

۶ ایجاد بُرش شبکه

در این فصل، بُرش (Slice) شبکه با استفاده از کنترلر FlowVisor، مورد بحث قرار خواهد گرفت. بنابراین در ادامه به موضوعات ذیل خواهیم پرداخت:

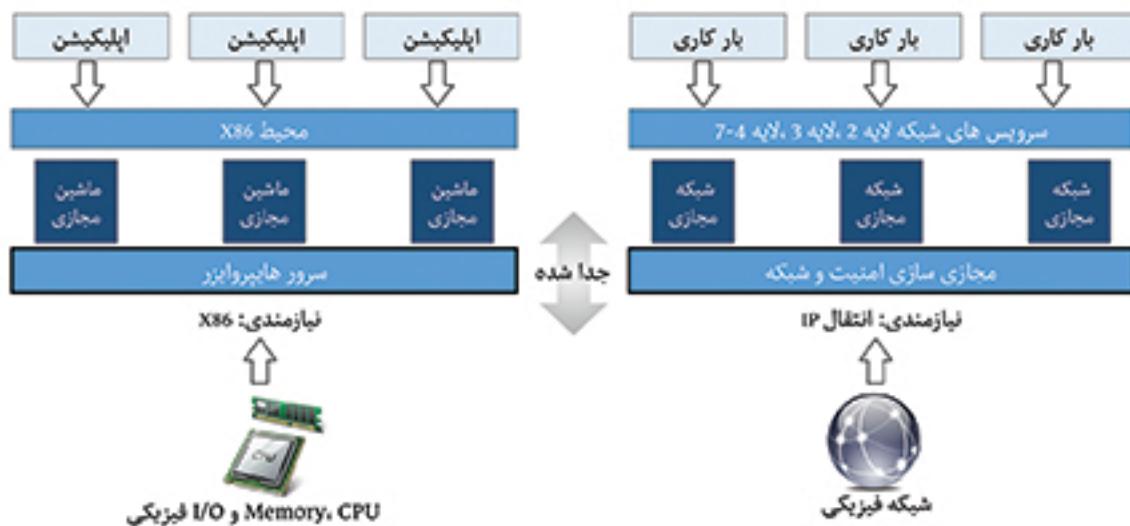
- مجازی سازی شبکه
- کنترلر FlowVisor به عنوان یک ابزار متن باز برای بُرش شبکه‌های مبتنی بر OpenFlow
- عملیات انطباق جریان، همچنین ساختار Action در بُرش FlowVisor API
- بُرش دادن شبکه با استفاده از Mininet

مجازی سازی شبکه

مجازی سازی شبکه، انتزاع ویژه‌ای از زیر ساخت شبکه فیزیکی است که به موجب آن، شبکه‌ای منطقی (بر روی بستر فیزیکی) ایجاد می‌شود که می‌تواند ساختاری متفاوت از ساختار فیزیکی داشته باشد. با استفاده از زیر ساخت فیزیکی شبکه، چندین زیر ساخت شبکه منطقی را بصورت مجازی (برای مثال تعدادی سوئیچ، مسیر و لینک) می‌توان فراهم آورد.

مجازی سازی شبکه را می‌توان با مجازی سازی یک سرور مقایسه کرد. در ادامه می‌توانید این مقایسه را مشاهده کنید:

شکل ۶-۱
مقایسه مجازی‌سازی
در کامپیوتر و شبکه



در سمت چپ این شکل می‌توانید مجازی‌سازی در لایه سخت افزار یک سرور را مشاهده کنید. این مجازی‌سازی در قالب یک هایپروایزر (Hypervisor) و ماشینهای مجازی روی آن، نمایش داده شده است. در این محیط، پردازشگر فیزیکی (CPU)، حافظه (RAM) و ورودی / خروجی (I/O) توسط یک هایپروایزر، انتزاع می‌شوند. بر روی هایپروایزر، یک ماشین مجازی می‌تواند اجرا شود. هایپروایزر اساساً این اطمینان را می‌دهد که هر ماشین مجازی به طور مجزا مدیریت شده و تنها به منابع سخت افزاری اصلی که برای آن معین شده، دسترسی داشته باشد. حال به موضوع بحث خود باز می‌گردیم. بطور مشابه، یک شبکه فیزیکی نیز می‌تواند مجازی شود. در سمت راست تصویر پیشین، لایه مجازی‌سازی شبکه نمایش داده شده است. این لایه مسئول ایجاد یک نمای تفکیک شده از زیرساخت شبکه فیزیکی است. ایجاد یک شبکه مجازی نیاز به تکنولوژی ساخت گره‌های مجازی (Virtual Nodes) دارد (تکنولوژیهایی همچون، Linux Network Namespaces, Xen Virtual Machine Monitor, VMWare, KVM^۱ و VirtualBox). روش‌های دیگری نیز برای ایجاد لینکهای مجازی وجود دارد. همه این روش‌ها اساساً بر مبنای تکنولوژی تونل زدن^۲ (Tunneling) ساخته می‌شوند. یکی از این روش‌ها، استفاده از فریم اترنت در یک گره مجازی (ماشین مجازی) و کپسوله کردن آن در قالب یک پسته IP بوده تا بتواند در تجهیزات (Hop) گوناگون شبکه، مسیر خود را طی کند. این تکنیک اساساً یک لینک اترنت مجازی را با استفاده از تکنولوژی تونل زدن فراهم می‌کند. همچنین تکنولوژیهایی مانند Open vSwitch وجود دارند که می‌توانند سوئیچهای مجازی ایجاد کنند. این تکنولوژیها نقش به سزایی را در

^۱ Kernel-based VirtualMachine

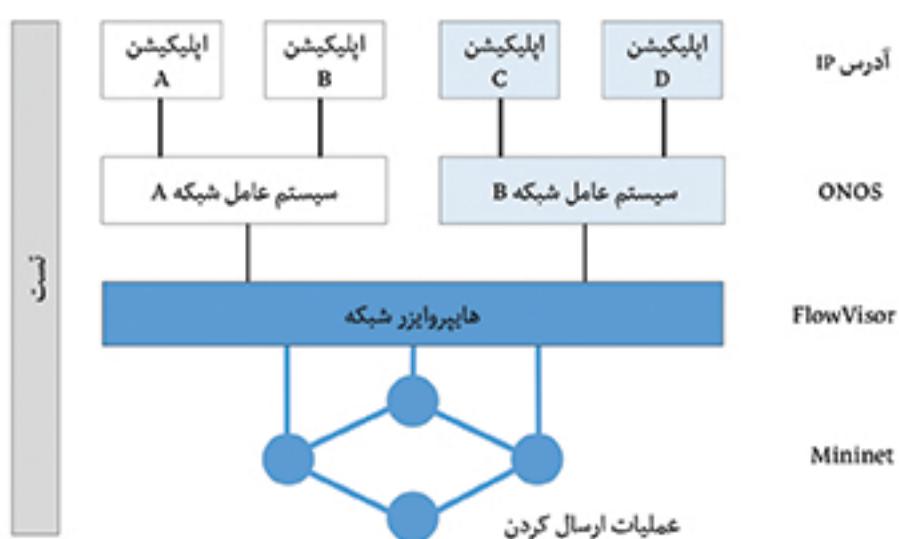
^۲ از جمله تکنولوژیهای تونل زدن می‌توان به Virtual Extensible Local Area Network (VxLAN), Stateless Transport Tunneling (STT), Ethernet Generic Routing Encapsulation (GRE) و ... اشاره کرد. (متوجه)

فرایند مجازی سازی شبکه ایفا می‌کنند. با تمامی توضیحات ذکر شده، حال می‌توان اینگونه بیان کرد، هدفی که مجازی سازی شبکه دنبال می‌کند متفاوت از هدفی است که SDN بدنبال آن است. هدف SDN جداسازی بین کامپوننتهای Data Plane و Control Plane است، در حالی که هدف مجازی سازی شبکه، ساختن شبکه‌های مجازی گوناگون برروی یک زیر ساخت شبکه فیزیکی است.

FlowVisor

یک SDN با کنترلرهای منطقی گوناگون، می‌تواند سطحی از تمرکز زدایی یا به عبارتی پراکندگی منطقی بین اجزای خود داشته باشد. در این بین، نوع جالبی از کنترلر به نام FlowVisor وجود دارد که می‌تواند برای افزودن سطحی از مجازی سازی شبکه‌های مبتنی بر OpenFlow به کار رفته و به چندین کنترلر اجازه دهد که بطور همزمان مجموعه‌ای از سوئیچهای فیزیکی همپوشان را کنترل کنند. این کنترلر در ابتدا به این منظور ایجاد شد که به محققان اجازه دهد آن را بر روی شبکه‌های استقرار یافته‌ای که ترافیک داده واقعی در آنها جریان دارد، اجرا کنند؛ همچنین گسترش خدمات در محیط‌های SDN را تسهیل بخشد. کنترلر FlowVisor را می‌توان به عنوان یک کنترلر OpenFlow با هدف ویژه در نظر گرفت که مانند یک پراکسی Transparanet بین سوئیچهای OpenFlow از یک طرف و چندین کنترلر OpenFlow از طرف دیگر، عمل می‌کند. در شکل زیر می‌توانید جایگاه یک کنترلر FlowVisor را در یک شبکه IP مشاهده کنید.

شکل ۶-۲
FlowVisor
جایگاه
در شبکه
SDN



حال در ادامه قصد داریم به ویژگی منحصر بفرد این کنترلر پردازیم. این کنترلر از قابلیتی به نام برش استفاده می‌کند که به موجب آن می‌تواند برروی یک بستر فیزیکی، چندین شبکه مجازی را ایجاد کند.

کنترلر FlowVisor می‌تواند برش‌هایی غنی از منابع شبکه را ایجاد کرده و کنترل هر برش را به یک کنترلر متفاوت واگذار کند و با این کار، باعث جدایی بین برش‌ها خواهد شد. FlowVisor (که در اصل در دانشگاه استنفورد ایجاد شد) به منظور پشتیبانی از قابلیت برش شبکه، بطور گسترده‌ای در شبکه‌های آموزشی و تحقیقاتی مورد استفاده قرار گرفت. با این روش، محققان مختلف قادرند برش مجزایی از شبکه خود را داشته و با استفاده از سیستم عامل شبکه و مجموعه‌ای از اپلیکیشن‌های نظارتی و مدیریتی، کنترل کاملی بر آن بخش از شبکه ایجاد کنند. FlowVisor به شما این امکان را می‌دهد که تحقیقات در حوزه شبکه را برای محیط‌های غیر آزمایشگاهی و روی ترافیک واقعی شبکه، عملی سازید. FlowVisor یک کنترل پراکسی متن باز بوده و می‌توانید کد منبع آن را برای سازگاری با نیازهای خود، سفارشی‌سازی کنید؛ با یک اینترفیس مانیتورینگ و پیکربندی در ^۷ JSON برای کاربران و یک زبان برنامه‌نویسی جاوا برای توسعه دهنده‌گان، هر شخصی از طریق انتخاب سرویسهای مختلف، توانایی سفارشی‌سازی FlowVisor را دارد. می‌توانید به سادگی و به سرعت، یک شبکه SDN را با تمام عملکردهای اساسی اش آزمایش کنید تا بتوانید مجازی‌سازی را فرا بگیرید و روش‌های جدید را به سرعت برای گسترش سرویسها، آزمایش کنید. از آنجایی که FlowVisor بر استانداردهای باز استوار است، می‌تواند در یک زیرساخت فیزیکی با طیف گسترده‌ای از محصولات اجرا شود. این کنترلر از برندهای مختلف سوئیچ OpenFlow (مانند NEC، Pronto، HP، OVS و دیگر نمونه‌ها) و سیستم عامل‌های گوناگون شبکه (مانند کنترلرهای OpenFlow) پشتیبانی می‌کند.

برای اطلاعات بیشتر درباره FlowVisor و کد منبع می‌توانید به آدرس <http://www.flowvisor.org> مراجعه کنید. دستورالعمل نصب در

آدرس زیر موجود است:

<http://github.com/OPENNETWORKINGLAB/flowvisor/wiki/Installation-from-Binary>



FlowVisor API

همانطور که بیشتر نیز بیان کردیم، کنترلر FlowVisor می‌تواند برش‌هایی از منابع شبکه فراهم کند و مدیریت هر برش را به یک کنترلر OpenFlow خاص محول کند. برش‌ها را می‌توان با هر ترکیبی از محتوای بسته دریافتی توسط سوئیچ (از لایه ۱ گرفته تا لایه ۴)، تعریف کرد. این ترکیبها شامل موارد زیر است:

- پورتهای سویچ (لایه ۱)
- آدرس MAC ارنت مبدأ / مقصد یا ارزش فیلد Ethernet Type (لایه ۲)
- آدرسهای IP مبدأ / مقصد یا نوع آنها (منظور IPv4 یا IPv6 یا موارد دیگر) (لایه ۳)
- شماره / نوع پورت TCP/UDP مبدأ / مقصد یا کد / نوع ICMP (لایه ۴)

کنترل FlowVisor تفکیک برش‌ها از یکدیگر را امکان‌پذیر کرده و آن را اجرایی می‌کند. این تفکیک به این معناست که ترافیک داده موجود در یک برش، توسط میزبانهای (کامپیوترها) دیگر برش‌ها، دریافت نمی‌شود. حال به سراغ API این کنترلر می‌رویم. در طور تدریجی XML-RPC جای خود را به JSON می‌دهد. به عبارت دیگر XML-RPC بصورت تدریجی در حال حذف شدن از این API است. به کاربران FlowVisor توصیه می‌شود تمام وابستگی‌های خود به FlowVisor API را به رابط JSON منتقل کنند. به یاد داشته باشید که Syntax مربوط به API، در این انتقال ممکن است در برخی موارد تغییر کند؛ لذا پیشنهاد می‌شود، آخرین مستندات FlowVisor را برای آشنایی با Syntax جدید، مطالعه کنید. برای دسترسی به این API، می‌توان از یک ابزار خط فرمان به نام fvctl (با فرمان dptcl مطالعه کنید). برای فراخوانی fvctl از این API استفاده کرد. به عنوان مثال، برای مشاهده لیست برش‌ها، می‌توان از `fvctl list-slices` مطابق فرمان زیر استفاده کرده و `list-slices` را فراخوانی کرد.

```
$ fvctl list-slices
```

شامل فرمانهای زیر می‌شود:

- فرمان `list-slices` برای فهرست کردن برش‌های پیکربندی شده بکار می‌رود.
- فرمان `list-slice-info <slicename>` آدرس URL مربوط به کنترلر را نشان می‌دهد که برش مشخصی را کنترل می‌کند. در این فرمان، اطلاعات صاحب برش که آن را ایجاد کرده است و دیگر اطلاعات نیز نشان داده می‌شود.
- فرمان `add-slice <slicename> <controller_url> <email>`، کاربر یک برش را قادر می‌سازد اطلاعاتی را اصلاح کند که به آن برش مربوط می‌شود. تنها پارامترهای `controller_port` و `controller_host`، `contact_email` می‌توانند هنگام نوشتن این خط فرمان تغییر کنند.
- فرمان `list-flowspace`، سیاستهای مربوط به یک برش (با محوریت جریان) را چاپ می‌کند که به نام FlowSpace هم شناخته می‌شود.
- فرمان `remove-slice <slicename>`، یک برش را حذف کرده و تمام FlowSpace‌هایی که با برش فوق در ارتباط هستند، آزاد می‌شوند.
- فرمان `update-slice-password <slicename>`، رمز عبور برشی که در فرمان فوق به جای `slicename` وارد شده است، را تغییر می‌دهد.

- فرمان `add-flowspace <NAME> <DPID> <PRIORITY> <FLOW_MATCH>`، یک سیاست جدید (FlowSpace) ایجاد می‌کند. فرمت `<SLICEACTIONS>` در ادامه توضیح داده می‌شوند.
- فرمان `update-flowspace <NAME> <DPID> <PRIORITY> <FLOW_MATCH>` سیاست یک برش را که در فرمان فوق به جای مقدار NAME مشخص شده است، اصلاح می‌کند. قالب `<SLICEACTIONS>` در ادامه `FLOW_MATCH`, `DPID`, `SLICEACTIONS` و `FLOW_MATCH` توضیح داده می‌شوند.
- فرمان `remove-flowspace <NAME>`، سیاستهای یک برش با نام NAME را حذف می‌کند.

ساختار `FLOW_MATCH`

چگونگی انطباق یک ورودی جریان با یک بسته دریافتی بر روی سوئیچ، در دستور العمل فیلدهای پیش رو توضیح داده شده است. اگر هر کدام از این شرح دستورالعملها، از Syntax یک جریان حذف شوند، فیلد `FLOW_MATCH`، به عنوان یک Wildcard در نظر گرفته می‌شود. بنابراین، اگر تمام این فیلدها حذف شوند، جریان ایجاد شده توسط FlowVisor با تمام بسته‌های وارد شده به سوئیچ، منطبق می‌شود؛ عبارت `any` یا `all` می‌تواند برای تعیین جریانی که با تمام بسته‌ها منطبق است به کار رود.

- دستورالعمل `in_port=port_no`، پورت فیزیکی با مقدار `port_no` را با شماره پورتی که بسته از طریق آن وارد سوئیچ شده (اینترفیس ورودی)، تطابق می‌دهد. پورتهای سوئیچ، زمانی که توسط فرمان `fvctl getDeviceInfo DPID` لیست شدنده، شماره گذاری می‌شوند.
- دستورالعمل `dl_vlan=vlan` تگ VLAN مربوط به استاندارد IEEE 802.1Q را با مقدار VLAN در بسته دریافتی بر روی سوئیچ، انطباق می‌دهد. به منظور منطبق کردن بسته‌هایی که با VLAN تگ گذاری نشده‌اند می‌توانید مقدار `0xffff` را به عنوان ارزش پارامتر `vlan` مشخص کنید. در غیر این صورت یک ارزش عددی از ۰ تا ۴۰۹۵ را به عنوان VLAN ID (۱۲ بیتی) برای انطباق مشخص کنید.
- دستورالعمل `dl_src=mac`، آدرس MAC مبدأ اترنت را با مقدار `mac` منطبق می‌کند. (صرفاً جهت یادآوری می‌گوییم که این آدرس MAC باید به عنوان ۶ جفت از اعداد هگزادسیمال که با دو نقطه از هم جدا شده‌اند، مشخص شوند. مانند: `00:0A:E4:25:6B:B0`)
- دستورالعمل `dl_dst=mac`، آدرس MAC مقصد اترنت با مقدار `mac` را با بسته‌های دریافتی بر روی سوئیچ، منطبق می‌کند.

- دستورالعمل `dl_type=ethertype` را با بسته‌های دریافتی بر روی سوئیچ، منطبق می‌کند. این مقدار باید با یک عدد صحیح بین ۰ تا ۶۵۵۲۵ مشخص شود. این محتوا می‌تواند بصورت یک عدد دسیمال با بصورت یک عدد هگزادسیمال با پیشوند ۰X نشان داده شود. (به طور مثال، برای انطباق بسته‌های ARP می‌توانید عدد ۰x0806 را به عنوان مقدار `ethertype` استفاده کنید).
- دستورالعمل `[nw_src=ip[/netmask]`، آدرس IPv4 مبدأ با مقدار `ip` را، با بسته‌های دریافتی بر روی سوئیچ، منطبق می‌کند. (مقدار `ip` در قالب یک آدرس IP مشخص می‌شود، به طور مثال ۱۹۲.۱۶۸.۰.۱). مقدار اختیاری `netmask`، مکانیزمی فراهم می‌کند که به موجب آن، عمل انطباق تنها با یک آدرس IPv4 انجام پذیرد. `netmask` به عنوان یک `CIDR style` تلقی شده و ساختاری مانند آدرس ۱۹۲.۱۶۸.۱.۰/۲۴ دارد.
- دستورالعمل `[nw_dst=ip[/netmask]`، آدرس IPv4 مقصد با مقدار `ip` را، با آدرس مقصد بسته دریافتی بر روی سوئیچ، منطبق می‌کند. مقدار `netmask` اجازه می‌دهد تنها آدرس فوق در عملیات انطباق مورد استفاده قرار گیرد. (به طور مثال آدرس ۱۹۲.۱۶۸.۱.۰/۲۴).
- دستورالعمل `nw_proto=proto`، نوع پروتکل IP با مقدار `proto` را با بسته‌های ورودی بر روی سوئیچ، انطباق می‌دهد. `proto` یک ارزش عددی بین ۰ تا ۲۵۵ (به طور مثال عدد ۶ برای سازگاری بسته‌های TCP) را دارا خواهد بود.
- دستورالعمل `nw_tos=tos/dscp`، فیلد ToS/DSCP با مقدار `tos/dscp` را، با تعداد بسته‌های دریافتی بر روی سوئیچ، منطبق می‌کند. این ارزش باید یک عددی بین ۰ تا ۲۵۵ باشد.
- دستورالعمل `tp_src=port`، پورت مبدأ لایه انتقال (به طور مثال TCP، UDP یا ICMP) با مقدار `port` را، با پورت مبدأ لایه انتقال (لایه ۴) بسته‌های دریافتی روی سوئیچ، منطبق می‌کند. ارزش عددی `port`، باید بین ۰ تا ۶۵۵۲۵ (برای بسته‌های TCP یا UDP) یا بین ۰ تا ۲۵۵ (برای بسته‌های ICMP) تعیین شود.
- دستورالعمل `tp_dst=port`، پورت مقصد لایه انتقال (به طور مثال TCP، UDP یا ICMP) با مقدار عددی `port` را، با پورت مقصد لایه انتقال (لایه ۴) بسته‌های دریافتی بر روی سوئیچ، منطبق می‌کند. ارزش عددی `port`، باید بین ۰ تا ۶۵۵۲۵ (برای بسته‌های TCP یا UDP) یا بین ۰ تا ۲۵۵ (برای بسته‌های ICMP) تعیین شود.

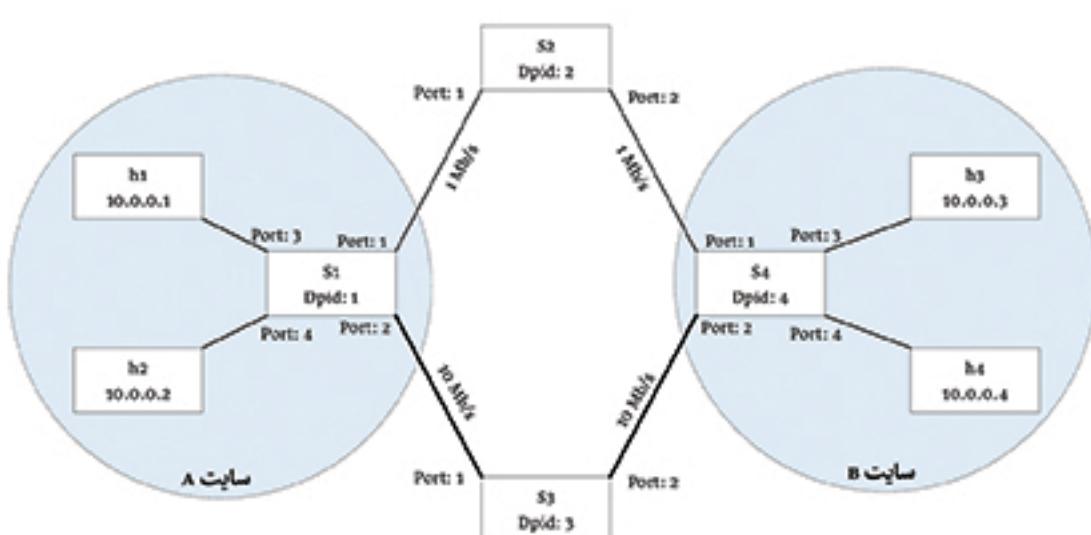
ساختار Action در برش

های برش، فهرستی از برشهایی است که بر یک FlowSpace خاص، کنترل دارند. این فهرست با ویرگول جدا شده و به فرم `[[Slice:slicename1=perm[Slice:slicename2=perm[...]]]]` نوشته

می‌شود. هر بُرش بطور بالقوه سه گونه مجوز دسترسی بر روی FlowSpace دارد که عبارتند از: READ، WRITE و DELEGATE. مجوزها در حال حاضر به عنوان عدد صحیح معین می‌شوند. دستور العمل عبارت است از: `WRITE=4,READ=2,DELEGATE=1`. حال مثالی می‌زنیم. بروش داریم که ساختاری مشابه ساختار رویرو دارد: `Slice:alice=5,bob=2`. بنابراین این بُرش، مجوزهای WRITE و DELEGATE را به بُرش alice (بدلیل آنکه $1+4=5$ بوده، لذا این دو دسترسی به `alice` داده شده) داده و تنها به بُرش bob مجوز Read (بدلیل آنکه به `bob` عدد ۲ اختصاص داده شده است) می‌دهد.

برش دادن شبکه با کنترلر FlowVisor

در این بخش می‌آموزید که چگونه شبکه‌های منطقی بر پستر زیرساخت فیزیکی بسازید، شبکه OpenFlow خود را بُرش دهید و هر بُرش را توسط کنترلر OpenFlow، تحت کنترل داشته باشید. همچنین در طول این فرایند، مفهوم FlowSpace را می‌آموزید و این که چگونه ویژگی کنترل متمرکز روی شبکه مبتنی بر OpenFlow، یک بُرش دهی غنی و انعطاف‌پذیر را در سطح این شبکه فراهم می‌کند. توپولوژی شبکه برای این تمرین در شکل بعد نشان داده شده است که شامل چهار سوئیچ OpenFlow و چهار میزبان مجازی است. سوئیچهای S1 و S4 از طریق سوئیچ S2 و یک پهنهای باند کم (که ۱ Mbps است و در اسکریپتی که در ادامه ذکر شده، به عنوان `LBW_path` مشخص شده است) به هم متصل‌اند؛ همچنین سوئیچهای S1 و S4 توسط سوئیچ S3 و از طریق یک سری از لینکها با پهنهای باند زیاد (که ۱۰ Mbps است و در اسکریپتی که در ادامه ذکر شده است، به عنوان `HBW_path` مشخص می‌شود) به هم وصل می‌شوند:



شکل ۶-۳
توپولوژی شبکه
برای ایجاد بُرشی
از شبکه

نصب Mininet را در فصل ۲ معرفی کردیم و به عنوان بخشی از آزمایشگاه OpenFlow، در فصل ۴، کار با آن را آموختیم. این توپولوژی شبکه را می‌توان با استفاده از اسکریپت زیر (با این فرض که فایل `flowvisor_topo.py` در پوشه جاری در دسترس است) در Mininet ساخت:

```
$ sudo mn --custom flowvisor_topo.py --topo slicingtopo --link tc
--controller remote --mac --arp
```

اسکریپت سفارشی شده Python که ما در این آزمایش مورد استفاده قرار می‌دهیم، یک توپولوژی را به نام `slicingtopo` تعریف می‌کند. این اسکریپت که در ادامه نمایش داده شده است، از طریق قابل دسترس است.

```
#!/usr/bin/python
# flowvisor_topo.py
from mininet.topo import Topo
class FVTopo(Topo):
    def __init__(self):
        # Initialize topology
        Topo.__init__(self)
        # Create template host,
        switch, and link
        hconfig =
        {'inNamespace':True}
        LBW_path = {'bw': 1}
        HBW_path = {'bw': 10}
        host_link_config = {}
        # Create switch nodes
        for i in range(4):
            sconfig = {'dpid': "8016x" % (i+1)}
            self.addSwitch('s%d' % (i+1), **sconfig)
        # Create host nodes (h1, h2, h3, h4)
        for i in range(4):
            self.addHost('h%d' % (i+1), **hconfig)
        # Add switch links according to the topology
        self.addLink('s1', 's2', **LBW_path)
        self.addLink('s2', 's4', **LBW_path)
        self.addLink('s1', 's3', **HBW_path)
        self.addLink('s3', 's4', **HBW_path)
        # Add host links
        self.addLink('h1', 's1', **host_link_config)
        self.addLink('h2', 's1', **host_link_config)
        self.addLink('h3', 's4', **host_link_config)
        self.addLink('h4', 's4', **host_link_config)
topos = { 'slicingtopo': ( lambda: FVTopo() ) }
```

پس از آنکه با استفاده از اسکریپت بالا توپولوژی شبکه خود را ساختید، مرحله بعد، پیکربندی کنترلر FlowVisor است که می‌توانید آن را در یک ترمینال SSH جدید، اجرا کنید. با این فرض که شما در گذشته کنترلر FlowVisor را در یک ماشین مجازی جداگانه نصب کرده‌اید، خط فرمان زیر، این پیکربندی را انجام می‌دهد:

```
$ sudo -u flowvisor fvconfig generate /etc/flowvisor/config.json
```

هنگامی که پیکربندی آماده شد، نام کاربری را `fvadmin` قرار داده و رمز عبور را خالی بگذارید و دکمه `Enter` را فشار دهید. برای به کار انداختن FlowVisor، از فرمان زیر استفاده کنید:

```
$ sudo /etc/init.d/flowvisor start
```

با استفاده از ابزار `fvctl`، کنترلر FlowVisor را فعال کنید. پارامتر خط فرمان `-f`، به فایل رمز عبور اشاره می‌کند. به دلیل آنکه هیچ رمز عبوری برای FlowVisor تعیین نشده است، فایل رمز عبور می‌تواند به `/dev/null` اشاره کند. برای اعمال این تغییر، FlowVisor باید دوباره شروع به کار کند:

```
$ fvctl -f /dev/null set-config --enable-topo-ctrl
$ sudo /etc/init.d/flowvisor restart
```

هنگام شروع به کار کنترلر FlowVisor، تمام سوئیچهای OpenFlow در Mininet باید به آن متصل باشند. با دریافت پیکربندی FlowVisor، اطمینان حاصل کنید که این پیکربندی به درستی اجرا می‌شود:

```
$ fvctl -f /dev/null get-config
```

حال خواهید دید در صورتیکه پیکربندی FlowVisor (در فرمت JSON) بدرستی انجام گرفته شده باشد، خروجی مشابه تصویر زیر نمایش داده می‌شود:

۶-۴ شکل ۶

پیکربندی
FlowVisor
در فرمت JSON

```
{
  "enable-topo-ctrl": true ,
  "flood-perm": {
    "dpid": "all",
    "slice-name": "fvadmin"
  },
  "flow-stats-cache": 30,
  "flowmod-limit": {
    "fvadmin": {
      "00:00:00:00:00:00:01": -1,
      "00:00:00:00:00:00:02": -1,
      "00:00:00:00:00:00:03": -1,
      "00:00:00:00:00:00:04": -1,
      "any": null
    }
  },
  "stats-desc": false,
  "track-flows": false
}
```

با استفاده از فرمان زیر، فهرست برشهای موجود در کنترلر FlowVisor را بدست آورید و مطمئن شوید که فرمان fvadmin (برش پیش فرض) تنها برش موجود است که در خروجی فرمان `fvctl` نشان داده شده است:

```
$ fvctl -f /dev/null list-slices
```

فرمان زیر را صادر کنید تا FlowSpace‌های موجود نمایش داده شوند و اطمینان حاصل کنید که هیچ FlowSpace در کنترلر وجود ندارد:

```
$ fvctl -f /dev/null list-flowspace
```

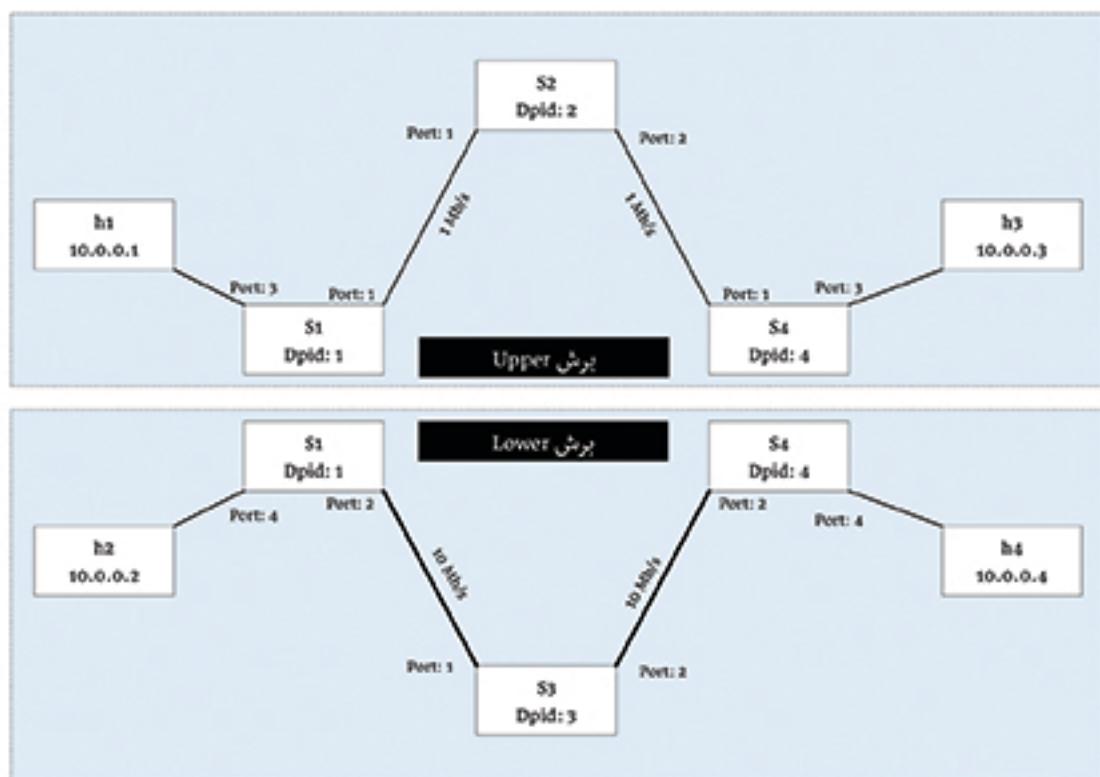
فهرست کردن DataPath این اطمینان را به وجود می‌آورد که تمام سوئیچها به کنترلر FlowVisor متصل هستند. می‌توانید با اجرای فرمان `fvctl` که در ادامه آمده است لیست سوئیچهای متصل شده به کنترلر را مشاهده کنید. قبل از اجرا کردن فرمان، ممکن است چند ثانیه منتظر بمانید. این تأخیر، زمان کافی برای اتصال سوئیچهای S1، S2، S3 و S4 را به FlowVisor می‌دهد.

```
$ fvctl -f /dev/null list-datapaths
```

در مرحله بعد، با فرمان زیر مطمئن شوید که کلیه لینکهای شبکه فعال هستند:

```
$ fectl -f /dev/null list-links
```

خروجی این فرمان، DPID ها و پورتهای مبدأ و مقصد متصل به یکدیگر را نمایش می‌دهد. حال ما آماده ایجاد برش در شبکه آزمایشی خود هستیم. در این آزمایش ما دو برش فیزیکی ایجاد خواهیم کرد که با نامهای برش Upper و برش Lower شناخته می‌شوند. این تپولوژی در شکل زیر نمایش داده شده است:



شکل ۶-۵
برش‌های Upper و Lower در شبکه آزمایشی

هر برش می‌تواند با یک کنترلر جداگانه کنترل شود. در این وضعیت، کنترلر، کلیه ترافیکهای داده‌ای را که مربوط به این برش هستند، کنترل می‌کند. فرمان زیر یک برش با نام `upper` ایجاد می‌کند و آن را در اختیار یک کنترلر قرار می‌دهد. این ارتباط روی پورت ۱۰۰۰۱ (tcp:localhost:10001) انجام می‌گیرد:

```
$ fectl -f /dev/null add-slice upper tcp:localhost:10001
admin@upperslice
```

پس از اجرا، رمز عبور برش را با فشردن کلید `Enter` خالی بگذارید. به همین طریق می‌توانید یک برش با نام `lower` ایجاد کرده و آن را به یک کنترلر بر روی پورت ۲۰۰۰۲ (tcp:localhost:10002) متصل کنید. پس از ایجاد این ارتباط، بار دیگر رمز عبور درخواست می‌شود که شما می‌توانید، با فشردن کلید `Enter`، آن را خالی بگذارید.

```
$ fvectl -f /dev/null add-slice lower tcp:localhost:10002
admin@lowerslice
```

حال با اجرای فرمان `list-slices` مطمئن شوید که برشها با موفقیت اضافه شده‌اند:

```
$ fvectl -f /dev/null list-slices
```

علاوه بر برش پیش فرض `fvadmin`, شما باید بتوانید هر دو برش `Upper` و `Lower` را در وضعیت فعال مشاهده کنید. در مرحله بعد، `FlowSpace`‌ها را ایجاد کنید. `FlowSpace`‌ها، بسته‌های خاصی را به برشهای خاصی مرتبط می‌کنند. زمانی که یک بسته با بیش از یک `FlowSpace` منطبق می‌شود، بحث اولویت‌بندی مطرح می‌شود و به دنبال آن `FlowVisor` بسته را به یک `FlowSpace` با بالاترین اولویت اختصاص می‌دهد. `FlowSpace`‌ها مجموعه‌ای از دستورالعملها با ساختار `field=value` هستند که با ویرگول از یکدیگر جدا می‌شوند. برای یادگیری بیشتر درباره فرمان `add-flowspace` می‌توانید فرمان زیر را تایپ کنید:

```
$ fvectl add-flowspace -h
```

در ادامه، شروع به ایجاد `FlowSpace`‌های مورد نیاز برای سناریو خود می‌کنیم. در ابتدا، یک `FlowSpace` با نام `dpid1-port1` (با ارزش اولویت ۱) ایجاد می‌کنیم که کل ترافیک مربوط به پورت ۱ سوئیچ `S1` را به برش `Upper` اختصاص می‌دهد. این عمل با فرمان زیر انجام می‌پذیرد:

```
$ fvectl -f /dev/null add-flowspace
dpid1-port1 1 1 in_port=1 upper=7
```

اگر به خط فرمان بالا دقت کنید مشاهده می‌کنید که ما به برش `Upper`, تمام مجوزهای `DELEGATE` و `READ` و `WRITE` یعنی $(1+4+2=7)$ را داده‌ایم. با روشنی مشابه، ما یک `FlowSpace` با نام `dpid1-port3` ایجاد می‌کنیم که کل ترافیک روی پورت ۳ از سوئیچ `S1` را در اختیار برش `Upper` قرار می‌دهد.

```
$ fvectl -f /dev/null add-flowspace
dpid1-port3 1 1 in_port=3 upper=7
```

سپس، با استفاده از ارزش `any`, می‌توانیم یک `FlowSpace` برای انطباق کل ترافیک در یک سوئیچ، ایجاد کنیم. در این حالت کلیه ترافیک‌هایی که به سوئیچ `S2` وارد می‌شوند، مربوط به برش `Upper` خواهند بود. بنابراین، سوئیچ `S2` را با فرمان زیر به برش `Upper` اضافه می‌کنیم:

```
$ fvectl -f /dev/null add-flowspace
dpid2 2 1 any upper=7
```

حال، دو FlowSpace دیگر با نامهای `dpid4-port1` و `dpid4-port3` را ایجاد کرده تا پورتهای ۱ و ۳ از سوچی S4 را به بُرش Upper اضافه کنیم:

```
$ fvctl -f /dev/null add-flowspace dpid4-port1 4 1 in_port=1
upper=7
$ fvctl -f /dev/null add-flowspace dpid4-port3 4 1 in_port=3
upper=7
```

با فرمان زیر مطمئن شوید که این FlowSpace‌ها به درستی اضافه شده‌اند:

```
$ fvctl -f /dev/null list-flowspace
```

باید تمام FlowSpace‌هایی را که ایجاد کرده‌اید مشاهده کنید. (در کل ۵ عدد)

```
$ fvctl -f /dev/null add-flowspace dpid1-port2 1 1 in_port=2
lower=7
$ fvctl -f /dev/null add-flowspace dpid1-port4 1 1 in_port=4
lower=7
$ fvctl -f /dev/null add-flowspace dpid3 3 1 any lower=7
$ fvctl -f /dev/null add-flowspace dpid4-port2 4 1 in_port=2
lower=7
$ fvctl -f /dev/null add-flowspace dpid4-port4 4 1 in_port=4
lower=7
```

سپس FlowSpace‌های مربوط به بُرش Lower را همانند فرمانهای اعمال شده برای بُرش Upper، ایجاد می‌کنیم. پس از آنکه این FlowSpace‌ها را ایجاد کردید، با استفاده از فرمان زیر اطمینان حاصل کنید که این FlowSpace‌ها به درستی اضافه شده‌اند.

```
$ fvctl -f /dev/null list-flowspace
```

حال می‌توانید دو کنترلر OpenFlow را بر روی کامپیوتر خود راه‌اندازی کنید. این دو کنترلر، روی پورتهای ۱۰۰۰۱ و ۱۰۰۰۲ مربوط به بُرشهایی Upper و Lower گوش داده و آماده سرویس دهی هستند. شما باید همچنین NetApp کوچکی بنویسید که مسیرهای مبتنی بر آدرس MAC مقصد را در جدول جریان سوچی نصب کند. بدین ترتیب پس از تأخیری کوتاه، هر دو کنترلر باید به FlowVisor متصل شوند. از این پس می‌توانید بررسی کنید که میزبان h1 می‌تواند میزبان h3 را Ping کند ولی نمی‌تواند میزبانهای h2 و h4 را Ping کند (و بالعکس).

فرمان زیر را در کنسول Mininet اجرا کنید:

```
mininet> h1 ping -c1 h3
mininet> h1 ping -c1 -W1 h2
mininet> h1 ping -c1 -W1 h4
```

سپس بررسی کنید که h2 می‌تواند h4 را Ping کند ولی نمی‌تواند h1 و h3 را Ping کند (و بالعکس). فرمان زیر را در کنسول Mininet اجرا کنید:

```
mininet> h2 ping -c1 h4
mininet> h2 ping -c1 -W1 h1
mininet> h2 ping -c1 -W1 h3
```

در نتیجه با استفاده از پورتهای سوچیج، به یک برش ساده از شبکه دست پیدا می‌کنیم. با این حال، با تعریف دیگر قوانین برش و توسعه NetApp‌های گوناگون، می‌توانید سرویسهای جالب تر و نوآورانه‌تری برای هر برش فراهم کنید. به عنوان مثال می‌توانید ترافیکها را جدا کرده و در سرتاسر برش‌های Upper و Lower شبکه، با آنها براساس این جدایی رفتار کنید. این موضوع را به عنوان یک تمرین به شما واگذار می‌کنیم.

خلاصه

در این فصل مفهوم مجازی‌سازی شبکه و بطور کلی نقش و عملکرد کنترلر FlowVisor را به عنوان ابزاری برای برش دادن شبکه‌های مبتنی بر OpenFlow شرح دادیم. FlowVisor API و ساختارهای مرتبط با آن، برای انطباق جریان و Action‌های مرتبط با برش دادن شبکه، معرفی شد و سپس بر این اساس یک آزمایش مرحله به مرحله، توضیح داده شد. حال، با ابزارهایی آشنا هستید که می‌توانند برای برش دادن شبکه و کنترل هر بروش به روشنی خلاقانه، مورد استفاده قرار بگیرند. در فصل بعد، بصورت کلی به بررسی نقش OpenFlow و SDN در رایانش آبری (Cloud Computing)، خواهیم پرداخت.