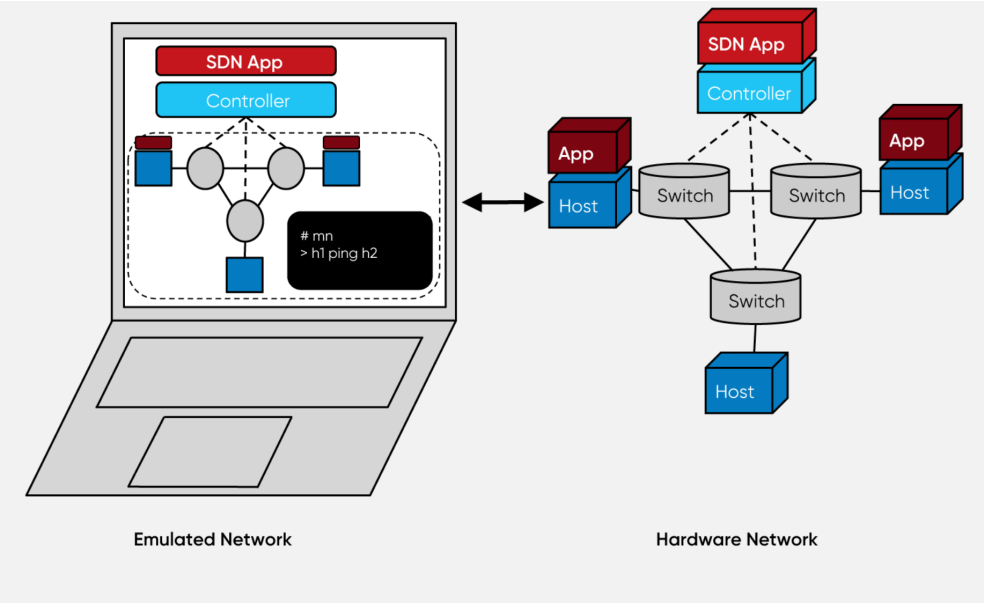
آشنایی با محیط مینی نت

مقلد MiniNet ،نرم افزاری است که اکثر محققین حوزه شبکه و پژوهشگران برای مدل سازی و تحلیل و ارزیابی کارآیی شبکه های SDN (شبکه های مبتنی بر نرم افزار) و پروتکل های موجود از آن استفاده می کنند. با استفاده از مقلد مینی نت می توان یک شبکه SDN را قبل از پیاده سازی واقعی، مدل سازی نموده و پارامترهای مختلف شبکه را در شبیه سازی تغییر داد و نتایج حاصل از شبیه سازی ها را مورد ارزیابی و مقایسه قرار داد. هم چنین با استفاده از مقلد مینی نت میتوان پارامتر های مربوط به ارزیابی شبکه های کامپیوتری را محاسبه کرد و تحلیل فنی از ساختار شبکه ارائه داد.

۱.۱ بررسی معماری مینی نت

ساختار کلی مینی نت در شکل قابل ۱.۱ مشاهده است.

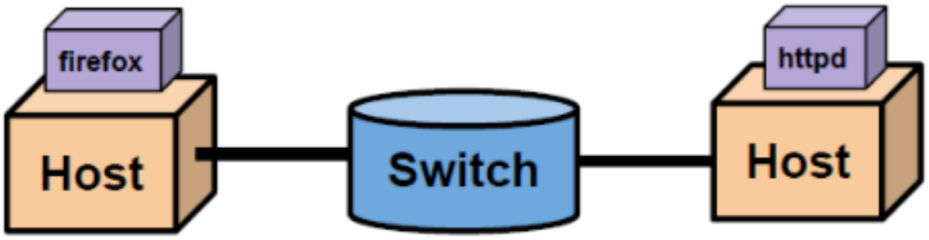
- Isoloated Host: هاست ها در مینی نت در واقع گروهی از فرآیند(پروسس)های سطح کاربر میباشند که مالکیت واسط های مجازی (Virtual Interface) کارت شبکه را به عهده میگیرد. هر کدام از هاست ها می توانند عملکرد خاصی(مانند وب سرور) ارائه دهند.



شکل ۱.۱ ساختار مینی نت

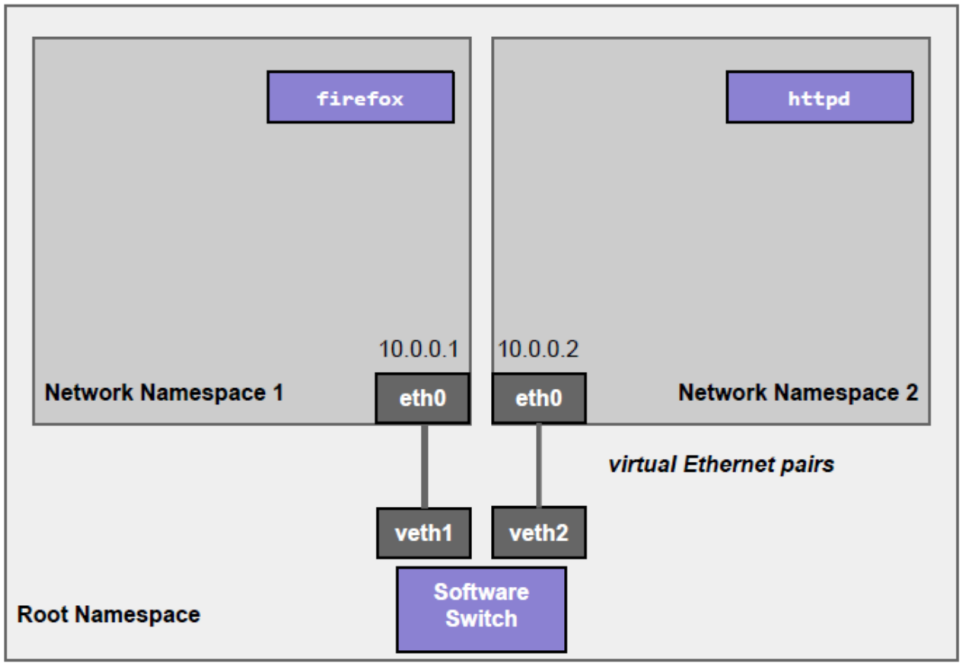
- Emulated Links: از لینک های ارتباطی بین هاست ها می توان نرخ ارسال داده را بین گره های شبکه تنظیم کرد.
- Emulated Switches: سوئیچ های موجود در مینی نت از نوع سوئیچ هایی با قابلیت استفاده در شبکه نرم افزار محور SDN میباشد. این سوئیچ ها (Open vSwitch) وظیفه سوئیچینگ پکت ها در شبکه را به عهده دارند.

هم چنین توجه کنید که در پیاده سازی این پروژه، کنترلر استفاده شده Ryu میباشد. در شکل ۲.۱ معماری مینی نت را با استفاده از مثالی نشان میدهیم.



شکل ۲.۱ مثالی از شبیه سازی در مینی نت

مثال ۱.۱.۱ (مثال یک) فرض می کنیم ساختار شبیه سازی شبکه ای را داریم که در آن دو هاست توسط یک سوئیچ به هم متصل شده اند. یکی از هاست ها وب سرور است و دیگر هاست به عنوان سرویس گیرنده(فایرفاکس) درخواست سرویس از وب سرور را دارد. ساختاری معماری مقلد مینی نت با در نظر گرفتن مثال فوق در شکل ۳.۱ قابل مشاهده است. در مینی نت هر هاست دارای یک واسط شبکه است که با eth مشخص میشود. هر یک از واسط های شبکه در مینی نت میتواند آیپی آدرس اختصاصی داشته باشند. به عنوان مثال در شکل ۳.۱ ، آیپی هاست اول برابر 10.0.0.1 و آیپی هاست دوم برابر 10.0.0.2 میباشد. به ساختار اتصالی واسط شبکه هر یک از هاست ها با پورت های سوئیچ دقت کنید.



شکل ۳.۱ معماری شبکه مینی نت

نکته ۱.۱ **(مراحل نصب مینی نت)** مراحل نصب مینی نت آورده شده است. توجه فرمایید اگر زمان کافی برای پیگیری دستورات نصب مینی نت ندارید، می توانید ماشین مجازی آماده مینی نت را دانلود کنید و مستقیماً از آن استفاده کنید. در ماشین مجازی آماده مینی نت، تمام بسته های مورد نیاز نصب شده است و آماده به شروع انجام آزمایش میباشد.

۲.۱ شروع به آزمایش با مینی نت

پس از نصب مینی نت، با ورود به CLI لینوکس، دستور

کد ۱.۱ اجرای مینی نت

```
۱  sudo mn
```

را وارد می کنیم. این دستور در واقع ساده ترین شبیه سازی توپولوژی شبکه در مینی نت را اجرا می کند. با اجرای این دستور ساختار توپولوژی شبیه سازی شامل یک سوئیچ OpenFlow Enabled و دو هاست می باشد که هر یک از هاست ها به سوئیچ متصل شده است. هم چنین کنترلر پیش فرض مینی نت اجرا شده است که در ساده ترین حالت ممکن کنترلر امکان پیاده سازی منطق سوئچینگ در لایه دوم را ارائه میدهد. بعد از اجرای دستور فوق، mininet CLI قابل مشاهده است که اطلاعات اضافی و دستورات از کاربر را هنگام شبیه سازی شبکه دریافت می کند. به عنوان مثال دستور ذیل، تمام گره های شبیه سازی در حال اجرا را نشان میدهد

کد ۲.۱ گره های مینی نت

```
mininet>nodes
```

توجه کنید که هاست ها در مینی نت به صورت پیش فرض با نام مخفف h نام گذاری شده اند. به عبارت دیگر، اجرای دستور ذیل

کد ۳.۱ دستور پینگ

```
mininet>h1 ping -c 1 h2
```

سبب میشود که هاست اول، هاست دوم را پینگ کند. جهت آشنایی بیشتر با دستورات مینی نت به مستندات و ویدیو ها رجوع کنید.

توپولوژی شخصی سازی شده در مینی نت

قابل توجه است که مینی نت از توپولوژی های پارامتری هم پشتیبانی می کند. در واقع با نوشتن چند خط کد پایتون و استفاده از کلاس ها و متد های مینی نت، شما قادر خواهید بود که به سلیقه خود، توپولوژی مینی نت را شبیه سازی کنید. توجه کنید که تمام هسته مینی نت و کنترلر ریو به زبان پایتون نوشته شده است. برای درک بهتر استفاده از توپولوژی شخصی سازی شده، تکه کد پایتون زیر را در نظر بگیرید. اسم این تکه کد را mytopo انتخاب میکنیم.

```
"""Custom topology example

Two directly connected switches plus a host for each
    switch:

   host --- switch --- switch --- host

Adding the 'topos' dict with a key/value pair to generate
    our newly defined
topology enables one to pass in '--topo=mytopo' from the
    command line.
"""

from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
```

```
۲۵          leftSwitch = self.addSwitch( 's3' )
۲۶          rightSwitch = self.addSwitch( 's4' )
۲۷
۲۸          # Add links
۲۹          self.addLink( leftHost, leftSwitch )
۳۰          self.addLink( leftSwitch, rightSwitch )
۳۱          self.addLink( rightSwitch, rightHost )
۳۲
۳۳
۳۴  topos = { 'mytopo': ( lambda: MyTopo() ) }
```

در این مثال با استفاده از API های مینی نت، به صورت دستی توپولوژی های شبکه را ایجاد کردیم که در آن دو سوئیچ به هم متصل شده است و به هر یک از سوئیچ ها نیز یک هاست متصل شده است. به نحوه فراخوانی متد ها و کامنت کد توجه کنید. پس از ایجاد این فایل پایتون، نحوه فراخوانی آن از داخل لینوکس در فایل های ضمیمه آورده شده است. به عنوان خود آزمایی تکه کد زیر را تحلیل کنید و سپس آن را اجرا کنید.

```
۱  from mininet.net import Mininet
۲  from mininet.topo import Topo
۳  from mininet.link import TCLink  # rate limit links
۴  from mininet.cli import CLI  # bring up the Mininet CLI
۵  topo = Topo()  # Create an empty topology
۶  topo.addSwitch("s1")  # Add switches and hosts
۷  topo.addSwitch("s2")
۸  topo.addHost("h1")
۹  topo.addHost("h2")
۱۰  topo.addHost("h3")
۱۱  # Wire the switches and hosts together,
۱۲  topo.addLink("h1", "s1", bw=20.0, delay='10ms', use_htb=
        True)
۱۳  topo.addLink("h2", "s1", bw=25.0, delay='10ms', use_htb=
        True)
۱۴  topo.addLink("s1", "s2", bw=11.0, delay='40ms', use_htb=
        True)
۱۵  topo.addLink("h3", "s2", bw=15.0, delay='7ms', use_htb=
        True)
۱۶  net = Mininet(topo=topo, link=TCLink)
۱۷  net.start()
۱۸  CLI(net)  # Bring up the mininet CLI
۱۹  net.stop()
```

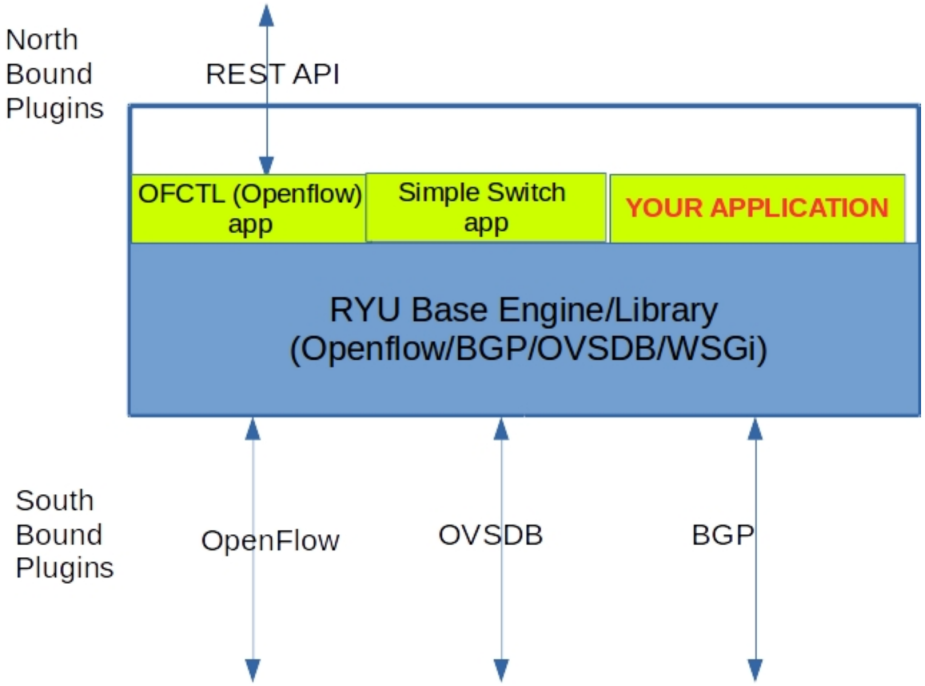
کد ۴.۱ اجرای خود آزمایی

```
۱  mininet@mininet-vm:~/myfolder$ sudo python ExMNtcSimple.py
```

۳.۱ کنترلر Ryu

ریو در واقع یک فریم ورک شبکه نرم افزار محور (SDN) می باشدکه به اجزای متفاوت (Component-Based) تقسیم شده است. ریو با ارائه دادن کامپوننت های نرم افزاری، ایجاد عملکرد های مدیریتی و کنترلی را برای کاربران (SDN) فراهم آورده است. ریو پروتکل های متفاوتی مانند

OpenFlow, OVSDB, BGP را پشتیبانی میکند. معماری ریو در شکل ۴.۱ آورده شده است.



شکل ۴.۱ معماری ریو

همانطور که در شکل ۴.۱ می بینید، شما می توانید با نوشتن منطق برنامه های خود در ریو و استفاده از در کنترلر مینی نت، عملکرد (Functionality) مورد نظر خود را در شبکه پیاده سازی کنید(در این پروژه میخواهیم الگوریتم پیدا کردن کوتاه ترین مسیر را پیاده سازی کنیم). حال به عنوان مثال فرض کنید که می خواهیم اپلیکیشنی در ریو بنویسیم که قابلیت عملکرد Switching Hub را ارائه دهد. در این حالت ابتدا منطق عملکردی که میخواهیم آن را پیاده سازی کنیم را متصور میشویم.

تعریف ۱.۳.۱ **منطق برنامه سوئچینگ** برای Switching Hub میتوان عملکرد های گوناگونی را ارائه داد. در این سناریو ما فقط دنبال دو هدف ذیل هستیم

- اپلیکیشن نوشته شده توسط ما باید بتواند آدرس های مک هر یک از هاست های متصل شده به سوئیچ در مینی نت را پیدا کند و در جدولی ذخیره کند
- در هنگام دریافت پکت که مک آدرس آن را میداند، بسته را بر روی پورت متصل به هاست مورد نظر ارسال کند
- در صورتی که پکتی دریافت کرد که آدرس مک آن را نمی داند، بسته مورد نظر را Flood کند

فایل پی دی اف ضمیمه شده، مرحله به مرحله پیاده سازی این مهم را در ریو و مینی نت به نمایش میگذارد. بررسی این مثال ساده در پیاده سازی اپلیکیشن نهایی پروژه شما که عملکرد الگوریتم کوتاه ترین مسیر را پیاده سازی میکند، کمک شایانی را ارائه میدهد. توجه کنید که ارتباط بین مینی نت و ریو نیز در این مثال بررسی شده است.