



# شبکهٔ تعریف شده با نرم افزار OpenFlow مبتنی بر

تجربه‌ای جدید با پلتفرم‌ها و ابزارها، برای ساختن شبکه‌ای بر پایه OpenFlow

مترجم: بهنام صابری

نویسنده: سیامک عضدالمکی

# شبکه تعریف شده با نرم افزار

## مبنی بر OpenFlow

نویسنده: سیامک عضدالمکی

مترجم: بهنام صابری

طراح و صفحه آرا: مهران ساداتی

ناظران فنی: رضا گوینده، مهران ساداتی

بازخوانان فنی: مهران ساداتی، پوریا غروی، محمدحسین فرجی

نوبت چاپ: چاپ اول بهار ۱۳۹۶

ناشر: انتشارات مهدی (عج)

چاپ: مهدی (عج)

شابک:

تیراز:

قیمت:

فروشگاه: ضلع جنوب شرقی میدان انقلاب، رویروی ایستگاه BRT، بازار بزرگ کتاب، ش. ۱۶، تلفکس: ۰۶۴۸۷۵۳۸، ۰۶۴۸۷۵۱۷  
دفتر: خیابان منیری جاوید (اردبیلهشت)، بالاتر از خیابان جمهوری، پ. ۱۹، طبقه ۲، واحد ۱، تلفکس: ۰۶۴۶۳۶۸۹



## تقدیم به

هادر عزیز و مهربانم که در تمامی لحظات زندگی به من یاری رساند  
پدرم که روشن همواره در کنار من بوده و  
فواهر و برادرانم که گرمای بودنشان ارامش بخش من بوده است.



## شرحی بر نویسنده

دکتر سیامک عضدالملکی دانش آموخته رشته مهندسی کامپیوتر از دانشگاه تهران در مقطع کارشناسی بوده است. او سپس اولین مدرک کارشناسی ارشد خود را در رشته معماری کامپیوتر از دانشگاه آزاد اسلامی دریافت کرد. پس از اخذ مدرک فوق، در شرکت داده پردازی ایران (شرکت IBM سابق) به عنوان برنامه نویس شروع بکار کرد و طی ۹ سال حضور خود در این شرکت، بعنوان مهندس میستم و مهندس ارشد تیم تحقیق و توسعه مشغول بکار بود. او توانست در سال ۲۰۰۶ دومین مدرک کارشناسی ارشد خود را از Carnegie Mellon دریافت کند. پس از فارغ التحصیلی از دانشگاه، در سال ۲۰۰۷ بعنوان پژوهشگر و برنامه نویس به موسسه AIT پیوست و این در حالی بود که او در حال تحصیل در مقطع دکتری بود. در آگوست ۲۰۱۰، او به گروه تحقیقاتی دانشگاه Essex پیوست و مدیریت تیم تحقیق را به او سپردند. یک سال بعد توانست مدرک دکتری خود را از دانشگاه UPC اسپانیا دریافت کند. پس از آن بعنوان محقق در حوزه فنی در چندین پژوهه مرتبط با اتحادیه اروپا حضور داشت. از سال ۲۰۱۰ شبکه های تعریف شده با نرم افزار (SDN) را بعنوان یکی از علایق تحقیقاتی خود برگزید. ایشان بیش از ۵۰ مقاله علمی در همایش ها، زورنال ها و کتاب های بین المللی به چاپ رساندند. سازمان GWDG، یکی از مراکزی بود که ایشان بعنوان محقق ارشد به آن پیوست و در آن به فعالیت های مرتبط با SDN پرداخت. او یکی از اعضای اصلی ACM و عضو ارشد IEEE بود. این دانشمند و محقق فرهیخته در سال ۲۰۱۵ از بین ما رفت. روحش شاد و یادش همواره گرامی باد.



## فهرست مطالب

### فصل ۱. معرفی OpenFlow

۵ .....	شناخت و درک "شبکه تعریف شده با نرم افزار" با چاشنی OpenFlow
۸ .....	فعالیت های پیرامون SDN و OpenFlow
۹ .....	اجزای سازنده
۱۷ .....	پیغام های OpenFlow
۱۹ .....	پیغام های کنترلر - به - سوئیچ
۲۲ .....	پیغام های متقارن (Symmetric)
۲۳ .....	پیغام های ناهمزمان (Asynchronous)
۲۵ .....	اینترفیس شمالی
۲۶ .....	خلاصه

### فصل ۲. پیاده سازی سوئیچ

۲۷ .....	سوئیچ مرجع OpenFlow
۳۱ .....	پیغام های ناهمزمان (Asynchronous Messages)
۳۲ .....	پیغام های متقارن (Symmetric Messages)
۳۳ .....	پیاده سازی های سخت افزاری
۳۴ .....	سوئیچ های مبتنی بر نرم افزار
۳۵ .....	آزمایشگاه Mininet با OpenFlow
۳۷ .....	شروع کار با Mininet
۴۰ .....	تجربه ای با Mininet
۴۶ .....	خلاصه

### فصل ۳. کنترلر های OpenFlow

۴۷ .....	کنترلر های SDN
۵۰ .....	پیاده سازی برخی کنترلر های موجود
۵۱ .....	کنترلر های NOX و POX
۵۲ .....	اجرای یک اپلیکیشن POX
۶۰ .....	NodeFlow
۶۳ .....	Floodlight

۶۵	.....	<b>OpenDaylight</b>
۶۶	.....	کنترلرهای ویژه
۶۸	.....	<b>خلاصه</b>
 <b>فصل ۴. راهاندازی محیط</b>		
۶۹	.....	شناخت و درک آزمایشگاه <b>OpenFlow</b>
۷۵	.....	کنترلرهای بیرونی
۷۶	.....	تمکیل آزمایشگاه <b>OpenFlow</b>
۸۱	.....	<b>OpenDaylight</b>
۸۲	.....	کنترل <b>ODL</b>
۸۵	.....	آزمایشگاه SDN مبتنی بر <b>ODL</b>
۹۰	.....	<b>خلاصه</b>
 <b>فصل ۵. توسعه "Net App"</b>		
۹۱	.....	شماره ۱- سوئیچ یادگیرنده اترنت <b>Net App</b>
۹۶	.....	ساخت یک سوئیچ یادگیرنده
۱۰۰	.....	شماره ۲- یک فایروال ساده <b>Net App</b>
۱۰۳	... <b>OpenDaylight</b> در <b>Simple Forwarding</b> شبکه <b>Net App</b>	
۱۰۶	.....	<b>خلاصه</b>
 <b>فصل ۶. ایجاد برش شبکه</b>		
۱۰۷	.....	مجازی‌سازی شبکه
۱۰۹	.....	<b>FlowVisor</b>
۱۱۰	.....	<b>FlowVisor API</b>
۱۱۲	.....	ساختار <b>FLOW_MATCH</b>
۱۱۳	.....	ساختار <b>Action</b> در برش
۱۱۴	.....	برش دادن شبکه با کنترلر <b>FlowVisor</b>
۱۲۲	.....	<b>خلاصه</b>
 <b>فصل ۷. OpenFlow در رایانش آبری</b>		
۱۲۳	.....	<b>Neutron و OpenStack</b>
۱۲۸	.....	معماری شبکه‌سازی <b>OpenStack</b>

پلاگین‌های Neutron

۱۳۱ ..... خلاصه

## فصل ۸. منابع متن باز

۱۳۸ ..... سوئیچ‌ها

۱۲۸ ..... Open vSwitch

۱۲۹ ..... Pantou

۱۴۲ ..... Indigo

۱۴۳ ..... LINC

۱۴۴ ..... XORPlus

۱۴۵ ..... OF13SoftSwitch

۱۴۶ ..... کنترلرها

۱۴۷ ..... Beacon

۱۴۸ ..... Floodlight

۱۴۹ ..... Maestro

۱۴۹ ..... Trema

۱۴۸ ..... FlowER

۱۴۹ ..... Ryu

۱۴۹ ..... متفرقه‌ها

۱۴۹ ..... FlowVisor

۱۵۰ ..... Avior

۱۵۱ ..... RouteFlow

۱۵۲ ..... CBench و OFlops

۱۵۰ ..... OSCARS

۱۵۰ ..... Twister

۱۵۰ ..... FortNOX

۱۵۰ ..... Nettle

۱۵۰ ..... Frenetic

۱۵۰ ..... OESS

۱۵۰ ..... خلاصه

۱۵۹	نمايه
۱۶۵	واژه‌نامه

## پیشگفتار مترجم

ترجمه این کتاب را با نام خدایی آغاز نمودم که در تمامی لحظه‌های زندگی، بودنش را حس کردم. در حدود ۵ سال از آشنایی من با مفهوم SDN می‌گذرد. در طی این سال‌ها SDN از یک مفهوم آزمایشگاهی به یک تحول بزرگ در صنعت شبکه تبدیل شد. SDN را می‌توان همچون خونی تازه در رگهای شبکه برشمرد که جذابیت‌های زیادی برای اهل فن به همراه دارد. پروژه‌های عظیمی که شرکت‌های معظم همچون گوگل و مایکروسافت در بهره‌گیری از SDN داشته‌اند، نشان از اهمیت موضوع دارد. بحثی که اگرچه شاید یک راه حل جامع برای کلیه شبکه‌های بزرگ و کوچک به همراه نداشته باشد، اما از درجه اهمیت بالایی در شبکه‌های بزرگ و مراکز داده برخوردار است. در کشور ما تلاش‌های پراکنده‌ایی حول SDN، در محافل دانشگاهی وجود داشته است. اما مسلماً با برنامه ریزی‌های مدون، می‌توان پیشرفت‌های این حوزه را شاهد بود و از SDN در شبکه‌های مخابراتی و شبکه‌های IP بصورت بومی بهره‌مند شد.

این کتاب سعی دارد آغازی باشد برای متخصصین این حوزه در خصوص آشنائی با SDN / Open-Flow و ابزارهای مرتبط با آن. برای مطالعه این کتاب لازم است دانشی در زمینه‌های مختلف داشته باشید. این دانش شامل آگاهی از مقاهم شبکه همچون TCP/IP، اینترنت و... است. همچنین تجربه برنامه‌نویسی زبان‌های سطح بالا همچون Java، C++ و Python می‌تواند به درک مطلوبتر از مقاهم کتاب فوق کمک کند.

مسیر آماده‌سازی این کتاب سخت اما شیرین بود. در فرایند ترجمه این کتاب سعی شد تا حد امکان کلمات فنی ترجمه نشود یا از کلماتی استفاده شود که هم مفهوم اصلی را رسانده و هم مناسب باشد. با این حال در صورتیکه دوستان اهل فن در خصوص بخش‌های مختلف، پیشنهاد و انتقادی دارند که به بهبود کیفیت کتاب حاضر کمک کنند، بنده را بعنوان شاگرد خود قبول کرده و از نظرات خود، بهره مند سازند.

در بین کارشناسان کشورمان، SDN با عنوانی مختلفی همچون "شبکه‌های مبتنی بر نرم‌افزار"، "شبکه‌های نرم‌افزاری" یا "شبکه‌های نرم‌افزار محور" نام‌گذاری شده، اما به نظر می‌رسد، با توجه به اینکه SDN مبتنی بر ایجاد و تعریف شبکه‌ها، با استفاده از سرویس‌های نرم‌افزاری است، نزدیک ترین نام همان "شبکه‌های تعریف شده با نرم‌افزار" است.

در انتهای می‌بایست از برخی دوستان بسیار خوبم که مرا در آماده‌سازی این کتاب باری رساندند، قدردانی کنم. ابتدا از جناب آقای مهندس داود قطره سامانی بعنوان یک دوست و یک متخصصین بسیار کارکشته قدردانی می‌کنم که ایده اولیه کار بر روی این کتاب را در یک روز مسد زمستانی مطرح کرد. سرکار خانم مهندس آرزو غفوری که در طی آماده‌سازی و نگارش این کتاب با

صیر و حوصله تمام به من یاری رساندند. همچنین از دوست بسیار خوب و مدیر خوش فکر جناب آقای مهندس محمد مفتح تشرک می کنم که همراهی فنی ایشان همواره باعث دلگرمی من بود. جناب آقای میر علی غروی که راهنمایی های ایشان در بخش ترجمه و نگارش فارسی، کیفیت اثر فوق را ارتقا داد. در ادامه تشکر خود را تقدیم به دوستان بسیار عزیزم مهندس پوریا غروی، مهندس محمد حسین فرج بیک و بخصوص مهندس مهران ساداقی می کنم که در بازخوانی فنی کتاب، کمک های بسیار کارمند و شایانی نمودند. اما در انتهای احترام خود را تقدیم به نویسنده این کتاب، استاد ارجمند جناب آقای دکتر سیامک عضدالملکی می کنم، که در طی مدت بسیار کوتاه آشنایی با ایشان شیفتہ رفتار، صداقت و همراهی شان شدم. این کتاب سعی داشته است که در حد توان خود، اسم این دانشمند ارزشمند کشورمان را زنده نگه دارد.

بهنام صابری زمستان ۱۳۹۵

## مقدمه

جداسازی کنترل شبکه از تجهیزات شبکه‌ای پایه و اساس SDN به شمار می‌رود. مفهوم SDN الگوی نسبتاً نوپایی است که منجر به آن شده که قابلیت‌های کنترل شبکه (که به Control Plane شهرت دارد) از بخش داده (با همان Data Plane) جدا شده و این جداسازی را قابل برنامه‌ریزی کرده است. مهاجرت Control Plane و انتقال آنها به بیرون از تجهیزات کامپیوتوری و قراردادی آن برروی کنترلرهای بیرونی منجر به آن شده که از نگاه اپلیکیشن‌ها، یک شبکه واحد مشاهده شود. این جداسازی، راه را برای ایجاد معماری‌های شبکه‌ای نوین، منعطف و قابل برنامه‌ریزی هموار کرده است. در کنار این ویژگی‌ها، معماری SDN‌هایی را ارائه می‌دهد که پیاده‌سازی سرویس‌های شبکه‌ای را ساده می‌کند. بر همین اساس سازمان‌ها، اپراتورهای شبکه و افراد ذینفع در این حوزه می‌توانند دستاوردهای فوق العاده‌ایی برای برنامه نویسی، اتوماتیک‌سازی و کنترل شبکه بسته به نیاز کسب و کار خود، بدست آورند. پروتکل OpenFlow اولین اینترفیس استاندارد است که قابلیت‌های زیادی را برای رسیدن به مفهوم SDN ارائه می‌دهد. این کتاب به بررسی زیر ساخت‌های OpenFlow خواهد پرداخت. سپس به راه‌اندازی سوئیچ‌ها و کنترلرهای OpenFlow می‌پردازد و همچنین Net App‌ها را بعنوان اپلیکیشن‌هایی برای اجرای برخی خصوصیات شبکه مطرح می‌کند. مجازی‌سازی شبکه، OpenFlow در رایانش آبری و پروژه‌های پیرامون این پروتکل، از جمله مباحث مطروحه در این کتاب است. اگر تشنۀ آشنایی با مفاهیم SDN و OpenFlow هستید، این کتاب می‌تواند شروع خوبی برای شما باشد تا بتوانید بیشتر با آنها آشنا شوید. با اینکه این کتاب به مفاهیم بنیادین OpenFlow در SDN می‌پردازد اما بیشتر می‌توان از آن بعنوان یک راهنمای آموزشی نام برد و نه یک کتاب مرجع. مهندسین شبکه، مدیران شبکه، توسعه دهنده‌گان نرم‌افزارهای سیستمی و هر شخصی که به OpenFlow علاقه‌مند باشد، می‌تواند از این کتاب بهره ببرد.

برای آنکه این کتاب برای شما مفید واقع شود، نیاز به سطح متوسطی از تجربه و دانش شبکه در مباحثی همچون TCP/IP، اترنت و فعالیت‌های معمول در حوزه شبکه است. همچنین داشتن تجربه برنامه‌نویسی در حوزه زبان‌های سطح بالا و زبان‌های اسکریپت نویسی همچون C/C++، جاوا و Python می‌تواند به درک بهتر بخش‌های مختلف کتاب کمک کند. تجربه کار با ماشین‌های مجازی و محیط شبکه‌های مجازی، باعث تسهیل در آماده‌سازی برخی از آزمایشها می‌شود. بدلیل وجود برخی فعالیت‌های موجود در کتاب، نیاز به کامپیوتوری با ۲GB حافظه RAM و ۱0GB فضای دیسک نیاز دارد. همچنین به یک ارتباط اینترنت برای دانلود برخی ابزار، از طریق ماشین‌های مجازی نیاز است.

## دستورالعمل‌های استفاده بهتر از کتاب

در راه نگارش این کتاب، نمادها و نشانه‌هایی مورد استفاده قرار گرفته است که هدف از بهره‌گیری از این ساختارها و قواعد، سهولت مطالعه کتاب و شفاف‌تر شدن برخی مباحث بوده است. در ادامه به این نشانه‌ها و دستورالعمل‌ها پرداخته خواهد شد:

- در این کتاب برخی کلمات فنی، ترجمه شده، که ممکن است برای اولین بار با این عبارت برخورد کرده باشد، برای آنکه خواننده از اهمیت کلمه فوق آگاه باشد و بداند که آن کلمه در واژه‌نامه انتهای کتاب نیز موجود است، نخستین باری که کلمه ذکر شده است، عبارت انگلیسی بهمراه معنی آن نوشته شده است. در باقی نقاط کتاب، تنها عبارت ترجمه شده بصورت **Bold** نمایش داده شده است.
- حروفی که با رنگ **آبی رنگ** (کم رنگ) و با فونتی متفاوت نمایش داده شده اند، کلماتی همچون کدها، رمز عبور، دستورالعمل‌ها، ورودی‌های کاربران، آدرس‌های URL خروجی برنامه و پسوند فایل‌ها است. در صورتی که این حروف به صورت **Bold** نمایش داده شوند به مفهوم خروجی‌های یک کد برنامه می‌باشند.
- کلماتی که بصورت ایتالیک نمایش داده شده‌اند، منوهای کلیک خور نرم‌افزارها هستند.

- [  ] •  
یادداشت‌ها به این صورت در متن ظاهر می‌شوند.
- [  ] •  
نکته‌ها به این صورت در متن ظاهر می‌شوند.

## ۱

# معرفی OpenFlow

برای رسیدن به درکی مطلوب از نقش پروتکل OpenFlow و اجزای تشکیل دهنده آن در معماری شبکه و همچنین کاربرد آن در توسعه اپلیکیشن‌های شبکه مبتنی بر OpenFlow، ارائه مقدمه‌ای مختصر و کوتاه درباره این پروتکل نوپا و نحوه کارکرد آن ضروری به نظر می‌رسد. این فصل دانشی را در اختیار خواننده قرار می‌دهد که از طریق آن بتواند با مفاهیم OpenFlow آشنا شده و نحوه کارکرد آن را در محیط‌های آزمایشگاهی و واقعی، به درستی درک کند. پروتکل OpenFlow را می‌توان به عنوان یکی از نخستین گام‌های رسیدن به مفهوم SDN<sup>۱</sup> برشمرد. بنابراین، قبل از آنکه بخواهیم پروتکل OpenFlow را تحلیل و بررسی نماییم، شایسته است مقدمه‌ای کوتاه بر SDN و مباحث پیرامون آن داشته باشیم.

**شناخت و درک "شبکه تعریف شده با نرم افزار" با چاشری OpenFlow**

شبکه تعریف شده با نرم افزار که امروزه از آن به اختصار SDN یاد می‌شود، در بسیاری از بحثها و در بین اهل فن، ایده‌ای انقلابی و جدید در حوزه شبکه‌های کامپیوتی محسوب می‌گردد. این ایده، نوید بخش سهولتی بسیار چشمگیر در کنترل شبکه و مدیریت آن است و از آنجایی که ویژگی برنامه پذیر بودن را دارا است، نوآوری‌هایی را به واسطه این قابلیت میسر می‌سازد. شبکه‌های کامپیوتی معمولاً از تعداد زیادی ابزار شبکه (مانند سوئیچ‌ها، مسیریاب‌ها، فایروال‌ها و غیره)، به همراه تعداد زیادی پروتکل پیچیده نرم افزاری که در داخل این شبکه‌ها بکار رفته و تعییه شده‌اند، تشکیل شده است. مهندسان شبکه مسئول پیکربندی سیاست‌هایی هستند که بتوانند طیف وسیعی از رویدادهای شبکه و طرح‌های کاربردی را پاسخگو باشند. آنها سیاست‌های سطح بالا (و برخی اوقات مفهومی) را به صورت دستی به

دستورات پیکربندی سطح پایین تبدیل می‌کنند. این وظایف پیچیده اغلب با دسترسی به ابزارهای بسیار محدود انجام می‌پذیرد. بنابراین، کنترل، مدیریت و ارتقاء کارائی شبکه، از جمله وظایف بسیار چالش برانگیز و خطأ پذیر بوده که بر دوش این مهندسان سنگینی می‌کند.

چالش دیگری که در حوزه شبکه مطرح می‌شود، موضوعی است که مهندسان شبکه و محققان از آن تحت عنوان<sup>۱</sup> Internet Ossification یاد می‌کنند. اینترنت از دو جنبه حائز اهمیت است: جنبه اول بستر اینترنت است که از زیر ساخت استقرار یافته بسیار گستره‌ای تشکیل شده است. جنبه دوم نقش آن در زندگی انسان بوده که روز به روز حیاتی تر و پر اهمیت تر می‌شود. در نهایت می‌توان اینگونه نتیجه‌گیری نمود که وابستگی انسان به اینترنت نیز بیشتر از قبل شده است. اما اینترنت با مشکل سکون و عدم پیشرفت در لایه‌های مختلف مواجه بوده که این مساله، هم در زیر ساخت فیزیکی و هم در پروتکل‌های این حوزه به چشم می‌خورد. با توجه به پیشرفت‌های تر شدن و پیچیدگی‌های نرم‌افزاری، وضعیت کنونی اینترنت و زیر ساختهای آن، قادر به روپارویی با چالش‌های نوظهور در این حوزه نیست.

مفهوم شبکه‌های برنامه‌پذیر (Programmable Networks)، به عنوان روشی برای تسهیل در تکامل اینترنت پیشنهاد شده است. این مفهوم که آنرا SDN می‌نامیم، الگوی شبکه‌سازی جدیدی است که در آن سخت افزار ارسال کننده (Forwarding Hardware) از بخش تصمیمات کنترلی (Control Plane) جدا شده است.<sup>۲</sup> بحثی که SDN به آن می‌پردازد تغییر وضعیت Control Plane است. به موجب آن، Control Plane که تا به امروز داخل هر دستگاه شبکه تعییه شده بود (برای مثال، در سوئیچ‌های اترنت) از آن بصورت فیزیکی جدا شده است. اما از لحاظ منطقی متمرکز عمل کرده و شبکه را به صورت یکپارچه مدیریت می‌کند. این تغییر وضعیت امکانی را فراهم می‌کند که به موجب آن، زیرساخت اصلی شبکه، از منظر اپلیکیشنی که نقش کنترل کننده شبکه را دارد، بصورت یک لایه انتزاعی (Abstraction Layer) دیده شود. این جداسازی باعث می‌شود معماری شبکه علاوه بر به

<sup>۱</sup> عبارت Internet Ossification با سفت و سخت شدن اینترنت، اصطلاحی است که به والعیت غیرقابل انعطاف بودن تکنولوژی‌های حال حاضر تشکیل دهنده اینترنت در بعد انتقال اشاره دارد. این تکنولوژی‌ها منجر به محدودیت‌های زیادی برای کاربران و خدمات بر روی این بستر شده است. عدم امکان پردازه برداری از Multicast در سطح متوسط و وسیع، ضعیف بودن مکانیزم‌های QoS و مشکلات مربوط به مهندسی ترافیک از جمله این محدودیت‌ها است. (متترجم)

<sup>۲</sup> بخش‌های Data Plane و Control Plane را می‌توان قلب سوئیچ‌ها و مسیریاب‌ها برای انتقال بسته‌های IP نام برد. Data Plane که با نام‌هایی همچون Forwarding Plane و User Plane هم نامیده می‌شود، وظیله انتقال ترافیک داده کاربر را بر عهده دارد. Control Plane کامپیوتنتی در مسیریاب‌ها است که وظیله انتقال سیگنال‌های ترافیک و مسئولیت مسیریابی را بر عهده دارد. من توان اینگونه بیان کرد که Control Plane با استفاده از پروتکل‌های مسیریابی مختلف همچون OSPF با EIGRP، وظیله ایجاد نشنه‌ای از شبکه را بر عهده داشته که این نشنه از طریق جمع آوری داده از مسیریاب‌های مجاور محقق می‌شود. سپس این نشنه توسط Data Plane مورد استفاده قرار گرفته و براساس این داده‌ها، بسته به مقصد ارسال می‌شود. هر دو کامپیوتنت Control Plane در داخل سوئیچ قرار دارند، اما همانطور که در متن اصلی به آن اشاره خواهد شد، در معماری نوینی که SDN مطرح می‌کند، کامپیوتنت Control Plane از داخل سوئیچ خارج شده و روی یک سخت افزار ثانویه به نام کنترلر (بصورت متمرکز) قرار گرفته است. (متترجم)

دست آوردن قابلیت برنامه پذیری، قابل انعطاف‌تر، مقرن به صرفه‌تر و خلاقانه‌تر باشد. این در حالی است که جداسازی فوق منجر به عدم وابستگی به هیچ سازنده و برند خاصی (Vendor-Agnostic) شده است.

علاوه بر انتزاع شبکه، معماری SDN با فراهم آوردن مجموعه‌ای از<sup>۱</sup> API‌ها، باعث تسهیل در پیاده‌سازی سرویس‌های شبکه می‌شود. از جمله این سرویس‌ها می‌توان به سرویس‌های مسیریابی شبکه، Multicast، امنیت، کنترل دسترسی، مدیریت پهنای باند، مهندسی ترافیک، QoS، بهره‌وری انرژی و فرم‌های گوناگون مدیریت سیاستها، اشاره کرد. در SDN، هوش شبکه به طور منطقی در کنترلرهای مبتنی بر نرم افزار قرار داده شده است (Control Plane) و تجهیزات شبکه همچون سوئیچ‌ها تبدیل به تجهیزات ساده‌ای شده‌اند که تنها وظیفه ارسال بسته (Data Plane) را دارند. ارتباط بین بخش هوش شبکه یعنی کنترل و بخش تجهیزات ساده که تنها فرمانبرداری می‌کنند، از طریق یک اینترفیس باز، برنامه نویسی می‌شود. یکی از اولین پروتکل‌هایی که روی این اینترفیس باز عمل می‌کند، OpenFlow نام دارد.

جدایی سخت افزار ارسال کننده از بخش کنترل خود، باعث تسهیل در اجرای پروتکل‌ها و اپلیکیشن‌های جدید، ترسیم شبکه و مدیریت مستقیم آن شده و همچنین باعث ادغام Middle Box‌های<sup>۲</sup> گوناگون با نرم افزار می‌شود. به جای اجبار در اجرای روش‌ها و پروتکل‌های کنونی برای مجموعه در هم تنیده‌ای از ابزارهای متفاوت، شبکه مبتنی بر SDN، به یک سخت افزار ارسال کننده ساده، به همراه کنترل کننده (های) تصمیم ماز شبکه تقلیل پیدا می‌کند. اما این سخت افزار ارسال کننده، ویژگی‌هایی را در خود داشته که در ادامه ذکر می‌گردد:

۱. یک جدول جریان (Flow Table) حاوی سطوحی است که قوانینی (Rules) را دنبال می‌کند. از این پس این سطوح را ورودی‌های جریان (Flow Entries) می‌نامیم. این ورودی‌های جریان از سه جزء اصلی تشکیل شده‌اند که در ادامه این فصل بیشتر به آنها خواهیم پرداخت. اولین جزء فیلد هدر (Header Field) نام دارد که عناصر آن مقادیری از لایه‌های ۲ تا ۴ پروتکل TCP/IP را تشکیل می‌دهند. این مقادیر جهت انطباق (Match) با بسته‌های ورودی به سوئیچ، استفاده می‌شوند. جزء دوم شامل تعدادی دستور العمل است که Action نامیده شده و مشخص کننده وضعیت آنی بسته‌های منطبق (Match) شده هستند. جزء آخر، شمارنده‌ها (Counters) بوده که تعداد بسته‌های منطبق شده با یک سطر از جدول جریان را شمارش می‌کنند.

<sup>۱</sup> Application Programming Interface

<sup>۲</sup> تجهیزات Middle Box را می‌توان Appliance هایی نام برد که در شبکه‌های کامپیوتی جهت تبدیل محتوای بسته، فیلتر کردن آنها و دیگر محاسبات روی ترافیک شبکه استفاده می‌شود. همچنین از Middle Box ها برای اهدافی به غیر از ارسال بسته، می‌توان استفاده کرد. نمونه‌های متقابل برای این نوع از تجهیزات، فایروالها هستند که وظیفه جلوگیری از عبور ترافیک‌های ناشناخته، ترجمه آدرس‌های IP (NAT) و... را بر عهده دارند. از دیگر انواع Middle Box ها می‌توان IDS و WAN Optimizer و Load Balancer را نام برد. (متوجه)

۲. یک پروتکل لایه انتقال که در خصوص ورودی‌های جدیدی که در جدول جریان نیستند، با یک کنترل ارتباط امن برقرار می‌کند. این پروتکل، چیزی جز پروتکل OpenFlow نیست.

### فعالیت‌های پیرامون SDN و OpenFlow

در حالی که OpenFlow در بازه زمانی کوتاه از تولیدش، توانست توجه بخش صنعت را بصورت چشمگیری به خود جلب کند، شایسته است ذکر کنیم، ایده شبکه‌های برنامه‌پذیر و همچنین جدایی Data Plane از Control Plane، از مالها پیش مطرح بوده است. شرکت OPENSIG<sup>۱</sup> در سال ۱۹۹۵ کارگاه‌های آموزشی برپا کرد تا شبکه‌های ATM، اینترنت و شبکه‌های تلفن همراه را بازتر، توسعه پذیرتر و برنامه‌پذیرتر کند. با الهام از این ایده‌ها، یک گروه کاری به نام<sup>۲</sup> IETF Label Switch<sup>۳</sup> تشکیل شد تا با استفاده از پروتکل GSMP<sup>۴</sup> بتواند Active Network را کنترل کند. فعالیت‌های این گروه به نتیجه رسید و آنها توانستند پروتکل GSMPv3 را در ماه ژوئن ۲۰۰۲ بطور رسمی منتشر کنند. شرکت Cisco در ابتدا ایده زیرساخت شبکه‌ای را ارائه داد که برای سرویس‌های سفارشی، برنامه‌پذیر باشد. با این حال Active Network عمده‌تاً به دلیل نگرانی‌های مرتبط با امنیت و کارائی ایده خود، هرگز نتوانست توجه انبوه کارشناسان را به خود جلب کند. پروژه ۴D که در سال ۲۰۰۴ شروع شد، طرفدار طراحی کامل‌آزادی در حوزه شبکه بود. این طراحی بر جدایی بین بخش تصمیمات مسیریابی و پروتکل‌هایی تاکید داشت که بر تعاملات بین عناصر شبکه، نقش اصلی را ایفا می‌کردند. ۴D نتوانست راه خود را ادامه دهد، اما ابتکاراتی که در این پروژه مطرح شد، ایده‌های الهام بخشی را برای پروژه‌های بعدی مانند NOX، (که سیستم عاملی برای شبکه مبتنی بر OpenFlow بود) فراهم کرد. بعدها در سال ۲۰۰۶ گروه کاری IETF NETCONF<sup>۵</sup> را بعنوان پروتکلی برای تغییر پیکربندی تجهیزات شبکه، پیشنهاد داد. این گروه کاری در حال حاضر نیز فعال بوده و آخرین استاندارد آن در ژوئن ۲۰۱۱ ارائه شد. گروه کاری ForCES<sup>۶</sup> نیز در این حوزه فعالیت کرده و رویکردنی موازی با SDN را در پیش گرفته است. Open Networking Foundation<sup>۷</sup> برخی از اهداف خود را با SDN در ForCES معماري دستگاه شبکه داخلی، از نو تعریف شده، بطوری که بخش کنترل از بخش ارسال Forwarding<sup>۸</sup> (Forwarding) جدا شده است. این در حالی است که ماهیت تفکیک شده شبکه، از دنیای بیرونی پنهان بوده و کماکان به عنوان یک عنصر واحد ایفای نقش می‌کند. حال به تاریخچه پروتکل OpenFlow

<sup>۱</sup> Open Signaling Working Group<sup>۱</sup>

<sup>۲</sup> Internet Engineering Task Force<sup>۲</sup>

<sup>۳</sup> General Switch Management Protocol<sup>۳</sup>

<sup>۴</sup> موتوجه‌هایی از نوع Label Switch، موتوجه‌هایی هستند که در شبکه‌های MPLS مورد استفاده قرار می‌گیرند و از مکانیزم گذاری بسته‌ها، به جای استفاده از مسیریابی براساس آدرس IP مقصد، بهره می‌برند. (متترجم)

<sup>۵</sup> IETF Forwarding and Control Element Separation<sup>۵</sup>

می پردازیم. دقیقاً قبل از پروژه OpenFlow، دانشگاه استنفورد بر روی پروژه SANE / Ethane فعالیت می کرد. این پروژه در سال ۲۰۰۶ معماری جدیدی را برای شبکه های سازمانی ارائه کرد. تمرکز Ethane بر بکارگیری یک کنترلر متمرکز برای مدیریت سیاست ها و امنیت در یک شبکه بود.

گروهی از اپراتورهای شبکه، ارانه دهنده های سرویس و فروشنده های محصولات حوزه فناوری اطلاعات به تازگی بنیاد شبکه باز با Open Network Foundation را که سازمانی صنعت محور است به وجود آورده اند. هدف این بنیاد استاندارد کردن پروتکل OpenFlow و توسعه SDN است. در زمان ترجمه این کتاب، آخرین ویژگیهای OpenFlow در نسخه ۱.۵.۱ عرضه شد. با این حال بدليل آنکه نسخه (Wire Protocol OX01) ۱.۰.۰ توسعه یافته و پیاده سازی شده اند، ما در این کتاب خود را محدود به استفاده از نسخه ۱.۰.۰ OpenFlow می کنیم.



### اجزای سازنده

اجزای سازنده برای استقرار SDN عبارتند از ۱) سوئیچ SDN بعنوان دستگاه ارسال کننده بسته (برای نمونه، یک سوئیچ OpenFlow)، ۲) کنترلر SDN بعنوان کنترل کننده شبکه و ۳) یک اینترفیس بعنوان رابط بین این دو جزء. این رابط که از پروتکل OpenFlow بهره می برد، عموماً با عنوان اینترفیس جنوبی (Southbound Interface) نامیده می شود. ۴) همچنین این بستر دارای رابطی به نام اینترفیس شمالی (Northbound Interface) یعنی اپلیکیشن های شبکه نیز هست. سوئیچ ها در یک شبکه SDN عموماً به عنوان سخت افزار ارسال کننده اصلی معرفی شده و از طریق یک اینترفیس باز (اینترفیس جنوبی) در دسترس هستند. این در حالی است که منطق کنترل و الگوریتم ها به یک کنترلر، انتقال (Offload) یافته است.

حال به سوئیچهای SDN می پردازیم که از پروتکل OpenFlow پشتیبانی می کنند. سوئیچهای OpenFlow را می توان به دو نوع مغض "OpenFlow-only" و هایبریدی "OpenFlow-enabled" دسته بندی کرد. سوئیچ های نوع مغض، هیچ یک از خصیصه های سوئیچ های قدیمی تر (که کنترل On-board داشته اند) را ندارند و برای تصمیمات مربوط به ارسال بسته، کاملاً متکی به یک کنترلر بیرونی هستند. اما در نقطه مقابل، سوئیچ های هایبریدی علاوه بر پروتکل ها و عملکردهای مرسوم

سوئیچ‌های قدیمی، از پروتکل OpenFlow نیز پشتیبانی می‌کنند. بیشتر سوئیچ‌های تجاری موجود که امروزه مورد استفاده قرار می‌گیرند از نوع سوئیچ‌های هایبریدی هستند.

همانطور که در ابتدای فصل شرح داده شد، یک سوئیچ OpenFlow (چه از نوع محض و چه از نوع هایبریدی) دارای جدول جریانی است که وظیفه جستجوی مشخصه‌های بسته و ارسال آن را بر عهده دارد. هر جدول جریان در داخل سوئیچ، مجموعه‌ای از ورودی‌های جریان را در خود نگه می‌دارد. این ورودی‌های جریان از سه بخش تشکیل شده‌اند:

۱. فیلدی‌های هدر یا فیلدی‌های انطباق که شامل پورت ورودی و فراداده (Meta Data) بوده و برای تطابق بسته‌های ورودی به سوئیچ استفاده می‌شوند.

۲. شمارنده‌ها، برای جمع آوری آمار جریانی خاص به کار می‌روند. از جمله این آمارها می‌توان به تعداد بسته‌های دریافتی، تعداد بایتها و مدت زمانی که جریان در جدول جریان وجود دارد، اشاره کرد.

۳. مجموعه‌ای از دستورالعمل‌ها که آنها را Action می‌نامیم، چگونگی رفتار سوئیچ با یک بسته دریافتی و منطبق شده توسط سوئیچ را مشخص می‌کند. مکانیزم فوق به این صورت است که بسته‌هایی که وارد سوئیچ می‌شوند با ورودی‌های جریان موجود در جدول جریان سوئیچ منطبق می‌شوند. در صورتی که عمل انطباق با یکی از ورودی‌های جریان صورت پذیرد، دستورالعمل‌هایی که (Action) برای آن ورودی جریان تعریف شده است بر روی بسته فوق اعمال می‌شوند. این بسته‌ها یک ورودی جریان، در یک جدول جریان را مشخص می‌کنند. عنوان مثال ممکن است این Action شامل ارسال یک بسته به یک پورت خاص و یا حذف آن باشد.

سیستم تفکیک شده در SDN (و OpenFlow) را می‌توان با یک برنامه کاربردی و سیستم عامل مرتبط با آن برنامه، در یک پلتفرم محاسباتی مقایسه کرد. در SDN، کنترلر، نقش یک سیستم عامل را ایفا کرده که یک اینترفیس برای شبکه فراهم می‌کند. همچنین اپلیکیشن‌های شبکه که روی کنترلر قرار دارند، نقش نرم افزارهای سیستم عامل را ایفا می‌کنند و برای اعمال کنترل و مدیریت وظایف در شبکه، مورد استفاده قرار می‌گیرند. در شکل زیر می‌توان ساختار لایه‌بندی شده این مدل را مشاهده کرد. در این ساختار فرض بر این است که کنترل شبکه بصورت متمنکز بوده و اپلیکیشن‌ها به گونه‌ای نوشته شده‌اند که گویی شبکه یک سیستم منفرد است. اگر چه این ساختار، انجام سیاستها و وظایف مدیریتی را ساده کرده است، اما محدودیت‌های خاص خود را داشته که از آن جمله می‌توان به اجبار در اتصال فیزیکی با فاصله جغرافیایی کم بین اجزای کنترل (مانند کنترلر) و ارسال (مانند سوئیچ) اشاره کرد. همانگونه که در شکل نشان داده شده است، کنترلری که تلاش می‌کند مانند یک سیستم عامل عمل کند، باید دست کم دو اینترفیس را پیاده سازی کند: یک اینترفیس جنوبی (برای مثال، OpenFlow) که اجازه برقراری ارتباط سوئیچ‌ها با کنترلر را فراهم می‌کند و یک اینترفیس شمالی که یک API برنامه پذیر را برای کنترل شبکه و اپلیکیشن‌ها / سرویس‌ها ارائه می‌دهد. فیلدی‌های هدر با همان فیلدی‌های تطابق، در شکل

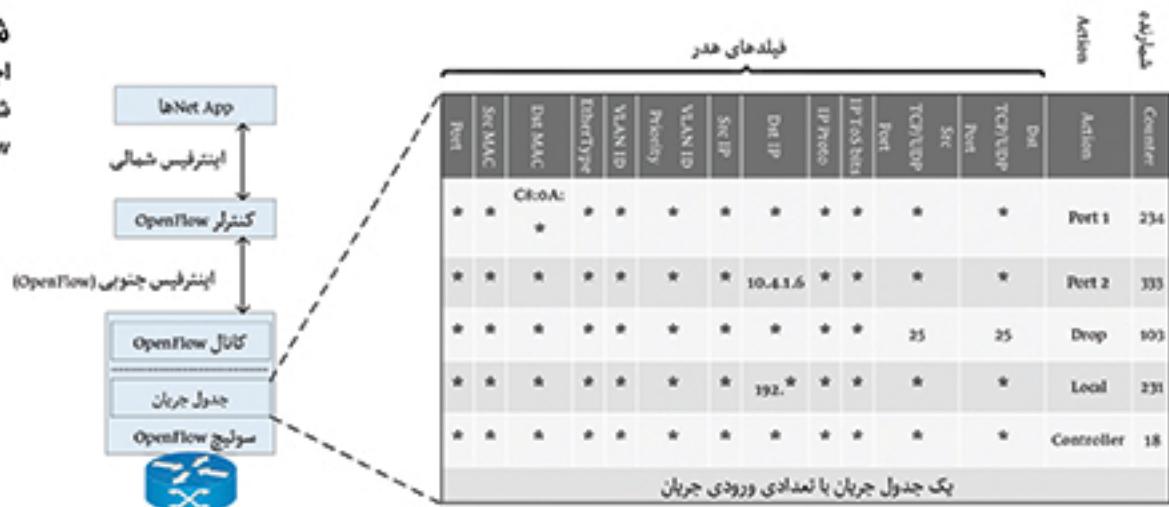
زیر نشان داده شده‌اند. هر ورودی جریان از جدول جریان، دارای یک ارزش خاص است. نکته‌ای که در این شکل وجود دارد، ارزش ANY است که در شکل، با علامت ستاره  $"^*$  نمایش داده می‌شود. این ارزش که ما آن را Wildcard هم می‌نامیم، نمایانگر آن است که فیلد مربوطه با هر مقداری می‌تواند منطبق شود.

### شکل ۱-۱

اجزای سازنده

شیکہ ہفتہ

OpenFlow



فیلد هایی که به عنوان عناصر انطباق در نظر گرفته شده اند، هر یک نمایانگر بخشی از یک بسته TCP/IP هستند. در ادامه به هر یک از این فیلد ها می پردازیم:

- فیلد Port که بصورت عددی بوده و نمایانگر پورتی (بر روی سوئیچ) است که جریان داده از طریق آن، وارد سوئیچ می شود. این پورت را پورت دریافت کننده (Incoming Port) می نامیم. محتوای این فیلد با عدد ۱ شروع می شود، همچنین طول این فیلد وابسته به پیاده سازی آن است. فیلد Port برای تمام بسته ها قابل استفاده است.
  - فیلد مربوط به آدرس های MAC مبدأ و مقصد یعنی فیلدهای Src Mac و Dst Mac، در تمامی بسته ها بر روی پورتهای فعال سوئیچ قابل اجرا بوده و طول آن ها ۴ بیت است.
  - فیلد Ethernet Type (یا همان EtherType) که نوع بسته اترنت ورودی را مشخص می کند، اندازه ۱۶ بیتی دارد و برای تمامی بسته ها روی پورت های فعال قابل اجراست. یک سوئیچ IEEE OpenFlow باید توانایی پشتیبانی از EtherType با هر دو نوع استاندارد Ethernet و IEEE 802.2 را داشته باشد و بتواند با هدر<sup>۱۷</sup> SNAP و همچنین<sup>۱۸</sup> OUI با مقدار ۰x0000000 باشد.

منطبق شود. بعنوان نمونه از مقدار 0x05FF می‌توان نام برد که برای انطباق این فیلد با تمام بسته‌های استاندارد ۸۰۲.۳، بدون هدرهای SNAP، بکار می‌رود.

- فیلد VLAN ID با تماشی بسته‌هایی که مقدار EtherType آنها 0x8100 باشد، توانایی انطباق دارد. اندازه این فیلد ۱۲ بیت است (که تعداد ۴۰۹۶ عدد VLAN را شامل می‌شود).
- فیلد VLAN ID Priority (یا همان فیلد VLAN PCP)، اندازه سه بیتی دارد و در تمام بسته‌هایی که مقدار EtherType آنها 0x8100 است، قابل اجراءست.
- فیلد های Src IP (آدرس IP مبدأ) و Dst IP (آدرس IP مقصد)، اندازه ۲۲ بیتی دارند و برای تمام بسته‌های ARP و IP قابل اجرا هستند. این فیلدها می‌توانند با یک Subnet Mask همراه شوند. اگر سوئیچ از Subnet Mask بر روی این دو فیلد پشتیبانی کند، می‌تواند انطباق‌های دقیقتری را بررسی بسته‌های ورودی، ارائه دهد.
- فیلد IP Proto در تمام بسته‌های IP Over Ethernet، IP و بسته‌های ARP قابل اجرا است و طول آن ۸ بیت است.
- اندازه فیلد IP ToS bits<sup>۱۰</sup> برابر ۸ بیت است و برای تمام بسته‌های IP قابل اجراءست. البته در ۶ بیت با ارزشتر قرار دارد.
- فیلدهای Src TCP/UDP Port و Dst TCP/UDP Port، آدرس‌های پورت لایه انتقال مبدأ و مقصد را مشخص می‌کنند و اندازه‌ای ۱۶ بیتی دارند. این فیلدها برای تمام بسته‌های TCP و ICMP و UDP قابل اجرا هستند. برای بسته‌های ICMP، تنها ۸ بیت ابتدایی (بیت‌های کم ارزشتر) برای تطابق در نظر گرفته می‌شوند.

قبل از صحبت در مورد عملیات Action، در ابتدا به فیلد شمارنده‌ها می‌پردازیم. این فیلد به ازای هر جدول، هر جریان، هر پورت و هر صفت نگهداری می‌شود. مجموعه شمارنده‌های مورد نیاز، در شکل زیر خلاصه شده است. عبارت "Byte" در این شکل و کل این کتاب به یک Octet هشت بیتی اشاره دارد. عبارت "Duration" به مدت زمانی اشاره دارد که ورودی جریان در جدول جریان سوئیچ قرار گرفته است.<sup>۱۱</sup> فیلد "Receive Errors"، تمام خطاهای واضح و آشکار از جمله فریم، Overrun، خطاهای CRC و دیگر موارد را در بر می‌گیرد.

Type of Service<sup>۱۰</sup>

<sup>۱۰</sup> نویسنده در این بخش از کلمه INSTALL استفاده کرده است که منظور همان فرآگیری جریان در جدول جریان سوئیچ و نصب آن است. (متترجم)

**شکل ۱-۲**  
فهرست شمارنده‌های  
مورد نیاز جهت  
استفاده در پیام‌های  
آماری

شمارنده‌ها به ازای پورت	شمارنده‌ها به ازای جدول
Received Packets (64 bits)	Active Entries (32 bits)
Transmitted Packets (64 bits)	Packet lookups (64 bits)
Received Bytes (64 bits)	Packet Matches (64 bits)
Transmitted Bytes (64 bits)	
Receive Drops (64 bits)	
Transmit Drops (64 bits)	
Receive Errors (64 bits)	
Transmit Errors (64 bits)	
Receive Frame Alignment Errors (64 bits)	
Receive Overrun Errors (64 bits)	
Receive CRC Errors (64 bits)	
Collisions (64 bits)	
شمارنده‌ها به ازای جریان	شمارنده‌ها به ازای صف
Received Packets (64 bits)	Transmitted Packets (64 bits)
Received Bytes (64 bits)	Transmitted Bytes (64 bits)
Duration (Seconds) (32 bits)	Transmitted Overrun Error (32 bits)
Duration (nano seconds) (32 bits)	

حال قصد داریم بحث خود را در مورد Action‌ها آغاز کرده و انواع آنها را مورد بررسی قرار دهیم. هر ورودی جریان می‌تواند بدون Action باشد. اما این قابلیت وجود دارد که حاوی یک یا چندین Action باشد. وظیفه Action‌ها دستور به سوئیچ OpenFlow برای چگونگی برخورد با بسته‌های دریافتی و منطبق شده (Match Packets) با ورودی‌های جریان است. اگر هیچ Action در جدول جریان، برای یک بسته ورودی وجود نداشته باشد، بسته دور ریخته (Drop) شده یا به سمت کنترلر ارسال می‌شود. لیست‌های Action، باید طبق توالی خاصی پردازش شوند. با این حال درون یک پورت منحصر به فرد، تضمینی بر ترتیب خروجی بسته‌ها وجود ندارد. برای نمونه دو بسته دریافتی منطبق شده، که پس از پردازش Action، به سمت یک پورت خروجی ارسال می‌شوند، ممکن است بطور اختیاری از تو مرتب شوند. سوئیچ‌های نوع محض تنها از Action‌های نوع الزامی (که در ادامه ذکر شده است) پشتیبانی می‌کنند. در حالیکه سوئیچ‌های نوع هایبریدی می‌توانند Action‌های نوع نرمال را هم پشتیبانی کنند. هر دو نوع سوئیچ می‌توانند از Action پخش کردن (Flood) پشتیبانی کنند. همانطور که پیشتر اشاره شد، Action‌ها بر دو نوع الزامی و نرمال طبقه بندی می‌شوند. Action‌های الزامی عبارتند از:

- Forward: سوئیچ‌های OpenFlow باید از ارسال کردن بسته به پورت‌های فیزیکی و مجازی که در ادامه نام برده شده، پشتیبانی کنند:

- **ALL**: بسته را به تمام اینترفیس‌های سوئیچ، به جز پورتی که بسته از آن وارد شده، ارسال می‌کند.
- **CONTROLLER**: بسته را کپسوله کرده و برای کنترلر ارسال می‌کند.
- **LOCAL**: بسته را به Stack شبکه سوئیچ ارسال می‌کند.
- **TABLE**: آن Action را که در جدول جریان سوئیچ است، اجرا می‌کند. (این فقط برای پیغام Packet-Out انجام می‌پذیرد)
- **IN\_PORT**: بسته ورودی به پورتی که از آن وارد شده، باز می‌گرداند. (به مسیر ابتدایی خود باز گردانده می‌شود)
- **Drop**: کلیه بسته‌هایی که با ورودی جریان منطبق شده‌اند، دور ریخته می‌شوند. ورودی جریانی که Action برای آن تعریف نشده باشد، بطور پیش فرض یک Action از نوع Drop محسوب می‌شود.
- در بخش قبل در مورد "Action‌های الزامی" صحبت شد. در ادامه "Action‌های اختیاری" شرح داده می‌شوند:
- **Forward**: یک سوئیچ می‌تواند بطور اختیاری از پورتهای مجازی زیر برای عمل ارسال پشتیبانی کند:
  - **NORMAL**: با استفاده از مدل های ارسال قدیمی که توسط سوئیچ (لایه ۲ سنتی، VLAN و پردازش لایه ۳) پشتیبانی می‌شود، بسته را پردازش می‌کند.
  - **FLOOD**: بسته را روی کلیه اینترفیس‌های سوئیچ، به جز اینترفیس ورودی پخش می‌کند. (Flood)
- **Enqueue**: این Action، بسته را به صفحی که به یک پورت اختصاص دارد، انتقال می‌دهد. رفتار ارسال بسته، توسط پیکربندی مربوط به صفحه تعریف می‌شود. هدف از این Action، فراهم کردن QoS ماده‌ای در سوئیچ است.
- **Modify Field**: همانطور که پیشتر بیان شد، Action‌های اختیاری مربوط به اصلاح فیلد، Action‌هایی هستند که تنها بر روی سوئیچ‌های هایبریدی قابل پیاده سازی بوده‌اند. این Action‌ها عبارتند از:
  - تنظیم VLAN ID: اگر VLAN وجود نداشته باشد، یک هدر جدید با VLAN ID مشخص (داده مرتبه ۱۲ بیتی) و اولویت صفر اضافه می‌شود. در صورتیکه هدر VLAN ID موجود باشد، VLAN ID با ارزش مشخص، جایگزین می‌گردد.

- تنظیم VLAN Priority: اگر VLAN نداشته باشیم، یک هدر جدید با اولویت مشخص (داده مرتبط ۲ بیتی) و VLAN ID صفر اضافه می‌شود. اگر از گذشته هدر VLAN ID موجود باشد، فیلد Priority با یک ارزش مشخص جایگزین می‌گردد.
  - برداشتن تگ VLAN: در صورت وجود تگ VLAN در بسته دریافتی، آن تگ را بر می‌دارد.
  - تغییر آدرس MAC مبدأ / مقصد اترنت: آدرس MAC مبدأ / مقصد اترنت را با مقدار جدید (که یک داده ۴۸ بیتی است)، جایگزین می‌کند.
  - اصلاح آدرس IP مبدأ / مقصد: آدرس IP مبدأ / مقصد موجود را با مقدار جدیدی (که یک داده ۳۲ بیتی است)، جایگزین کرده و IP Checksum را به روز می‌کند. این Action تنها بر روی بسته‌های IPv4 قابل اجراست.
  - اصلاح بیت‌های IP ToS: فیلد IP ToS موجود را با داده مرتبط با این فیلد، که اندازه‌ای معادل ۶ بیت دارد، جایگزین می‌کند. این Action تنها بر روی بسته‌های IPv4 قابل اجراست.
  - اصلاح پورت مبدأ / مقصد لایه انتقال: پورت TCP/UDP مبدأ / مقصد موجود را با داده ۱۶ بیتی جایگزین کرده و Checksum پورت مربوطه را بروز می‌کند. این کار تنها در بسته‌های TCP و UDP قابل اجراست.
- به محض ورود بسته به سوئیچ OpenFlow، فیلدهای هدرو بسته ورودی، از بسته استخراج شده و به ترتیب با فیلدهای هر یک از ورودی‌های جدول جریان مطابقت داده می‌شوند. عملیات بررسی انتظامی، از اولین ورودی جدول جریان آغاز و در ورودی‌های بعدی جدول جریان (سطح جدول جریان) ادامه می‌یابد. اگر یک ورودی جریان در جدول یافته شد که با مشخصه‌های بسته وارد شده به سوئیچ منطبق شود، سوئیچ مجموعه‌ای از Action‌های تعریف شده در آن ورودی منطبق را، بر روی جریان داده فوق اعمال می‌کند (فراموش نکنیم که جریان داده، مجموعه‌ای از بسته‌ها هستند که وارد سوئیچ شده و مقصد آنها یک آدرس خاص می‌باشد). پس از آنکه بسته‌ای با یک ورودی جریان منطبق شد، شمارنده‌های مرتبط با آن ورودی جریان به روز می‌شوند. اگر در فرایند جستجوی جدول جریان، بسته نتواند انتظامی با ورودی‌های جریان بدست آورد، Action انتخابی توسط سوئیچ، به دستورات تعریف شده در ورودی جریان جدول نواقص (Table Miss Flow Entry) وابسته خواهد بود. جدول جریان باید دارای یک ورودی نقص - جدول (Table-miss Entry) باشد تا به کمبودهای جدول رسیدگی کند. این ورودی ویژه، مشخص کننده مجموعه‌ای از Action‌های قابل انجام برای زمانی است که هیچ انتظامی برای یک بسته ورودی در جدول جریان یافته نشود. این Action‌ها شامل حذف بسته، فرستادن بسته به تمام اینترفیس‌های سوئیچ یا ارسال بسته به کنترلر از طریق کانال امن OpenFlow می‌باشد.

می‌شود. فیلد‌های هدر مورد استفاده در فرایند جستجوی جدول، به نوع بسته‌هایی که در ادامه شرح داده شده‌اند، وابسته‌اند:

- با توجه به شماره پورتی که بسته از آن وارد سوئیچ می‌شود، قوانینی را می‌توان برای آن وضع کرد. فیلد هدر Port در این بخش جهت انطباق مورد استفاده قرار می‌گیرد.
- همانطور که در شکل ۱ نشان داده شده، هدرهای لایه ۲ اترنت (شامل MAC Dst، MAC Src، EtherType و غیره)، برای بررسی انطباق کلیه بسته‌های ورودی به سوئیچ، مورد استفاده قرار می‌گیرند.
- اگر بسته حاوی VLAN باشد، (یعنی بخش EtherType آن بسته، مقدار ۰x8100 را داشته باشد) فیلد‌های VLAN ID و VLAN Priority (PCP) در جستجو، به کار می‌روند.
- فیلد‌هایی که در فرایند جستجوی بسته‌های IP استفاده می‌شوند (که در بخش EtherType آنها مقدار ۰x0800 وجود داشته باشد)، شامل فیلد‌های IP می‌شوند. این فیلد‌ها، IP، ToS، IP Src، IP Dst، Proto و غیره هستند.
- برای بسته‌های IP که از نوع UDP یا TCP هستند، (منظور بسته‌هایی که فیلد IP Proto آنها برابر با ۶ یا ۱۷ است) جستجو شامل پورت‌های لایه انتقال می‌شود. (منظور پورت‌های مبدأ / مقصد TCP/UDP)
- برای بسته‌های IP که از نوع ICMP هستند (منظور مقدار فیلد IP Proto آن برابر با ۱ است)، جستجو شامل فیلد‌های Type و Code می‌شود.
- برای بسته‌های IP که فیلد Fragment Offset آنها Nonzero است یا Flag More مربوط به Fragment آنها، بر روی ۱ تنظیم شده است، پورت‌های لایه انتقال آنها، جهت جستجو روی صفر تنظیم می‌شوند.
- بطور اختیاری برای بسته‌های ARP (که در بخش EtherType آنها مقدار ۰x0806 وجود دارد)، فیلد‌های جستجو شامل فیلد‌های مبدأ و مقصد IP می‌شود.

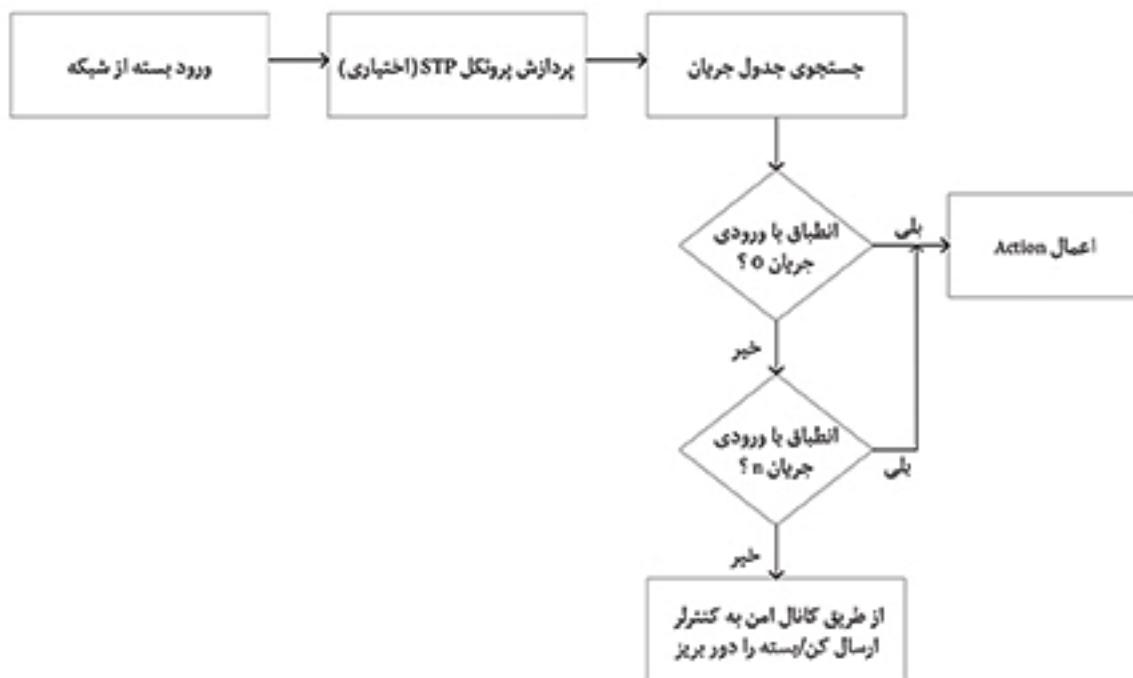
بسته‌ها بر اساس اولویتی که در کنترلر تعریف شده، با ورودی‌های جریان مطابقت داده می‌شوند. یک ورودی جریان که مطابقتی دقیق را مشخص می‌کند (یعنی ورودی‌ای که Wildcards ندارد و از مقدار ANY یا \* استفاده نکرده است)، همواره بالاترین اولویت را دارد. تمام ورودی‌هایی که Wildcard دارند، اولویتی پائین‌تر نسبت به ورودی‌های بدون Wildcard دارند. ورودی‌های با اولویت بالاتر باقیستی قبل از ورودی‌های اولویت پائین‌تر مطابقت داده شوند. اگر ورودی‌های گوناگون، اولویتی یکسان داشته باشند، سوئیچ می‌تواند در انتخاب هر ترتیبی از این ورودی‌ها آزاد باشد. عده‌های بالاتر، اولویت‌های بالاتری دارند. شکل بعد نشان دهنده فرایند انطباق بسته‌ها در یک سوئیچ OpenFlow است. لازم است

به خاطر داشته باشید اگر یک فیلد جدول جریان، ارزش ANY داشته باشد (\*، Wildcard) داشته باشد، با هر ارزشی که در هدر بسته ورودی است تطابق دارد.

انواع مختلفی از فریم‌بندی اترنت از جمله II، Ethernet II، 802.3، با SNAP یا بدون آن و... وجود دارد. اگر بسته، یک فریم از نوع Ethernet II باشد، EtherType، به روش مورد انتظار بکار می‌رود. اگر بسته، یک فریم 802.3 با یک هدر SNAP و یک OUI برابر با 0x0000000 باشد، شناسه پروتکل SNAP با EtherType آن 0x05FF ورودی منطبق می‌شود. یک ورودی جریان که مقدار EtherType آن است، با تمام فریم‌های اترنت نوع 802.3 که قادر هدر SNAP باشند و همینطور با هدراهای SNAP که قادر یک OUI با 0x0000000 باشند، منطبق هستند.



شکل ۱-۳  
فرایند انتساب بسته  
در یک سولیج  
OpenFlow



### پیغام‌های OpenFlow

ارتباط بین کنترلر و سوئیچ توسط پروتکل OpenFlow برقرار می‌شود، یعنی جاییکه مجموعه‌ای از پیام‌های تعریف شده می‌توانند از طریق یک کانال امن، بین این دو عنصر مبادله شوند. کانال امن، اینترفیسی است که هر سوئیچ OpenFlow را به یک کنترلر متصل می‌کند. سوئیچی که در حال روش

شدن بوده، تلاش می‌کند از طریق یک ارتباط<sup>۷۷</sup> TLS، به کنترلر متصل شود. پورت TCP پیش‌فرض کنترلر، ۶۶۳۳ بوده و کنترلر از طریق این پورت آماده دریافت ارتباط از سوی سوئیچ است. سوئیچ و کنترلر با تبادل گواهی‌های (Certificates) تائید شده توسط کلید خصوصی (Site-specific Private Key)، یکدیگر را تائید می‌کنند. هر سوئیچ باید بتواند توسط کاربر بگونه‌ای پیکربندی شود که با استفاده از یک گواهی (گواهی کنترلر) اعتبار کنترلر را تائید کند. از سوی دیگر یک کنترلر باید بتواند توسط کاربر بگونه‌ای پیکربندی شود که با استفاده از یک گواهی (گواهی سوئیچ) اعتبار سوئیچ را تائید کند. ترافیکی که به سوی کانال امن ارسال می‌شود یا از سوی کانال امن دریافت می‌شود، با جدول جریان بروزی نمی‌شود و بنابراین سوئیچ باید ترافیک ورودی از این اینترفیس را قبل از بررسی توسط جدول جریان، به عنوان ترافیک محلی در نظر بگیرد. در صورتی که یک سوئیچ، به علت خطاهایی همچون TLS Session Timeout، Echo Request Timeout دست دهد، باید با یک یا چندین کنترلر پشتیبان ارتباط برقرار کند. اگر کوشش‌ها حتی برای ایجاد ارتباط با کنترلر پشتیبان نیز موفقیت آمیز نباشد، سوئیچ باید وارد حالت اضطراری (Emergency Mode) شده و فوراً ارتباط TCP جاری را مجدداً برقرار کند. سپس فرایند تطبیق توسط ورودی‌های جدول جریان اضطراری (که با بیت‌های اضطراری علامت گذاری شده‌اند)، انجام می‌گیرد. مقدار Timeout برای اضطراری پیام‌های "اصلاح جریان اضطراری" باید روی صفر تنظیم شده باشد تا از جدول جریان حذف نشوند. در غیر این صورت سوئیچ باید اضافه کردن پیغام‌های اصلاح جریان (Flow Modify) خودداری کرده و یک پیغام خطأ ارسال کند. تمام ورودی‌های عادی جدول جریان، هنگامی که سوئیچ در حالت اضطراری قرار می‌گیرد، حذف می‌شوند. به محض آنکه سوئیچ توانست با کنترلر مجدداً ارتباط برقرار کند، تغییری در آنها ایجاد نمی‌شود و ورودی‌های جریان اضطراری باقی می‌مانند. کنترلر از این پس اختیار حذف تمام ورودی‌های جریان اضطراری را دارد.

اولین بار که یک سوئیچ روشن می‌شود (Bootup می‌شود)، حالت کنونی را حالت اضطراری فرض می‌کند پیکربندی حالت پیش‌فرض برای ورودی‌های جریان، خارج از حوزه پروتکل OpenFlow است.

<sup>۷۷</sup> ارتباط امنی را بین کامپیوترهای شبکه ایجاد می‌کنند. TLS پروتکلی است که هدف اصلی آن فراهم آوردن حریم خصوصی و پیکارچگی داده بین اپلیکیشن‌های ارتباطی است. این پروتکل دو ویژگی اساسی یعنی پایداری و خصوصی بودن کانال ارتباطی را ارائه می‌دهد. اپلیکیشن‌ها و وب سایتها برای داشتن ارتباطی امن از این پروتکل استفاده می‌کنند. (متترجم)

کنترلر، پیکربندی و مدیریت سوئیچ را به عهده دارد، رویدادها (Events) را از سوئیچ دریافت می‌کند و بسته‌ها را از طریق اینترفیس OpenFlow (کانال امن) به سوئیچ می‌فرستد. با استفاده از پروتکل OpenFlow، یک کنترلر راه دور می‌تواند ورودی‌های جدول جریان را در داخل سوئیچ به روز کرده، بر حسب نیاز، ورودی جریان جدیدی را به جدول جریان سوئیچ اضافه کند یا از آن حذف کند. این امر می‌تواند بصورت واکنشی (Reactive) (در پاسخ به رسیدن بسته‌ها) یا پیشگیرانه (Proactive) رخ دهد.<sup>۱۸</sup> پروتکل OpenFlow را می‌توان به عنوان یک امکان برای انجام فعل و انفعال بین کنترلر و سوئیچ (اینترفیس جنوبی) به حساب آورد، زیرا این پروتکل ارتباط بین سخت افزار سوئیچ و یک کنترلر شبکه را تعریف می‌کند. پروتکل OpenFlow نسخه ۱.۳.x، با هدف ایجاد امنیت، از قابلیت ارتباط TLS رمزگاری شده پشتیبانی می‌کند. استفاده از این قابلیت بصورت اختیاری است. همچنین این نسخه، تبادل گواهی بین سوئیچ‌ها و کنترلرها را فراهم می‌کند؛ با این حال پیاده‌سازی دقیق و ساختار گواهی‌ها در حال حاضر مشخص نیست. در ضمن گزینه‌های اختیاری و مطلوب امنیتی در خصوص سناریوهای مربوط به شبکه‌هایی با چندین کنترلر، خارج از حوزه مشخصه‌های جاری این نسخه از OpenFlow است، زیرا روش مشخصی برای اختصاص مجوزهای دسترسی محدود به یک کنترلر مجاز وجود ندارد. پروتکل OpenFlow، سه نوع پیغام را تعریف می‌کند که هر کدام زیر گروه‌های متعددی دارند:

- پیغام‌های کنترلر - به - سوئیچ (Controller-to-Switch)
- پیغام‌های متقارن (Symmetric)
- پیغام‌های غیر همزمان (Asynchronous)

### پیغام‌های کنترلر - به - سوئیچ

پیغام‌های کنترلر - به - سوئیچ توسط کنترلر آغاز شده و به سمت سوئیچ ارسال می‌شوند. این نوع پیغام برای مدیریت مستقیم یا بررسی وضعیت سوئیچ بکار رفته و در عین حال ممکن است نیازمند پاسخی از جانب سوئیچ باشد یا نباشد. پیغام‌های کنترلر - به - سوئیچ در زیر گروه‌هایی دسته بندی می‌شوند:

### • خصوصیات (Features)

به محض ایجاد یک TLS Session، کنترلر یک پیغام Feature Request به سوئیچ ارسال می‌کند. سوئیچ باید با یک پیغام Features Reply پاسخ دهد. خروجی این پیغام، خصوصیات و ظرفیت‌هایی است که توسط سوئیچ پشتیبانی می‌شود را به کنترلر اعلام می‌کند.

<sup>۱۸</sup> در فصل ۲، در مورد دو مدل واکنشی و پیشگیرانه بیشتر توضیح داده خواهد شد. (مترجم)

### • پیکربندی (Configuration)

کنترلر قادر است پارامترهای پیکربندی را در سوئیچ، مورد بررسی قرار داده و تنظیم کند. سوئیچ تنها به پرسشی که از جانب کنترلر دریافت می‌کند، پاسخ می‌دهد و خود به تنها یعنی اقدام به تغییر پیکربندی نمی‌کند.

### • وضعیت اصلاح (Modify - State)

این پیغام‌ها توسط کنترلر برای مدیریت وضعیت سوئیچ‌ها ارسال می‌شود. این پیغام‌ها برای حذف کردن، اضافه کردن یا اصلاح کردن ورودی‌های جدول جریان بکار می‌روند؛ همچنین از پیغام‌های وضعیت اصلاح برای تنظیم اولویت پورت سوئیچ استفاده می‌شود. پیغام‌های اصلاح جدول جریان می‌توانند به صورت‌های زیر باشند:

**▪ اگر Flag مربوط به پیغام `OFPFF_CHECK_OVERLAP` برای درخواست‌های ADD تنظیم شده باشد، سوئیچ ابتدا ورودی جریان جدید را با ورودی‌های جریان داخل جدول جریان خود بررسی می‌کند تا دریابد آیا بین ورودی جدید با ورودی‌های موجود همپوشانی (Overlap) وجود داشته با خبر. اگر یک تک بسته با دو ورودی منطبق باشد، هر دو ورودی جریان همپوشانی می‌شوند و اولویت یکسانی خواهند داشت. اگر یک همپوشانی متناقض بین ورودی جریان موجود و درخواست ADD داشته باشیم، سوئیچ نباید درخواست ADD را قبول کرده و با نوع خطای `OFPFMFC_OVERLAP` و کد خطای `OFPET_FLOW_MODE_FAILED` پیغام `ofp_error_msg` را ارسال کند. برای تقاضای ADD معتبر (ناهمپوشان)، یا تقاضاهایی که Flag مربوط به بررسی همپوشانی آنها، تنظیم نشده است، سوئیچ باید ورودی جریان را با پایین ترین مقدار جدول جریان درج کند، تا بتواند تمام Wildcard‌ها را در فرایند انطباق جریان، یعنی ساختار `flow_match`، پشتیبانی کند. با این عملکرد، اولویت‌بندی جریانها در فرایند انطباق قابل مشاهده است. اگر یک ورودی جریان با فیلدها و اولویت‌های هدایت یکسان در جدول جریان وجود داشته باشد، ورودی جریان به همراه شمارنده خود باید حذف و ورودی جریان جدید باید اضافه شود. اگر یک سوئیچ نتواند برای یک جریان داده دریافتی، ورودی جریانی را به جدول جریان اضافه کند، سوئیچ باید با نوشتن نوع خطای `PFOFMFC_ALL_TABLES_FULL` و کد خطای `OFPET_FLOW_MOD_FAILED` پیغام `ofp_error_msg` را ارسال کند. اگر در یک پیغام اصلاح جریان، فهرستی از Action‌هایی وجود داشته باشد که به پورتی که هرگز در سوئیچ نبوده اشاره کنند،**

سوئیچ باید با نوشتن نوع خطای `OFPET_BAD_ACTION` و کد خطای `OFPBAC_BAD_OUT`، پیغام `ofp_error_msg` را بفرستد. اگر پورت مورد ارجاع در آینده اضافه شود (برای مثال وقتی یک مازول حاوی چندین پورت فیزیکی به شاسی سوئیچ اضافه می‌شود)، سوئیچ ممکن است یا بدون اطلاع، بسته‌های ارسال شده به پورت مبدأ را دور بریزد یا فوراً یک پیغام خطای `OFPBAC_BAD_PORT` ارسال کند و پیغام اصلاح جریان را قبول نکند.

**MODIFY**: اگر ورودی جریان با یک فیلد هدر یکسان در جدول جریان وجود نداشته باشد، فرمان `MODIFY` مانند فرمان `ADD` عمل می‌کند و ورودی جریان جدید باید با شمارنده‌های صفر شده درج شود. در غیر این صورت فیلد Action، بر روی ورودی موجود تغییر می‌کند اما فیلد شمارنده‌ها و همچنین مقدار Idle Timeout بدون تغییر باقی می‌مانند.

**DELETE**: برای تقاضاهای حذف، اگر هیچ جریان ورودی منطبقی یافت نشود، هیچ خطایی ثبت نمی‌شود و اصلاح جدول جریان انجام نمی‌پذیرد. اگر ورودی جریان با درخواست حذف منطبق شود، ورودی باید حذف و سپس هر ورودی نرمال با `OFPPFF_SEND_FLOW_REM`، باید یک پیغام حذف جریان (Flow Removal) تولید کند. ورودی‌های جریان اضطراری حذف شده، پیغام حذف جریان تولید نمی‌کنند. فرمانهای `DELETE` و `DELETE_STRICT` (به بخش `DELETE_STRICT` و `MODIFY` در ادامه توجه کنید) را می‌توان بطور اختیاری توسط پورت خروجی فیلتر کرد. اگر فیلد `out_port` حاوی مقداری غیر از `OFPP_NONE` باشد، محدودیتی را هنگام تطابق نشان می‌دهد. این محدودیت بیان می‌کند که این قانون (ورودی) باید حاوی یک Action خروجی هدایت شده به آن پورت باشد. این فیلد توسط پیغام‌های `MODIFY_STRICT` و `MODIFY_ADD` نادیده گرفته می‌شود.

**DELETE و MODIFY**: فرمانهای اصلاح جریان (FlowMod)، مشابه نسخه‌هایی با پسوند `_STRICT` هستند. در نسخه‌های بدون پسوند `RESTRICT`، فیلدهای Wildcard فعال بوده و تمام جریان‌هایی که با توضیح درون این پیغام منطبق هستند، اصلاح یا حذف می‌شوند. در نسخه‌های `_STRICT` تمام فیلدهای دارای Wildcard و اولویت دار با ورودی جریان کاملاً منطبق می‌شوند و تنها یک جریان یکسان و منطبق، اصلاح یا حذف می‌شود. برای نمونه، اگر پیغامی برای حذف ورودی‌ها به سوئیچی که تمامی Flag‌های آن برابر Wildcard است، ارسال شود، فرمان `DELETE` همه جریان‌ها را از تمام جدول‌ها حذف می‌کند. با این حال فرمان

نها ورودی جریانی را حذف می‌کند که برای تمام بسته‌ها و `DELETE_STRICT` اولویت‌های ویژه بکار می‌رود. برای فرمان `DELETE` و فرمان بدون پسوند `MODIFY` که حاوی Wildcard هستند، انطباق هنگامی اتفاق می‌افتد که یک ورودی جریان کاملاً منطبق باشد یا از محتوای درون فرمان `flow_mod` خاص‌تر باشد. برای مثال، اگر یک فرمان `DELETE` اعلام کند تمام جریان‌ها با پورت مقصد ۸۰ حذف شوند، یک ورودی جریان که تمام Wildcard است، حذف نخواهد شد. با این حال یک فرمان `DELETE` که تمامی فیلدهای آن Wildcard است، یک ورودی منطبق با کلیه ترافیک‌های روی پورت ۸۰ را، حذف خواهد کرد.

#### • حالت خواندن (Read - State)

این پیغامها تمام آمارها را از جدول‌های جریان سوئیچ، پورت‌ها و تک تک ورودی‌های جریان جمع آوری می‌کند.

#### • ارسال بسته (Send - Packet)

این پیغامها توسط کنترلر ایجاد شده و به سوئیچ ارسال می‌شود. کنترلر با استفاده از این پیغامها به سوئیچ اعلام می‌کند که بسته‌ها بر روی کدام پورت خاص به سمت مقصد ارسال شوند.

#### • مانع (Barrier)

پیغامهای Barrier Request و Barrier Reply توسط کنترلر ایجاد می‌شوند. این پیغامها برای اطمینان از توجه به وابستگی‌های پیغام یا اطلاع رسانی برای کارهای تکمیل شده، بکار می‌رود. به عبارت ساده‌تر هنگامی که کنترلر این پیغام را ایجاد می‌کند، به سوئیچ دستور می‌دهد که در ابتدا کلیه درخواستهای قبل از پیغام Barrier را به پایان برساند و سپس به سراغ درخواستهای ارسال شده جدید (در خواسته‌ای ارسال شده بعد از پیغام Barrier) برود.

#### پیغامهای متقارن (Symmetric)

پیغامهای متقارن که توسط سوئیچ یا کنترلر ایجاد و ارسال می‌شوند، بدون درخواست از هر یک از دو سمت فرستاده می‌شوند. پیغامهای متقارن در پروتکل OpenFlow به سه دسته فرعی تقسیم می‌شوند:

#### Hello •

به محض آنکه ارتباط بین سوئیچ و کنترلر تنظیم و برقرار شود، پیغامهای Hello رد و بدل می‌شوند.

**Echo •**

پیغامهای Echo Reply و Echo Request می‌توانند هم از سوئیچ و هم از کنترلر ارسال شوند. پس از آنکه پیغام Echo Request ایجاد و ارسال شد، باید با یک پیغام Echo Reply به آن پاسخ داده شود. این پیغامها می‌توانند برای نشان دادن تأخیر، پهنای باند، و / یا برقرار بودن ارتباط بین کنترلر و سوئیچ (که یک Heartbeat است) بکار روند.

**Vendor •**

این پیغامها روشنی استاندارد برای سوئیچ‌های OpenFlow فراهم آورده، تا بتوانند قابلیت‌های افزونه‌ای را در داخل پیغامهای OpenFlow، برای نسخه‌های آتی این پروتکل ارائه دهند.

**پیغامهای ناهمزمان (Asynchronous)**

پیغامهای ناهمزمان توسط سوئیچ ایجاد و ارسال می‌شوند. این پیغامها برای بروز نگاه داشتن کنترلر در خصوص رویدادهای شبکه و همچنین ایجاد تغییر در وضعیت سوئیچ بکار می‌روند. سوئیچ‌ها، پیغامهای ناهمزمان را به کنترلر ارسال می‌کنند تا رسیدن یک بسته، تغییر وضعیت سوئیچ یا یک خطا را به اطلاع کنترلر برسانند. چهار نوع پیغام ناهمزمان اصلی به شرح زیر وجود دارد:

**Packet-in •**

پیغام Packet-in در دو حالت به کنترلر ارسال می‌شود. حالت اول، برای تمام بسته‌های دریافتی که ورودی جریان منطبقی برای آنها در جدول جریان وجود نداشته باشد. حالت دوم، بسته‌های دریافتی که با ورودی جریان دارای یک Action معادل "send-to-controller"، منطبق می‌شوند. اگر سوئیچ، حافظه کافی برای بافر کردن بسته‌هایی که به کنترلر ارسال می‌شوند، داشته باشد، پیغام Packet-in تنها بخشی از هدر بسته دریافتی را در خود قرار می‌دهد ( بصورت پیش فرض، ۱۲۸ بایت). اما اگر سوئیچ از بافر داخلی پشتیبانی نکند (یا فضای بافر داخلی آن پر شده باشد)، بسته بطور کامل در پیغام قرار داده شده و به کنترلر ارسال می‌گردد. در صورتیکه سوئیچ از بافر استفاده کند، برای Packet-in بسته‌های ارسالی به کنترلر، یک Buffer ID در نظر گرفته می‌شود.

**• حذف جریان (Flow-Removal)**

زمانیکه یک ورودی جریان توسط یک پیغام اصلاح جریان به سوئیچ اضافه می‌شود (به بخش اصلاح وضعیت رجوع شود)، مقدار Idle Timeout همانند مقدار Hard Timeout، تعیین کننده زمانی است

که ورودی بدلیل نبود فعالیتی باید از بین برود. تنها یک تفاوت بین Hard Timeout و Idle Timeout وجود دارد که براساس آن، مقدار Hard Timeout است که تعیین می‌کند ورودی بدون توجه به فعالیت یا عدم فعالیتش، چه زمانی باید از بین برود. زمانی که جریان خاتمه می‌باید پیغام اصلاح جریان، مشخص می‌کند که آیا سوئیچ باید پیغام حذف جریان به کنترلر بفرستد یا خیر. پیغامهای اصلاح جریان که ورودی‌های جریان را حذف می‌کنند، نمی‌توانند باعث ایجاد پیغامهای حذف جریان شوند.

#### • وضعیت پورت (Port-status)

پیغامهای وضعیت پورت توسط سوئیچ زمانی به کنترلر ارسال می‌شوند که حالت پیکربندی (Configuration-state) پورت تغییر می‌کند. این رویدادها عبارتند از تغییرات در حالت پورت (به طور مثال، غیرفعال شدن پورت توسط کاربر)، یا یک تغییر در حالت پورت، مانند آنچه توسط استاندارد 802.1D مشخص شده است (منظور پروتکل Spanning Tree است). سوئیچ‌های OpenFlow می‌توانند بصورت اختیاری از پروتکل STP پشتیبانی کنند. انتظار می‌رود این سوئیچ‌ها تمام بسته‌های STP را قبل از انجام بررسی جریان، بصورت محلی پردازش کنند. بنابراین وضعیت پورتها همانگونه که با STP مشخص شده است، بدین منظور بکار می‌رود که بسته‌های ارسال شده به پورت `OFP_FLOOD` را تنها به آن پورتهایی محدود کند که در محدوده Spanning Tree باشند. تغییرات پورت به عنوان نتیجه Spanning Tree از طریق پیغامهای بروز رسانی پورت (Port-Update) به کنترلر فرستاده می‌شوند. به یاد داشته باشید، Action‌های مربوط به ارسال، که پورت خروجی مربوط به `OFP_ALL` را مشخص می‌کنند، حالت پورت را که توسط STP معین می‌شود، نادیده می‌گیرند. بسته‌هایی که بر روی پورتهای غیر فعال شده توسط پروتکل STP دریافت می‌شوند، باید مسیر عادی پردازش جدول جریان را دنبال کنند.

#### • خطأ (Error)

سوئیچ قادر است بوسیله پیغامهای خطأ، کنترلر را نسبت به مشکلات آگاه کند.

هسته اصلی خصوصیات OpenFlow، مجموعه ساختارهای زبان C است که برای پیغامهای پروتکل OpenFlow بکار می‌رود. خوانندگان علاقمند می‌توانند، این ساختار داده‌ها و توضیحات جزئی تر آنها را در آدرسهای اینترنتی زیر بیابند:

[www.openflow.org/documents/openflow-spec-v1.0.0.pdf](http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf)

[www.opennetworking.org/ja/sdn-resources-jp/onf-specifications](http://www.opennetworking.org/ja/sdn-resources-jp/onf-specifications)



## اینترفیس شمالی

سیستم‌های مدیریت بیرونی (External Management Systems) یا اپلیکیشن‌های شبکه (Net Apps) ممکن است تمايل به بدست آوردن اطلاعات درباره شبکه زیربنایی و یا کنترل بخشی از رفتار یا سیاست شبکه را داشته باشند. علاوه بر این، کنترل‌ها ممکن است ارتباط با یکدیگر را به دلایل گوناگونی ضروری قلداد کنند. برای مثال، یک اپلیکیشن کنترلی ممکن است نیاز به مدیریت و رزرو منابع تجهیزاتی همچون سوئیچ داشته باشد. یعنی منابعی که در اختیار چندین کنترل قرار گرفته است. عنوان مثالی دیگر می‌توان از کنترل‌ی نام برد، که نقش یک کنترل اصلی را ایفا کرده و با یک کنترل پشتیبان اطلاعات مربوط به سیاستهای خود را به اشتراک می‌گذارد. از این دست مثال‌ها که نیاز به اتصال چندین کنترل به یکدیگر در آنها احساس می‌شود، بسیار زیاد است. برخلاف ارتباط کنترل با سوئیچ که از یک استاندارد تبعیت می‌کند (که اینترفیس جنوبی خوانده می‌شود)، در حال حاضر استاندارد پذیرفته شده‌ای برای اینترفیس شمالی (یعنی اینترفیس بین کنترل و Net App) وجود ندارد و تمايل برنامه‌نویسان و اهل فن در این حوزه، به ساخت و پیاده‌سازی اپلیکیشن‌های خاص، بصورت موردي است. یک دلیل بالقوه این است که اینترفیس شمالی تماماً در نرم افزار تعریف شده است، در حالیکه تعاملات بین کنترل و سوئیچ به پیاده‌سازی سمت سخت افزار نیازمند است. اگر کنترل را به عنوان یک سیستم عامل شبکه در نظر بگیریم، یک اینترفیس جنوبی با تعریف دقیق باید وجود داشته باشد که از طریق آن، اپلیکیشن‌ها بتوانند به سخت افزار (سوئیچ‌ها) دسترسی داشته باشند، با دیگر اپلیکیشن‌ها به تعامل پردازد و خدمات آن سیستم را بدون نیاز به این که توسعه دهنده اپلیکیشن جزئیات انجام کار کنترل را بداند، مورد استفاده قرار دهد. در حالی که طیف گسترده‌ای از کنترل‌ها وجود دارند، اپلیکیشن‌های شبکه در این لحظه، در مراحل اولیه بوده، مستقل از هم عمل می‌کنند و با یکدیگر تطابق ندارند. تا زمانیکه یک استاندارد در اینترفیس شمالی وجود نداشته باشد، اپلیکیشن‌های SDN به توسعه خود بصورت موردي ادامه می‌دهند و رسیدن به مقاومتی همچون قابل حمل بودن (Portable) و انعطاف پذیری (Flexible) برای اپلیکیشن‌های شبکه، ممکن است زمانبر باشد.

### خلاصه

پروتکل OpenFlow ادماهه برخی از تلاشهای صورت گرفته در گذشته است که برای فراهم آوردن بستری تفکیک شده بین بخش Control Plane و Data Plane، در تجهیزات شبکه صورت پذیرفته است. پیشینه چنین کوششهایی در این فصل ارائه شد. معرفی اجزای اصلی و کلیدی در استقرار SDN، به طور خاص در پروتکل OpenFlow، به همراه پیغامهای مورد استفاده در شبکه مبتنی بر OpenFlow، در این فصل بررسی شد. پس از معرفی پروتکل OpenFlow، در فصل بعد به معرفی و پیاده‌سازی مرجع سوئیچ OpenFlow در نرم افزارها و سخت افزارها می‌پردازیم. همچنین مقدمه‌ای در خصوص محیط آزمایشگاهی Mininet خواهیم داشت.