بررسی شبکه با تعریف نرم افزار:

گذشته، حال، و آینده شبکه های قابل برنامه ریزی

چکیده

به‌تازگی ایده‌ی شبکه‌های قابل برنامه‌ریزی دوباره به شتاب قابل توجهی با توجه به ظهور شبکه‌ی تعریف شده با نرم‌افزار (SDN) رسیده است. SDN، اغلب به‌عنوان یک "ایده‌ی اصلی و جدید در شبکه"، به‌طور چشمگیری موجب تسهیل مدیریت شبکه و نوآوری فعال از طریق برنامه‌ریزی شبکه شده است. بررسی مقاله در شبکه‌های قابل برنامه‌ریزی با تاکید در SDN بوده است. یک چشم‌انداز تاریخی از برنامه‌ریزی شبکه از ایده‌های اولیه به تحولات اخیر ارائه شده است. سپس به‌طور خاص معماری SDN و استاندارد OpenFlow، در مورد جایگزین‌های فعلی برای پیاده‌سازی و تست پروتکل‌ها و سرویس‌های مبتنی بر SDN، بررسی برنامه‌های کاربردی آینده‌ی SDN ، اکتشاف و تحقیقات براساس SDN بحث و بررسی شده است.

کلیدواژه‌ها: شبکه با تعریف نرم افزار، برنامه‌ریزی شبکه، بررسی، نقشه‌ی داده‌ها، کنترل هواپیما، مجازی‌سازی

1. مقدمه

شبکه‌های کامپیوتری به‌طور معمول از تعداد بیشتری از دستگاه‌های شبکه مانند روترها، سوئیچ‌ها و انواع متعددی از middleboxesبا بسیاری از پروتکل‌های پیچیده که بر روی آنها اجرا می‌شوند ساخته شده‌اند (به‌عنوان مثال، دستگاه‌هایی که ترافیک را برای مقاصد دیگر حمل و نقل بسته، مانند فایروال دستکاری می‌کنند). اپراتورهای شبکه

برای پیکربندی سیاست‌ها به طیف گسترده‌ای از حوادث شبکه و برنامه‌های کاربردی پاسخ‌گو هستند. آنها نیاز به تبدیل وظایف پیچیده و سطح بالا با دسترسی به ابزار محدود دارند.. به‌عنوان یک نتیجه، مدیریت شبکه و تنظیم عملکرد کاملا چالش برانگیز و مستعد خطا است. واقعیت این است که دستگاه‌های شبکه معمولا جعبه سیاه یکپارچه و به‌صورت عمودی برای چالش اپراتورهای شبکه و مدیران هستند.

یکی دیگر از چالش‌برانگیزترین موضاعاتی که محققان با آن روبرو هستند " تشکیل اینترنت" است. از آنجا که پایه استقرار آن بخشی از زیرساخت‌های حیاتی جامعه ما است (درست مثل حمل‌ونقل و شبکه‌های انرژی)، اینترنت با دشواری هم با تکامل فیزیکی و هم زیرساخت‌ها و پروتکل‌ها و عملکرد آن منطبق می‌شود. با این حال، به‌عنوان برنامه‌های کاربردی اینترنت و خدمات در حال ظهور و پیچیده و سخت، ضروری است که اینترنت قادر به تکامل این چالش‌های جدید باشد.

ایده‌ی "شبکه‌های قابل برنامه‌ریزی" به‌عنوان راهی برای تسهیل تکامل شبکه پیشنهاد شده است. به‌طورخاص، شبکه‌های با تعریف نرم‌افزار (SDN) یک الگوی شبکه جدید است که در آن سخت‌افزار حمل‌ونقل از کنترل تصمیم‌گیری جدا است. این کار به‌طور قابل توجهی برای مدیریت شبکه و نوآوری و تکامل فعال ساده است. ایده‌ی اصلی این روش به توسعه‌دهندگان نرم‌افزار اجازه می‌دهد تا به منابع شبکه به همان شیوه‌ای که در ذخیره‌سازی و منابع محاسباتی انجام می‌دهند تکیه کنند. در SDN، شبکه‌های اطلاعاتی به‌طور منطقی بر کنترل مبتنی بر نرم افزار متمرکز هستند (نقشه‌ی کنترل)، و دستگاه‌های شبکه، دستگاه‌های ساده‌ی حمل‌ونقل بسته (نقشه‌ی داده) از طریق یک برنامه‌ریزی رابط باز هستند (به‌عنوان مثال، ForCES [1]، OpenFlow [2] و غیره).

SDN در حال حاضر توجه دانشگاه و صنعت را به خود جلب کرده است. یک گروه از اپراتورهای شبکه، ارائه‌دهندگان خدمات و فروشندگان به‌تازگی بنیاد گسترش شبکه ایجاد کرده‌اند [3]، یک سازمان صنعتی به ترویج SDN و استانداردسازی پروتکل OpenFlow [2] مشغول است. از نظر دانشگاهی، مرکز تحقیقات شبکه OpenFlow [4] با تمرکز بر تحقیق SDN ایجاد شده است. تلاش‌های استاندارد زیادی در SDN در IETF و IRTF و دیگر سازمان‌های تولید استاندارد وجود دارد.

زمینه‌ی شبکه با تعریف نرم‌افزار کاملا جدید است، در عین حال با سرعت بسیار زیادی درحال‌رشد است. بااین‌حال، چالش‌های پژوهش مهمی وجود دارد. در این مقاله، شبکه‌های قابل برنامه‌ریزی با ارائه‌ی یک دیدگاه تاریخی در این زمینه و همچنین جزئیات توصیف پارادایم SDN و معماری را بررسی خواهیم کرد. ساختار مقاله به شرح زیر است: بخش دوم، با توصیف تلاش‌های اولیه آغاز می‌شود و بر شبکه‌های قابل برنامه‌ریزی توجه دارد. بخش سوم نمای کلی از SDN و معماری آن را نشان می‌دهد. همچنین به توصیف پروتکل OpenFlow می‌پردازد. بخش چهارم سیستم عامل‌های موجود برای توسعه و تست راه حل SDN از جمله شبیه‌سازی و ابزار شبیه‌سازی، پیاده‌سازی کنترل SDN و همچنین تأیید و اشکال‌زدائی ابزار را توصیف می‌کند. در بخش پنجم، در مورد برنامه‌های مختلف SDN در زمینه‌هایی مانند مراکز داده و شبکه‌های بی‌سیم بحث می‌کنیم. در نهایت، بخش ششم چالش تحقیقات و جهت‌گیری‌های آینده را مورد بحث قرار می‌دهد.

2. شبکه قابل برنامه‌ریزی اولیه

SDN دارای پتانسیل بسیار زیادی برای تغییر شبکه است و OpenFlow به‌عنوان یک "تبلیغ ایده‌های جدید در شبکه " است[5]. مزایای ارائه شده در محدوده‌ی کنترل متمرکز، الگوریتم ساده، شکل سخت‌افزار شبکه، از بین بردن middleboxe، طراحی و استقرار شخص ثالث برنامه‌ها است.

در حالی که OpenFlow توجه قابل ملاحظه از صنعت دریافت کرده است، شایان ذکر است که ایده‌ی شبکه‌های قابل برنامه‌ریزی و کنترل منطق جدا شده است. در این بخش، یک نمای کلی از تلاش‌های اولیه برای شبکه‌های قابل برنامه‌ریزی، پیش‌سازهای پارادایم SDN که پایه و اساس بسیاری از ایده‌های گذشته است و ما امروزه شاهد آن هستیم، ارائه می‌کنیم.

A) گسترش signaling: (OPENSIG) با گروهی که در سال 1995 با یک سری از کارگاه‌ها که به "ساخت ATM، اینترنت و شبکه‌های تلفن همراه باز، توسعه و برنامه‌ریزی " مشغول بودند آغاز شد[6]. آنها باور داشتند که جدایی بین

ارتباطات سخت‌افزار و کنترل نرم‌افزار لازم است اما برای تحقق بخشیدن چالش برانگیز است. این عمدتا به‌دلیل عمودی و یکپارچه بودن سوئیچ‌ها و روترها است، که به ماهیت بسته‌ی ساخته شده از استقرار سریع خدمات جدید شبکه و محیط‌های غیرممکن بستگی دارد. هسته اصلی پیشنهاد آنها فراهم کردن دسترسی به سخت افزار شبکه از طریق باز کردن رابط‌های شبکه قابل برنامه‌ریزی است؛ که اجازه می‌دهد خدمات جدید از طریق یک محیط برنامه‌ریزی توزیع شده استقرار یابند.

انگیزه این ایده‌ها، منجر به تشکیل گروه کاری IETF شد، که مشخصات پروتکل مدیریت سوئیچ عمومی (GSMP)[7]، یک پروتکل هدف کلی برای کنترل یک سوئیچ برچسب را مشخص می‌کردند. GSMP اجازه می‌دهد تا یک کنترل‌کننده برای ایجاد و انتشار اتصالات در سراسر سوئیچ، اضافه کردن و حذف برگ بر روی یک اتصال چندپخشی، مدیریت سوئیچ پورت، درخواست اطلاعات پیکربندی، درخواست و حذف رزرو منابع سوئیچ و درخواست آمار ایجاد شود. کار گروه به‌طور رسمی به این نتیجه رسیدند و آخرین استانداردهای پیشنهادی، GSMPv3، در ماه ژوئن سال 2002 منتشر شد.

B) شبکه‌های فعال: در اواسط 1990، شبکه‌های فعال [8]، [9] ابتکار عمل این ایده از زیرساخت‌های شبکه را پیشنهاد دادند که می‌توانست برای خدمات سفارشی برنامه‌ریزی شود. دو روش اصلی در این زمینه وجود دارد، عبارتند از: (1) سوئیچ قابل برنامه‌ریزی، با انتقال داده‌ها و خارج از باند کانال‌های مدیریت؛ (2) کپسول‌ها، که هر قطعه از برنامه هستند که می‌توانند در پیام‌های کاربر جای گیرند. هر قطعه از برنامه توسط روتر اجرا می‌شود. با وجود انگیزه‌ی فعالیت قابل توجهی، شبکه‌های فعال هرگز به جمع‌آوری و استفاده گسترده و توسعه صنعت، به دلایل امنیتی و نگرانی عملکرد روی نیاوردند [10].

C) DCAN: ابتکار عمل دیگری که در اواسط 1990 روی داد کنترل شبکه‌های ATM بود (DCAN) [11]. هدف این پروژه طراحی و توسعه زیرساخت‌های لازم برای کنترل مقیاس‌پذیر و مدیریت شبکه ATM بود. در این پروژه فرض

بر این بود که، کنترل و مدیریت توابع در بسیاری از دستگاه‌ها (سوئیچ‌های ATM در مورد DCAN) باید جدای از خود دستگاه‌ها باشد و به موجودیت‌های خارجی اختصاص داده شده برای این منظور که اساسا با مفهوم DCAN همراه هستند واگذار شود. DCAN فرض می‌کند که یک پروتکل بین مدیر و شبکه وجود دارد، همان خطوطی که امروزه در طرح‌هایی مانند OpenFlow اتفاق می‌افتد. موارد بیشتر را می‌توان در پروژه‌ی DCAN در [12] یافت.

هنوز در خطوط SDNs و جدایی پیشنهادی کنترل و نقشه‌ی داده بر روی شبکه‌های ATM، در میان دیگران، در کار ارائه شده در [13] معماری کنترل چندگانه ناهمگن، مجاز به اجرای همزمان شبکه‌های ATM فیزیکی با پارتیشن‌بندی منابعی که بین آن کنترل‌ها جابه‌جا می‌شوند، است.

D) پروژه‌ی 4بعدی: با شروع در سال 2004، پروژه‌ی 4بعدی [14]، [15]، [16] از یک طراحی حمایت کرد که بر جدایی بین مسیریابی تصمیم‌گیری منطقی و پروتکل حاکم بر تعامل بین عناصر شبکه تاکید داشت. که "تصمیم‌گیری" در مورد یک دیدگاه جهانی از شبکه، سرویس‌های "انتشار" و "کشف" نقشه‌ها، برای کنترل «داده» جهت حمل ترافیک را پیشنهاد داد. این ایده‌ها یک القای مستقیم برای کاهای بعدی مانند NOX ارائه کردند [17]، که "سیستم عامل برای شبکه‌ها" یک پیشنهاد در چارچوب شبکه OpenFlow بود.

E) NETCONF: در سال 2006، گروه پیکربندی شبکه IETF، NETCONF را به‌عنوان یک پروتکل مدیریت برای تغییر تنظیمات دستگاه‌های شبکه پیشنهاد داد. پروتکل به دستگاه‌های شبکه امکان افشای API از طریق توسعه پیکربندی داده‌ها را برای ارسال و بازیابی می‌داد.

پروتکل مدیریت دیگری، که به‌طور گسترده از گذشته تا به امروز استفاده می‌شد، SNMP نام داشت [19]. SNMP در اواخر دهه 80 مطرح شد و به‌عنوان یک شبکه‌ی مدیریت پروتکل بسیار محبوب مطرح شد، که از مدیریت ساختار رابط (SMI) برای واکشی داده‌های موجود در مدیریت پایگاه اطلاعات (MIB) استفاده می‌کرد. این مسئله می‌تواند برای تغییر متغیرها در MIB به‌منظور تغییر تنظیمات پیکربندی استفاده شود. بعد از آن علی‌رغم آنچه که بود، SNMP

برای پیکربندی تجهیزات شبکه استفاده نمی‌شد، بلکه به عنوان یک عملکرد و ابزاری برای نظارت بر خطا به‌کار می‌رفت. علاوه بر این، کاستی‌های متعددی در مفهوم SNMP شناسایی شد، قابل توجه‌ترین آن‌ها عدم امنیت قوی بود. این در نسخه‌های بعدی پروتکل نشان داده شده است.

NETCONF، در آن زمان توسط IETF پیشنهاد شده بود و توسط بسیاری به‌عنوان یک رویکرد جدید برای مدیریت شبکه دیده که کاستی مذکور را در SNMP حل می‌کرد به‌کار می‌رفت. اگر چه پروتکل NETCONF هدف ساده‌ی دستگاه، پیکربندی و عمل به‌عنوان یک بلوک ساختمان برای مدیریت را انجام می‌داد، هیچ جدایی بین داده‌ها و نقشه‌های کنترل وجود نداشت. همان را می‌توان در مورد SNMP نیز اعلام کرد. یک شبکه با NETCONF باید به‌طورکامل به‌عنوان قابلیت‌های جدید در هر دو دستگاه شبکه و مدیریت قابل برنامه‌ریزی در نظر گرفته شود، به‌طوری‌که هر قابلیت جدیدی را بتواند ارائه دهد. علاوه بر این، باید طوری طراحی شود که در درجه اول به پیکربندی خودکار نه برای کنترل غیرمستقیم سرویس کمک کند. با این وجود، هر دوNETCONF و SNMP ابزارهای مدیریتی مفیدی هستند که ممکن است به‌صورت موازی در سوئیچ ترکیبی از دیگر راه‌حل‌های شبکه که قادر به برنامه‌ریزی هستند حمایت کند.

گروه NETCONF درحال‌حاضر فعال است و استاندارد پیشنهادی اخیر در ژوئن 2011 منتشر شد.

F) **Ethane**: کار قبل از OpenFlow بود پروژه Ethan/SANE بود [20]، که در سال 2006، یک معماری جدید برای شبکه‌های سازمانی تعریف کرد. تمرکز Ethan بر استفاده از کنترل متمرکز برای مدیریت سیاست و امنیت در شبکه می باشد. یک مثال قابل توجه ارائه‌ی کنترل دسترسی مبتنی بر تشخیص است. مشابه SDN، Ethan دو قطعه را به‌کار برد: یک کنترل‌کننده برای تصمیم‌گیری اگر یک بسته باید فرستاده شود، و یک کلید Ethan متشکل از یک جدول جریان و یک کانال امن برای کنترل.

Ethan پایه و اساس تبدیل شدن به شبکه با تعریف نرم‌افزار است. برای قرار دادن Ethan در پارادایم SDN امروز، کنترل دسترسی مبتنی بر هویت Ethan به احتمال زیاد به عنوان یک برنامه در بالای کنترل SDN مانند NOX [17]، Maestro [21]،Beacon [22]، SNAC [23]، Helios [24]، و غیره اجرا خواهد شد.

3 . معماری شبکه‌های با تعریف نرم‌افزار

شبکه‌های ارتباطی داده‌ها معمولا شامل کاربر نهایی دستگاه، یا میزبان متصل شده توسط زیرساخت‌های شبکه است. این زیرساخت‌ها توسط میزبان و به‌کارگیری عناصر سوئیچینگ مانند روتر و سوئیچ و همچنین لینک‌های ارتباطی برای حمل داده‌ها بین میزبان به اشتراک گذاشته می‌شود. روترها و سوئیچ‌ها معمولا سیستم‌های "بسته" اغلب با رابط محدود کنترل فروشنده هستند. بنابراین، یک بار مستقر می‌شوند و در تولید زیرساخت های شبکه متداول کاملا دشوار هستند؛ به‌عبارت‌دیگر، استقرار نسخه‌های جدید پروتکل‌های موجود (به‌عنوان مثال، 6IPv)، به‌طورکامل به استقرار پروتکل‌ها و خدمات جدید نیست و تقریبا مانع غیر قابل عبور در شبکه‌های فعلی است. اینترنت، به‌عنوان شبکه‌ای از شبکه‌ها، از این قاعده مستثنی است.

همانطور که قبلا ذکر شد، به اصطلاح اینترنت "استخوان"[2] است و تا حد زیادی به اتصال میان نسبت داده اطلاعاتی و کنترل نقشه بستگی دارد به این معنی که تصمیم‌گیری در مورد اطلاعات جریان از طریق شبکه در هیئت مدیره هر عنصر شبکه ساخته شده است. در این نوع محیط، استقرار برنامه‌های کاربردی شبکه جدید با اهمیت است، همان‌طور که آن‌ها نیاز به اجرای مستقیم زیرساخت‌ها دارند. حتی کارهای ساده مانند پیکربندی و یا اجرای سیاست ممکن است نیاز به مقداری تلاش به‌دلیل عدم وجود یک رابط کنترل مشترک به دستگاه‌های شبکه‌های مختلف داشته باشند. روش دیگر، استفاده از "middleboxe" (به‌عنوان مثال، فایروال، تشخیص نفوذ سیستم‌ها، مبدل آدرس شبکه و غیره) بالای زیرساخت‌های شبکه‌های زیربنایی ارائه شده است و به‌عنوان راهی برای دور زدن استخوان‌سازی اثر شبکه است. شبکه‌های تحویل محتوا (CDN ها) [25] مثال خوبی هستند.

شبکه با تعریف نرم‌افزار برای تسهیل نوآوری و فعال کردن کنترل ساده و برنامه‌ریزی شده از مسیر داده‌های شبکه، توسعه داده شده است. همانطور که در شکل 1 می‌بینید، جدایی سخت‌افزار حمل‌ونقل از منطق کنترل اجازه می‌دهد تا استقرار پروتکل‌های جدید و برنامه‌های کاربردی، تجسم شبکه و مدیریت، و تحکیمmiddleboxes های مختلف برای کنترل نرم‌افزار راحت‌تر شود. به‌جای اجرای سیاست‌ها و پروتکل‌های در حال اجرا در دستگاه‌های پراکنده، شبکه به حمل"ساده" سخت‌افزار و کنترل شبکه تصمیم‌گیری (بازدیدکنندگان) کاهش می‌یابد.

A) معماری کنونی SDN

در این بخش، ما به بررسی دو معماری شناخته شده SDN، یعنی ForCES [1] و OpenFlow [2] خواهیم پرداخت. هر دوی OpenFlow و ForCES به‌دنبال اصل اساسی SDN از جدایی بین کنترل و داده‌ها هستند؛ و هر دو تبادل اطلاعات بین نقشه‌ها را استاندارد می‌کنند. بااین‌حال، آنها از لحاظ فنی در طراحی، معماری، مدل و رابط پروتکل حمل بسیار متفاوت هستند.

1) ForCES: روش پیشنهادی توسط IETF برای ForCES (حمل و جداسازی عنصر کنترل) باهمدیگر کار می‌کنند، باز تعریف معماری داخلی دستگاه شبکه ملزم به داشتن عنصر کنترل جدا از عنصر حمل است. بااین‌حال، دستگاه شبکه هنوز به‌عنوان یک نهاد نشان داده می‌شود. مثال رانندگی مورد استفاده توسط گروه تمایل به ترکیب سخت‌افزار حمل جدید با در نظر کنترل شخص ثالث در درون یک دستگاه شبکه است. بنابراین کنترل و داده‌ها در نزدیکی یکدیگر نگهداری می‌شوند (به‌عنوان مثال، همان جعبه یا اتاق). در مقابل، کنترل به‌طور کامل از دستگاه شبکه در سیستم SDN "مانند OpenFlow" جدا است.

ForCES دو عنصر منطقی به نام حمل عنصر (FE) و عنصر کنترل (CE) را تعریف می‌کند، که هر دو پروتکل ForCES را برای ایجاد ارتباط اجرا می‌کنند. FE مسئول استفاده از سخت‌افزار زیرین برای ارائه هر بسته است. CE کنترل و توابع

سیگنالینگ و به‌کارگیری پروتکل FE در چگونگی رسیدگی به مسئولیت بسته را اجرا می‌کند. پروتکل بر اساس مدل masterslave کار می‌کند، که در آن FE ها slave و CE ها master هستند.

بلوک مهم معماری ForCES، LFB (بلوک توابع منطقی) هستند. LFB یک بلوک عملکردی به خوبی تعریف شده در FE ها است که توسط CE ها از طریق پروتکل ForCES کنترل می‌شود. LFB، CES را قادر به کنترل پیکربندی و چگونگی پردازش بسته‌ها توسط FES می‌کند.

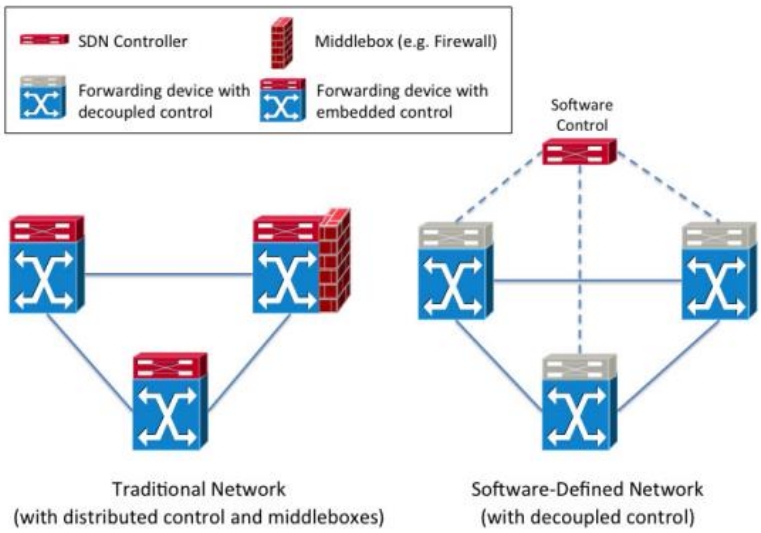
ForCES از سال 2003 تحت استاندارد است، و گروه کاری انواع اسناد منتشر کرده است از جمله: بیانیه کاربرد، چارچوب معماری تعریف اشخاص و اثر متقابل آنها، مدل‌سازی زبان تعریف توابع منطقی در حمل عنصر و پروتکل برای ارتباط بین کنترل عناصر و حمل درون یک عنصر شبکه. گروه کاری در حال حاضر فعال است.

2) OpenFlow: توسط اصل SDN از جدایی کنترل و اطلاعات حمل OpenFlow [2]، مانند ForCES ، تبادل اطلاعات بین دو نقشه را استاندارد می‌کند.

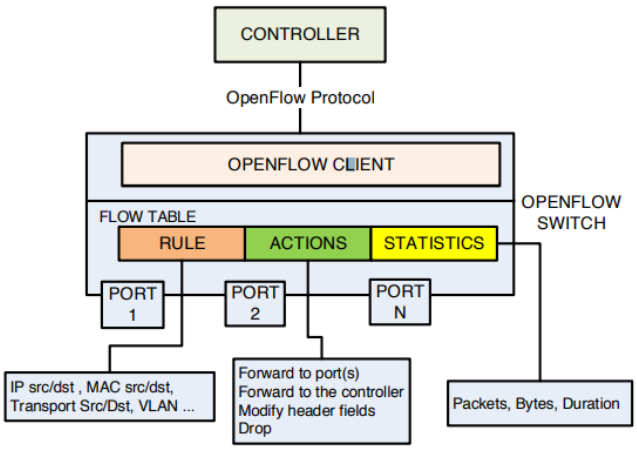
در معماری OpenFlow، که در شکل 2نشان داده شده است، حمل دستگاه و یا سوئیچ OpenFlow، شامل یک یا چند جدول جریان و یک لایه انتزاعی است که با یک کنترل از طریق پروتکل OpenFlow به ارتباطات امن می‌پردازد. جداول جریان شامل عناصر جریان است، که هر کدام تعیین‌کننده‌ی چگونگی تعلق بسته به یک جریان پردازش و فرستاده است. جریان شامل: (1) زمینه بازی و یا قوانین تطبیق برای انطباق بسته‌های ورودی به‌کار می‌رود: زمینه بازی ممکن است حاوی اطلاعات موجود در هدر بسته، پورت ورودی، و ابرداده باشد؛ (2) شمارنده، برای جمع‌آوری آمار برای جریان خاص، از جمله تعداد بسته‌های دریافت شده، تعداد بایت و مدت زمان جریان مورد استفاده قرار می‌گیرد. (3) مجموعه‌ای از دستورالعمل‌ها و یا اقدامات، بر یک مسابقه اعمال شود. آنها دیکته می‌کند که چگونه تطبیق بسته را به‌دست گیرند.

ورود بسته در یک سوئیچ OpenFlow، هدر بسته استخراج شده و در برابر زمینه تطبیق بخشی از مدخل جدول جریان انطباق می‌یابد. اگر یک ورودی منطبق یافت شود، سوئیچ مجموعه‌ای مناسب از دستورالعمل‌ها و یا اقدامات را

که با ورود جریان همسان در ارتباط است اعمال می‌کند. اگر جریان جدول بر یک مسابقه عمل جستجو را انجام دهد، عمل انجام شده توسط سوئیچ به دستورالعمل تعریف شده با ورود جدول جریان بستگی دارد. هر جدول باید حاوی یک جریان و ویژگی‌های جدید باشد.



شکل 1: معماری SDN منطق کنترل را از تجهیزات حمل جدا می‌کند و قادر به تثبیت middleboxes، مدیریت ساده سیاست و توابع جدید است.



شکل 2: ارتباط بین کنترل‌کننده و دستگاه حمل از طریق پروتکل OpenFlow اتفاق می‌افتد. جداول جریان توسط قوانین تطبیق، اقدامات انجام شده زمانی که جریان بر قوانین منطبق است و شمارنده برای جمع‌آوری آمار جریان تشکیل شده است.

این ورود خاص، با مجموعه‌ای از اقدامات مشخص می‌شود زمانی که هیچ بازیی برای یک بسته ورودی، از قبیل انداختن بسته، ادامه روند تطبیق در جدول جریان بعدی، یا هدایت رو به جلو بسته به کنترل‌کننده‌ی OpenFlow وجود نداشته باشد. شایان ذکر است که نسخه OpenFlow 1.1 از جداول چندگانه و پردازش خط لوله پشتیبانی می‌کند. امکان دیگر، در مورد سوئیچ ترکیبی، به‌عنوان مثال، سوئیچ‌هایی که هر دو OpenFlowو غیر OpenFlow را دارند، هدایت رو به جلوی بسته‌های غیر منطبق با استفاده از طرح‌های حمل به‌طور منظم IP است.

ارتباط بین کنترل‌کننده و سوئیچ از طریق پروتکل OpenFlow اتفاق می‌افتد، که یک مجموعه از پیام‌هایی که می توانند بین این اشخاص در یک کانال امن رد و بدل شوند تعریف می‌کند. با استفاده از پروتکل OpenFlow یک کنترل از راه دور می‌تواند، به‌عنوان مثال، اضافه کردن، به‌روزرسانی و یا حذف نوشته‌های جریان از سوئیچ جداول جریان را انجام دهد. که می‌تواند به‌صورت واکنشی (در پاسخ به ورود یک بسته) و یا فعالانه اتفاق بیافتد.

3) بحث: در [26]، شباهت‌ها و تفاوت میان ForCES و OpenFlow بحث شده است. بین تفاوت‌ها، این واقعیت که مدل حمل استفاده شده توسط ForCES متکی بر بلوک‌های تابع منطقی (LFB) است برجسته‌تر است، در حالی که OpenFlow از جداول جریان استفاده می‌کند. آن‌ها نشان دادند که در اقدامات OpenFlow در ارتباط با یک جریان، می‌توان برای ارائه کنترل بیشتر و انعطاف‌پذیری برای اهداف مدیریت شبکه، مدیریت و توسعه ترکیب شد. در ForCES، ترکیبی از LFB های مختلف می‌تواند برای رسیدن به هدف مورد استفاده قرار گیرد.

ما همچنین باید-تکرار کنیم که ForCES نباید از مدل SDN تحت زیربنای OpenFlow پیروی کند، اما می‌تواند برای رسیدن به همان هدف‌ها و پیاده‌سازی قابلیت‌های مشابه مورد استفاده قرار گیرد [26].

حمایت قوی از صنعت، تحقیقات و دانشگاه که شبکه‌های بنیاد باز (ONF) و پیشنهادSDN آن، OpenFlow، قادر به جمع‌آوری کاملا چشمگیر بوده است. نتیجه‌ی بحرانی ناشی از این بخش‌های مختلف توسط تعداد قابل توجهی از مقالات پژوهشی، پیاده‌سازی نرم‌افزار مرجع و حتی سخت‌افزار تولید شده است. به‌طوری که برخی استدلال می‌کنند که معماری OpenFlow درحال‌حاضر یک استاندارد بالفعل SDN است. در این روند، باقی‌مانده‌ی این بخش بر مدل

OpenFlow SDN تمرکز می‌کند. به طور خاص، اجزای مختلف معماریSDN توصیف خواهد شد،که عبارتند از: سوئیچ، کنترل، رابط‌های موجود در کنترل برای ارتباط با دستگاه‌های حمل (ارتباط عازم جنوب) و شبکه برنامه‌های کاربردی (ارتباطات عازم شمال). بخش چهارم نیز به OpenFlow تمرکز دارد و آن را به‌عنوان سیستم عامل موجود برای توسعه SDN و تست، از جمله شبیه‌سازی و ابزار شبیه‌سازی، کنترل پیاده‌سازی SDN و همچنین تأیید و اشکال‌زدائی ابزار، توصیف می‌کند. بحث ما در مورد برنامه‌های کاربردی آینده SDN و جهت تحقیقات کلی‌تر و معماری SDN است.

B) دستگاه حمل

زیرساخت‌های شبکه ممکن است شامل تعدادی تجهیزات مختلف فیزیکی شبکه و یا دستگاه‌های حمل مانند روترها، سوئیچ، سوئیچ‌های مجازی، نقاط دسترسی بی‌سیم باشند. در یک شبکه با تعریف نرم‌افزار، چنین دستگاه‌هایی اغلب به‌عنوان سخت‌افزار حمل عمومی هستند که از طریق یک رابط باز در یک لایه انتزاع، به‌عنوان منطق کنترل و الگوریتم تخلیه به یک کنترل‌کننده عمل می‌کنند. چنین دستگاه حملی معمولا به SDN اشاره دارد، همانطور که در شکل 3 نشان داده شده است.

در شبکه OpenFlow، سوئیچ‌ها در دو گونه هستند: خالص و ترکیبی. سوئیچ OpenFlow خالص هیچ ویژگی وراثت و یا کنترل ندارد، و به‌طورکامل بر کنترل برای تصمیم‌گیری حمل تکیه دارد. پشتیبانی سوئیچ ترکیبی OpenFlow علاوه بر عملیات‌های سنتی و پروتکل‌ها است. اکثر سوئیچ‌های تجاری امروزی و در دسترس هیبریدی هستند.

1) پردازش قوانین حمل: معماری SDN مبتنی بر جریان مانند OpenFlow ممکن است از جدول حمل اضافی، فضای بافر و شمارنده آماری که اجرای آن در سوئیچ ASIC سنتی دشوار است بهره ببرد. برخی از طرح‌های اخیر [27]، [28] با اضافه کردن یک CPU با هدف خاص، یا سوییچ که ممکن است برای تکمیل توابع خاص و کاهش پیچیدگی

طراحی ASIC مورد استفاده قرار گیرد حمایت کرده‌اند. این کار موجب سود بیشتری برای انعطاف‌پذیری بیشتر پردازش سوئیچ به عنوان برخی از جنبه‌های تعریف نرم‌افزار شده است.

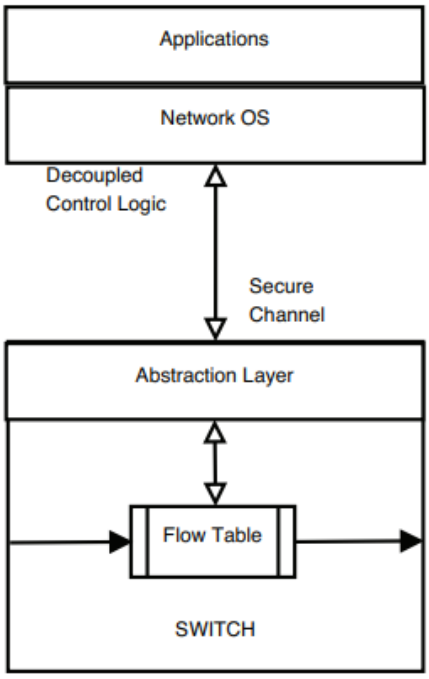
در [29]، کارتهای شتاب مبتنی بر پردازنده شبکه برای انجام سوئیچینگ OpenFlow مورد استفاده قرار می‌گیرد. آنها جزئیات طراحی و نتایج گزارش شده را پیشنهاد و توصیف کردند که کاهش20٪ در تاخیر بسته دارد. در [30]، یک طراحی معماری به‌منظور بهبود عملکرد جستجو در سوئیچ OpenFlow در لینوکس ارائه شده است. نتایج اولیه نشان داد که گزارش سوئیچینگ بسته تا 25٪ در مقایسه با سوئیچینگOpenFlow مبتنی بر نرم‌افزار افزایش یافته است. مطالعه دیگری در عملکرد داده سوئیچینگ OpenFlow مبتنی بر لینوکس در [31] ارائه شده است، که سوئیچینگ OpenFlow، سوییچینگ اترنت لایه 2 و عملکرد مسیریابی لایه 3را مورد مقایسه قرار می‌دهد. انصاف، توان حمل و تاخیر بسته در شرایط بار متنوع مورد تجزیه و تحلیل قرار گرفت. در [32]، یک مدل اولیه برای سرعت حمل و مسدود کردن احتمالی یک سوئیچ OpenFlow، مشتق شده بود در حالی که پارامترهای مدل اندازه‌گیری از معیار زمان سوئیچ از سخت‌افزار OpenFlow فعلی، همراه با یک کنترل‌کننده‌ی OpenFlow به‌دست آمده بود.

2) نصب قوانین حمل: موضوع دیگر در مورد مقیاس‌پذیری شبکه OpenFlow محدودیت حافظه در دستگاه‌های حمل است. قوانین OpenFlow پیچیده‌تر از قوانین حمل در روترهای IP سنتی است. آنها از تطابق انعطاف‌پذیر و زمینه‌های تطبیق و نیز اقداماتی متفاوت به محض ورود بسته حمایت می‌کنند. یک سوئیچ به‌طور معمول از چند هزار تا ده هزار قوانین حمل پشتیبانی می‌کند [33]. همچنین، حافظه قابل آدرس‌دهی سه‌تایی محتوا (TCAM) برای حمایت از قوانین حمل استفاده می‌شود، که می‌توانند گران و قدرتمند باشند. بنابراین، فضای قوانین یک تنگنای مقیاس‌پذیری از OpenFlow است و استفاده بهینه از فضا قوانین برای جریان با توجه به سیاست شبکه‌ها و محدودیت‌ها، یک موضوع چالش‌برانگیز و مهم است.

برخی از پیشنهادات رسیدگی به محدودیت‌های حافظه در سوئیچ‌های OpenFlow را نشان می‌دهند. Devoflow [34] یک فرمت گسترش‌یافته به OpenFlow برای شبکه‌های با کارآیی بالا است. که دسته‌ای جریان‌ها را (به‌عنوان مثال جریان کوتاه) در سوئیچ OpenFlow حمایت می‌کند و تنها کنترلر را به منظور رسیدگی به جریان (یعنی

جریان بزرگتر) درگیر می‌کند. ارزیابی عملکرد انجام شده در [34] نشان داد که Devoflow از فضای جدول جریان 53-10 بار کمتر استفاده می‌کند. در DIFANE [35]، "نفوذ" سوئیچ‌ها موجب تغییر مسیر بسته‌ها به "قدرت" سوئیچ می‌شود که تمام قوانین حمل را در حالی که جدول جریان قوانین نفوذ سوئیچ برای استفاده در آینده لازم است ذخیره می‌کند. کنترلر مسئول پارتیشن‌بندی قوانین در سوئیچ قدرت است.

Palette [36] و یک سوئیچ بزرگ [37] مسئله قوانین مکان‌یابی را نشان داده‌اند. هدف آنها به حداقل رساندن تعداد این قوانین است که نیاز به نصب در دستگاه‌های حمل و استفاده از سیاست‌های مسیریابی پایان به پایان به عنوان ورودی به یک قانون بهینه‌ساز دارند. سیاست پایان به پایان مجموعه‌ای شامل قوانین اولویت‌بندی، برای مثال، کنترل دسترسی و تعادل بار است، در حالی که مشاهده‌ی کل شبکه به عنوان یک تک سوئیچ مجازی است. از سوی دیگر سیاست‌های مسیریابی، از طریق آنچه که ترافیک مسیر نامیده می‌شود باید در شبکه جریان داشته باشد. ایده‌ی اصلی در Palette سیاست پارتیشن‌بندی پایان به پایان در جداول و سپس توزیع آنها در بیش از یک سوئیچ است. خود الگوریتم شامل دو مرحله است: تعیین تعداد K جدول مورد نیاز و سپس تنظیم قوانین پارتیشن‌بندی در بیش از K جدول. از سوی دیگر، یک سوئیچ بزرگ، مسئله‌ی قوانین مکان‌یابی را به‌طور جداگانه برای هر مسیر حل می‌کند، انتخاب مسیر بر اساس معیارهای شبکه (مثلا زمان تاخیر، ازدحام و پهنای باند) و سپس ترکیب نتایج برای رسیدن به یک راه‌حل جهانی است.



شکل 3: منطق کنترل از هم جدا می‌تواند به‌عنوان یک عامل مشاهده‌ی سیستم شبکه، بر اساس برنامه‌های کاربردی ساخته شده برای "برنامه" شبکه باشد.

C) کنترلر

سیستم جدا شده با یک سیستم عامل مقایسه می‌شود [17]، که در آن کنترلر یک برنامه رابط به شبکه فراهم می‌کند. که می‌تواند برای پیاده‌سازی وظایف مدیریت و ارائه ویژگی‌های جدید استفاده شود. لایه‌بندی این مدل در شکل 3 نمایش داده شده است. این انتزاع فرض می‌کند که کنترل متمرکز است و برنامه‌های کاربردی اگر شبکه یک سیستم واحد باشد، نوشته شده است. این مسئله مدل SDN را قادر می‌سازد در یک طیف گسترده‌ای از برنامه‌های کاربردی و فن‌آوری‌های شبکه ناهمگن و رسانه‌های فیزیکی به‌عنوان بی‌سیم (به‌عنوان مثال 802.11 و 802.16)، سیمی (به‌عنوان مثال اترنت) و شبکه‌های نوری اعمال شود.

به‌عنوان یک مثال عملی از انتزاع لایه‌بندی در دسترس از طریق رابط‌های برنامه‌نویسی نرم‌افزار باز (API ها)، شکل 4 معماری یک کنترل‌کننده SDN بر اساس پروتکل OpenFlow را نشان می‌دهد. این کنترل‌کننده خاص یک چنگال

از کنترلرهای علامت [22] نامیده می‌شود [38]. در این شکل ممکن است ملزم به رعایت جدایی بین کنترل‌کننده و لایه‌های نرم‌افزاری باشیم. نرم‌افزار می‌تواند در جاوا نوشته و ساخته شود و می‌تواند در تعامل با ماژول کنترل از طریق یک API جاوا باشد. این مثال خاص از یک کنترلر SDN است که اجازه می‌دهد تا از اجرای ساخته شده در ماژول‌ها بتوانند با اجرای آنها در کنترلر OpenFlow ارتباط برقرار کنند (به‌عنوان مثال خدمات OpenFlow). از سوی دیگر، کنترل‌کننده می‌تواند با دستگاه‌های حمل با پروتکل OpenFlow از طریق حضور لایه انتزاعی در سخت‌افزار حمل ، که در شکل 3 نشان داده شده است ارتباط برقرار کند.

در حالی که لایه‌بندی انتزاعی در دسترس فوق از طریق رابط‌های برنامه کاربردی اجازه می‌دهد تا سیاست اجرا و وظایف مدیریت ساده شود، اتصالات بین کنترل و نقل و نگهداری شبکه عناصر باید به‌هم پیوسته باشد. انتخاب‌ها در طول اجرای چنین معماری لایه‌بندی به‌طور چشمگیری تحت تاثیر عملکرد و مقیاس‌پذیری شبکه قرار می‌گیرد. در زیر، برخی از نگرانی‌های مقیاس‌پذیری و برخی از طرح‌های پیشنهادی با هدف غلبه بر این چالش‌ها مطرح شده است. بحث مفصل در مورد لایه کاربرد و اجرای خدمات و اجرای سیاست به بخش ششم موکول می‌شود

1) کنترل مقیاس‌پذیری: نگرانی اولیه‌ای که مطرح است، تخلیه کنترل تجهیزات و مقیاس‌پذیری سوئیچینگ و عملکرد کنترل شبکه (ها) است. کنترلرEthane اصلی [20]، در یک کامپیوتر رومیزی میزبانی می‌شود، که مسئولیت رسیدگی به درخواست 11،000 جریان جدید در هر مورد و پاسخ در 1.5 میلی‌ثانیه را دارد. مطالعه‌ی اخیر [39] در چندین پیاده‌سازی کنترلر OpenFlow(NOX-MT، Maestro، Beacon)، بر روی شبیه‌سازی یک شبکه بزرگتر با 100،000 نقطه انتهایی و 256 سوئیچ ، انجام شده است که همه قادر به تحمل حداقل 50،000 درخواست جریان جدید در هر ثانیه در هر یک از حالات تست شده است. در ماشینeightcore ، اجرای NOX-MT چند رشته‌ای، 1.6 میلیون درخواست جریان جدید در هر ثانیه با میانگین زمان پاسخ 2 میلی‌ثانیه را نشان داده است. همانطور که نتایج نشان می‌دهد، کنترل‌کننده قادر به اداره یک تعداد شگفت‌آور از درخواست جریان جدید است و باید قادر به مدیریت تمام بزرگترین شبکه‌ها نیز باشد. علاوه‌براین، کنترل‌کننده‌های جدید در حال توسعه مانند McNettle [40] سرور چندهسته‌ای قدرتمند را هدف قرار می‌دهند و در حال طراحی مرکز داده بزرگ (حدود 20 میلیون جریان

درخواست در هر ثانیه و تا 5000 سوئیچ) هستند. بااین‌حال، کنترل‌های متعددی ممکن است جهت کاهش زمان تاخیر و یا افزایش تحمل خطا مورد استفاده قرار گیرد.

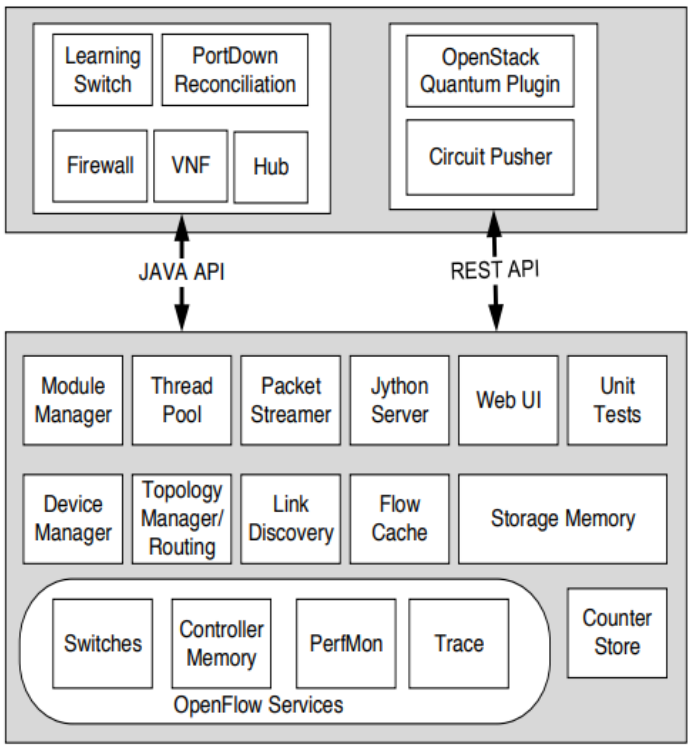
نگرانی اصلی، مشکل قرار دادن کنترلر [41] است، که در تلاش برای تعیین تعداد بهینه کنترلر و موقعیت آنها در توپولوژی شبکه، اغلب بین بهینه‌سازی برای متوسط و بدترین انتخاب زمان تاخیر است. تاخیر لینک استفاده شده برای ارتباطات بین کنترل و سوئیچ از اهمیت زیادی در تعریف یک شبکه و یا ارزیابی عملکرد آن دارد [34]. یکی از انگیزه‌های اصلی در [42] بود که به بررسی چگونگی تاثیر کنترل‌کننده و انجام شبکه با پهنای باند و مسائل تاخیر بر روی لینک کنترل می‌پردازد. این کار نتیجه می‌دهد که پهنای باند در کنترل لینک، چگونه می‌تواند بسیاری از جریان‌ها را با کنترلر به‌عنوان زمان از دست دادن زمانی که در شرایط اشباع است پردازش کند. تاخیر سوئیچ به کنترل، تأثیر عمده‌ای بر رفتار کلی شبکه دارد. این فاصله می‌تواند به‌طور چشمگیری با تاخیر لینک و تاثیر عملکرد برنامه‌های کاربردی شبکه رشد یابد.

همچنین، مدل‌سازی کنترل تا حد زیادی مقیاس‌پذیری شبکه را تحت تاثیر قرار می‌دهد. برخی از مسائل مهم مقیاس‌پذیری در [43] همراه با بحث در مورد مقیاس‌پذیری تجارت در طراحی شبکه با تعریف نرم‌افزار ارائه شده است.

2) مدل‌های کنترل: در زیر، برخی از گزینه‌های طراحی SDN و روش‌های مختلف کنترل شبکه با تعریف نرم‌افزار بحث شده است، که بسیاری از آنها عبارتند از:

- **تمرکز در مقابل توزیع**

اگر چه پروتکل‌ها مانند OpenFlow مشخص می‌کنند که یک سوئیچ توسط یک کنترل‌کننده کنترل می‌شود و در نتیجه تمرکز را بیان می‌کنند، شبکه با تعریف نرم‌افزار ممکن است یک کنترل متمرکز یا توزیع شده داشته باشد.



شکل 4: معماری نور افکن به عنوان مثال از یک کنترلر OpenFlow

ارتباطات کنترلر به کنترلر توسط OpenFlow تعریف نشده است، بنابراین برای هر توزیع یا افزونگی در کنترل جریان موثر است.

کنترل متمرکز فیزیکی نشان‌دهنده‌ی یک نقطه‌ی شکست برای کل شبکه است. بنابراین، جریان باز اجازه می‌دهد تا کنترل‌های متعدد به یک سوئیچ متصل شوند، تا کنترلر پشتیبان‌گیری از شکست محفوظ بماند.

Onix [44] و HyperFlow [45] ایده‌ی تلاش بیشتر برای حفظ تمرکز منطقی اما توزیع‌شده‌ی فیزیکی کنترل را مطرح کردند. این باعث کاهش سربار توسط ارتباط با کنترلر محلی شد، در حالی که هنوز به برنامه‌های کاربردی اجازه می‌دهد تا با یک دیدگاه ساده شده مرکزی از شبکه نوشته شوند. پتانسیل حرکت نزولی [46] مربوط به سازگاری و staleness است که در سراسر کنترل توزیع شده است و پتانسیل این را دارد که منجر به یک برنامه با یک نمای دقیق از شبکه به عمل اشتباه می‌شود.

یک روش ترکیبی، مانند Kandoo [47]، می‌تواند کنترلر محلی برای برنامه‌های محلی را بهینه کند و مسیر کنترل‌کننده‌ی جهانی را برای تصمیماتی که نیاز به شبکه متمرکز دارند تغییر دهد. این عمل باعث کاهش بار بر روی

فیلتر کنترل‌کننده‌های جهانی با تعداد درخواست جریان جدید می‌شود، درحالی‌که موجب ارائه اطلاعات مسیر با پاسخ سریع‌تر برای درخواست می‌شود که می‌تواند توسط یک برنامه کنترل محلی در نظرگرفته شود.

شبکه با تعریف نرم‌افزار همچنین می‌توانید برخی از سطوح تمرکززدایی منطقی، با کنترل منطقی متعدد را داشته باشد. یک نوع جالب از کنترل پروکسی،Flowvisor[48] نام دارد، که می‌تواند برای اضافه کردن یک سطح مجازی‌سازی شبکه به شبکه‌ی OpenFlow استفاده شود و به کنترل‌های متعدد اجازه می‌دهد تا به‌طور همزمان مجموعه‌های در تداخل سوئیچ فیزیکی را کنترل کنند. این عمل در ابتدا برای تحقیقات تجربی در شبکه‌های مستقر و در کنار تولید ترافیک توسعه یافته بود و موجب سهولت گسترش خدمات جدید در محیط SDN می‌شد.

کنترل منطقی غیر متمرکز در بین دامنه متعددی از شبکه‌های پوشا مورد نیاز خواهد بود. اگرچه ممکن است دامنه موافق کنترل متمرکز نباشد، اما یک سطح معینی از به اشتراک‌گذاری ممکن است مناسب باشد (به‌عنوان مثال، برای اطمینان از توافقنامه سطح خدمات برای ملاقات ترافیک جریان بین دامنه).

- **کنترل دانه دانه**

به‌طور سنتی، واحد اساسی شبکه بسته است. هر بسته حاوی اطلاعات آدرس لازم برای یک سوئیچ شبکه برای تصمیم‌گیری در مورد مسیریابی است. با این حال، اکثر برنامه‌های کاربردی داده را به‌عنوان یک جریان از بسته‌های فردی ارسال می‌کنند. شبکه‌ای که آرزوی ارائه کیفیت سرویس QoS و یا تضمین خدمات به برنامه‌های خاصی که از کنترل مبتنی بر جریان‌های فردی بهره‌مند هستند را دارد. کنترل منجر به جمع‌آوری جریان انتزاع و نه جریان فردی است. تجمع جریان ممکن است بر اساس منبع، مقصد، نرم‌افزار و یا هر ترکیبی در آن باشد.

در یک شبکه با تعریف نرم‌افزار که در آن عناصر شبکه کنترل از راه دور هستند، سربار ترافیک ناشی از داده‌ها و کنترل است. به این ترتیب، استفاده از سطح بسته دانه دانه منجر به تاخیر اضافی به‌عنوان کنترلر می‌شود که مجبور به اتخاذ یک تصمیم برای هر ورود بسته است. هنگام کنترل جریان فردی، تصمیم گرفته شده برای اولین بسته می‌تواند برای

جریان تمام بسته‌های پس از آن نیز به‌کار رود. سربار ممکن است با گروه بندی جریان مانند ترافیک بین دو میزبان و انجام کنترل تصمیم‌گیری در جریان جمع آوری شده، کاهش یابد.

- **واکنش در مقابل سیاست‌های بلادرنگ**

طبق مدل کنترل واکنش، مانند مدل پیشنهادی توسط Ethane [20]، عناصر حمل باید مشورت کنترل‌کننده در هر زمان تصمیم‌گیری را جویا شوند، مانند زمانی که یک بسته از یک جریان جدید به یک سوئیچ می‌رسد. در مورد کنترل دانه دانه مبتنی بر جریان، یک تاخیر عملکرد کوچکی به‌عنوان اولین بسته از هر جریان جدید به کنترل‌کننده برای تصمیم‌گیری وجود خواهد داشت (به عنوان مثال، رو به جلو و یا رها کردن)، و پس از آن بسته‌های بعدی در جریان با سرعت زیاد در سخت‌افزار ارسال خواهند شد. در حالی که تاخیر متحمل شده توسط اولین بسته ممکن است در بسیاری از موارد قابل اغماض باشد، ممکن است نگرانی دیگری به‌وجود آید اگر کنترل از راه دور به لحاظ جغرافیایی دور باشد (هر چند این می‌تواند توسط توزیع فیزیکی کنترل کاهش یابد [45]) و یا اگر جریان مانند جریان تک بسته‌ای کوتاه مدت باشد. همچنین برخی از مسائل مقیاس‌پذیری در شبکه‌های بزرگتر وجود دارد، به‌عنوان مثال کنترل‌کننده باید قادر به تحمل حجم بیشتری از درخواست جریان جدید باشد.

روش دیگر، روش‌های سیاست کنترل فشار فعال قوانین از کنترل به سوئیچ است. یک مثال خوب از کنترل فعال DIFANE [35]، است که قوانین را در یک سلسله مراتب از سوئیچ‌ها پارتیشن‌بندی می‌کند، به‌طوری که کنترل به ندرت نیاز به جریان جدید و ترافیک در داده‌های نگه داشته شده، داشته باشد. در آزمایشات آنها DIFANE موجب کاهش تاخیر بسته اول از یک میانگین رفت‌وبرگشت ms10 در زمان (RTT) با یک کنترل NOX متمرکز به ms 0.4 RTT متوسط برای جریان جدید تک بسته‌ای می‌شود. این عمل برای افزایش توان عملیاتی جریان جدید نشان داده شده است، به‌عنوان آزمایش نسخه NOX اوج 50،000 جریان تک بسته در هر ثانیه به‌دست آمده است درحالی‌که راه‌حل DIFANE جریان 800،000 تک بسته در هر ثانیه را به‌دست آورده است. جالب توجه است، که کنترلر محلی سوئیچ OpenFlow توسط اجرای یک گلوگاه قبل از کنترلر اصلی NOX مشاهده شده است. این واقعیت تجاری به

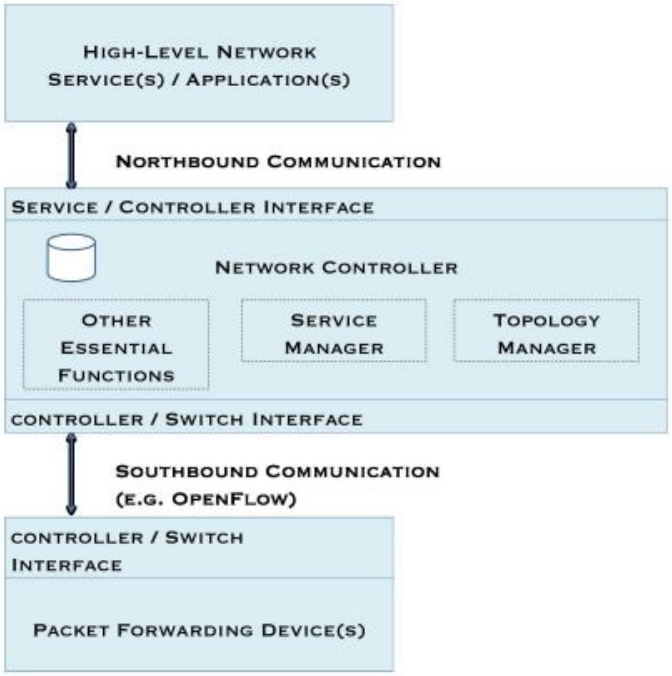
پیاده‌سازی سوئیچ OpenFlow که محدود به ارسال 60-330 جریان درخواست جدید در ثانیه در زمان انتشار خود بود نسبت داده شده است (2010).

همانطور که در شکل 5 نشان داده شده، یک کنترلر که به‌عنوان یک شبکه عمل می‌کند سیستم عامل باید حداقل دو رابط پیاده‌سازی داشته باشد: یکی برای رابط کاربری "جنوب" است که اجازه می‌دهد تا سوئیچ با کنترل ارتباط برقرار کند و دیگری رابط کاربری "شمال" است که یک API به کنترل شبکه و برنامه‌های کاربردی سطح بالا/خدمات ارائه می‌کند.

D) ارتباطات به‌سمت جنوب: کنترلر- سوئیچ

یک جنبه‌ی مهم از SDNs ارتباط بین اطلاعات و کنترل است. همانند عناصر حمل که توسط یک رابط باز کنترل می‌شوند، مهم است که این لینک در دسترس و امن باقی بماند.

پروتکل OpenFlow می‌تواند به‌عنوان یکی از پیاده‌سازی‌های ممکن از فعل و انفعالات کنترل سوئیچ مشاهده شود، همان‌گونه که به‌عنوان ارتباط بین سخت‌افزار و سوئیچینگ شبکه و کنترلر تعریف می‌شود.



شکل 5: یک کنترلر با رابط به‌سمت شمال و به‌سمت جنوب

برای امنیت بیشتر، 1.3.0OpenFlow اختیاری فراهم می‌کند که از امنیت رمزگذاری لایه حمل (TLS) پشتیبانی می‌کند ارتباطات بین سوئیچ و کنترل (بازدیدکنندگان) تغییر می‌یابد؛ بااین‌حال، اجرای دقیق و گواهی فرمت در حال حاضر در TLS مشخص نشده است. همچنین خارج از دامنه مشخصات فعلی، امنیت گزینه‌های ریز دانه مربوط به حالات با کنترل‌های متعدد هستند، که هیچ روش مشخصی برای اعطای مجوزهای دسترسی جزئی به یک کنترل‌کننده مجاز ندارند. ما به بررسی پیاده‌سازی کنترلر OpenFlow با جزئیات بیشتر در بخش چهارم خواهیم پرداخت.

E) ارتباطات به‌سمت شمال: کنترلر- خدمات

سیستم‌های مدیریت خارجی و یا خدمات شبکه ممکن است مایل به استخراج اطلاعات در مورد شبکه‌های زیربنایی و یا یک جنبه از رفتار شبکه و یا سیاست کنترل باشند. علاوه براین، ممکن است کنترلر ملزم به برقراری ارتباط با هر یک از انواع دیگر به دلایل مختلف باشد. برای مثال، یک نرم‌افزار کنترل داخلی ممکن است نیاز به رزرو منابع متعدد در سراسر حوزه کنترل و یا کنترل "اصلی" باشد و ممکن است نیاز به سیاست اطلاعات جهت به اشتراک‌گذاری با یک نسخه پشتیبان تهیه شده باشد، که بر خلاف کنترل سوئیچ ارتباطات است.

در حال حاضر استاندارد پذیرفته شده برای تعامل به‌سمت شمال وجود ندارد و به احتمال زیاد بر اساس برنامه‌های کاربردی خاصی پیاده‌سازی می‌شود. در بخش 6 بیشتر در این مورد بحث خواهیم کرد.

F) تلاش استانداردسازی

TABLE I
CURRENT SOFTWARE SWITCH IMPLEMENTATIONS COMPLIANT WITH THE OPENFLOW STANDARD.

Software Switch	Implementation	Overview	Version
Open vSwitch [55]	C/Python	Open source software switch that aims to implement a switch platform in virtualized server environments. Supports standard management interfaces and enables programmatic extension and control of the forwarding functions. Can be ported into ASIC switches.	v1.0
Pantou/OpenWRT [56]	C	Turns a commercial wireless router or Access Point into an OpenFlow-enabled switch.	v1.0
ofsoftswitch13 [57]	C/C++	OpenFlow 1.3 compatible user-space software switch implementation.	v1.3
Indigo [58]	C	Open source OpenFlow implementation that runs on physical switches and uses the hardware features of Ethernet switch ASICs to run OpenFlow.	v1.0

به‌تازگی، سازمان‌های استاندارد شده‌ی متعددی به سمت SDN تمایل یافته‌اند. به‌عنوان مثال، همانطور که قبلا ذکر شد، حمل IETF و جداسازی اجزاء کنترلی (ForCES) [1] مشغول به کار در مکانیزم استانداردسازی رابط‌ها و پروتکل‌ها با هدف تمرکز برکنترل شبکه و انتزاع زیرساخت‌های شبکه هستند. بنیاد گسترش شبکه (ONF) [3] در حال تلاش برای استاندارد کردن پروتکلOpenFlow است.

به‌عنوان کنترل خلاصه برنامه‌های کاربردی شبکه از زیرساخت‌های سخت‌افزاری زیرین، آنها بر استانداردسازی رابط بین موارد زیر تمرکز دارند: (1) برنامه‌های کاربردی شبکه و کنترلر (به‌عنوان مثال رابط به‌سمت شمال) و (2) کنترلر و زیرساخت‌های سوئیچینگ (به‌عنوان مثال، رابط به‌سمت جنوب) که این را پروتکل OpenFlow خود تعریف می‌کند. برخی از مطالعات گروهی در مورد بخش استاندارد مخابرات ITU (ITU-T) [49] درحال حاضر در مورد الزامات کار و توصیه‌های ایجاد SDNs تحت دیدگاه‌های مختلف بحث می‌کند. به‌عنوان مثال، SG13 بر شبکه‌های آینده، از جمله محاسبات ابری، موبایل و شبکه‌های نسل آینده و ایجاد شرایط لازم برای مجازی‌سازی شبکه تمرکز دارد. ITU-T دیگری مانند SG11 برای پروتکل‌ها و مشخصات آزمون آغاز شده است، در اوایل سال 2013، الزامات و معماری در SDN مورد بحث قرار گرفت. گروه پژوهشی نرم‌افزار با تعریف شبکه (SDNRG) درIRTF نیز بر SDN تمرکز داشته است و تحت دیدگاه‌های مختلف با هدف شناسایی روش‌های جدید است که می‌تواند تحقیقات آینده‌ی چالش‌ها را شناسایی و مورد بحث قرار دهد. برخی از موارد مورد علاقه عبارتند از مقیاس‌پذیری راه‌حل، انتزاع، امنیت و زبان برنامه‌نویسی و پارادایم که در زمینه SDN مفید است.

این گروه و دیگر گروه‌های کاری کار مهمی انجام می‌دهد، هماهنگی تلاش‌ها برای تکامل استانداردهای موجود و پیشنهاد جدید است. هدف، تسهیل انتقال شبکه فن‌آوری به پروتکل‌ها و معماری جدید است، مانند برخی از SDN این گروه، مانند ITU-T SG13، حمایت از ایجاد یک هماهنگی مشترک فعالیت در (JCA-SDN) SDN برای همکاری و هماهنگی بین استاندارد و همچنین تلاش برای استفاده از کار انجام شده توسط نرم‌افزار منبع باز جامعه (OSS)، مانند OpenStack و OpenDayLight به عنوان توسعه بلوک‌های ساختمان برای اجرای SDN است.

4. ابزارهای توسعه SDN

SDN به منظور تسهیل تکامل شبکه و نوآوری با هدف استقرار سریع خدمات پروتکل‌های جدید مطرح شده است. در این بخش، یک نمای کلی از ابزار و محیط در حال حاضر برای توسعه‌ی سرویس‌های مبتنی بر SDN و پروتکل‌ها ارائه می‌کنیم.

A. شبیه‌سازی و ابزار شبیه‌سازی

Mininet [53] اجازه می‌دهد تا یک شبکه OpenFlow برای شبیه‌سازی در یک دستگاه واحد با توسعه اولیه و روند استقرار داشته باشیم. خدمات جدید، برنامه‌ها و پروتکل‌ها در ابتدا می‌توانند بر روی شبیه‌ساز محیط زیست قبل از رفتن به سخت‌افزار واقعی، توسعه یافته و مورد آزمون قرار گیرند. به طور پیش فرض Mininet از 1.0V، OpenFlow حمایت می‌کند هر چند ممکن است برای حمایت از یک سوئیچ نرم‌افزار برای پیاده‌سازی یک نسخه جدیدتر تغییر کند

TABLE II
MAIN CURRENT AVAILABLE COMMODITY SWITCHES BY MAKERS, COMPLIANT WITH THE OPENFLOW STANDARD.

Maker	Switch Model	Version
Hewlett-Packard	8200zl, 6600, 6200zl, 5400zl, and 3500/3500yl	v1.0
Brocade	NetIron CES 2000 Series	v1.0
IBM	RackSwitch G8264	v1.0
NEC	PF5240 PF5820	v1.0
Pronto	3290 and 3780	v1.0
Juniper	Junos MX-Series	v1.0
Pica8	P-3290, P-3295, P-3780 and P-3920	v1.2

Ns-3 [54] شبیه‌ساز شبکه از سوئیچ OpenFlow در محیط خود پشتیبانی می‌کند، هر چند نسخه فعلی تنها در OpenFlow 0.89v اجرا می‌شود.

B. نرم‌افزار موجود پلتفرم سوئیچ

در حال حاضر چند سوئیچ نرم‌افزار SDN در دسترس وجود دارد که می‌توانند مورد استفاده قرار گیرند، برای مثال، اجرای یک بستر آزمایشی SDN یا توسعه خدمات بر روی SDN . جدول 1 یک لیست از پیاده‌سازی نرم‌افزار سوئیچ فعلی با شرح مختصری از جمله زبان پیاده‌سازی و نسخه استاندارد OpenFlow که پیاده‌سازی فعلی پشتیبانی می‌کند را ارائه می‌کند.

C. سوییچ بومی SDN

در حال حاضر یکی از فن‌آوری های اصلی SDN در سخت‌افزار شبکه اجرا شده استاندارد باز جریان است. در این بخش ما قصد نداریم یک مرور دقیق از سخت‌افزار OpenFlow و سازندگان آن ارائه کنیم، بلکه یک لیست از سوئیچ SDN در حال حاضر موجود در بازار و برخی از اطلاعات در مورد آنها، از جمله نسخه پیاده‌سازی OpenFlow ارائه می‌کنیم. یک شاهد روشن از تعهد قوی صنعت برای SDN در دسترس بودن سخت افزار شبکه است که OpenFlow را فعال می‌کند. جدول2 برخی از سوئیچ‌های تجاری که درحال حاضر موجود هستند، تولید کننده و نسخه ای از OpenFlow که آنها اجرا می‌کنند ارائه می‌کند.

D. پلتفرم کنترلر دردسترس

جدول 3 یک عکس فوری از پیاده‌سازی کنترل‌کننده جریان را نشان می‌دهد. تا به‌امروز، تمام کنترلرها در جدول از نسخه‌ی پروتکل 1.0OpenFlowپشتیبانی کرده‌اند. این جدول همچنین خلاصه‌ای از کنترلرهای ذکر شده را فراهم می‌کند.

از جمله در جدول 3 دو کنترلر برای مقاصد خاص پیاد سازی هستند: Flowvisor [48]، که قبلا ذکر شد و RouteFlow [66]. اعمال سابق مانند یک پروکسی شفاف بین سوئیچ OpenFlow و کنترل OpenFlow متعدد عمل می‌کنند. که قادر به ایجاد برش شبکه هستند و می‌تواند کنترل هر قطعه را به یک کنترل‌کننده‌ی مختلف محول

کنند. از سوی دیگر، RouteFlow ، پروژه منبع باز است و به ارائه مسیریابی مجازی IP در سخت‌افزار OpenFlow می‌پردازد. این مسئله از کنترلر OpenFlow، یک سرور مستقل، و یک شبکه مجازی محیط زیست تشکیل شده است که بازتولید اتصال فیزیکی زیرساخت‌ها و موتورهای مسیریابی IP را اجرا می‌کند. موتورهای مسیریابی تولید اطلاعات پایه حمل (FIB) در جداول IP لینوکس با توجه به پروتکل‌های مسیریابی پیکربندی می‌شوند (به‌عنوان مثال،OSPF ، BGP). یک فرمت از RouteFlow در[67] ارائه شده است، که پلتفرم کنترل مسیریابی (RCPs) در زمینه‌ی OpenFlow / SDN را مورد ارزیابی قرار می‌دهد. آنها یک مدل کنترل محور پیشنهاد دادند که با اجرای نمونه اولیه از خدمات مسیریابی مستقلBGP و سیستم گسترده همراه است.

E. کد امنیتی و اشکال‌زدایی

ابزار تأیید و اشکال‌زدائی از منابع حیاتی برای توسعه نرم‌افزار سنتی و کمتر مهم برای SDN است. در واقع، برای این ایده که شبکه‌های قابل حمل "نرم‌افزار" به موفقیت برسند، رفتار شبکه باید به‌طور کامل تست شده و تأیید شود.

NICE [68] یک ابزار تست خودکار برای کمک به کشف اشکالات برنامه در OpenFlow از طریق چک کردن مدل و اعدام نمادین است.

Anteater [69] یک رویکرد متفاوت برای بررسی ویژگی‌های شبکه ارائه داده است که در داده وجود دارد، مانند اتصال و یا ثبات. سود اصلی از این روش پروتکلagnostic است.

تلاش‌های دیگر، ابزار اشکال‌زدایی متفاوتی پیشنهاد کرده‌اند که موجب ارائه ترافیک جمع‌آوری شده از کنترل [71] شد. OFRewind اجازه می‌دهد تا حوادث شبکه (کنترل و داده‌ها) در دانه‌بندی‌های مختلف ثبت شود و بعد از تکثیر یک سناریوی خاص مورد استفاده قرار گیرد. ndb نقاط شکست و backtraces بسته برایSDN را پیاده سازی می‌کند. همان‌طور که با نرم‌افزار محبوب اشکال‌زدایی gdb، کاربران می‌توانند با دقت به حوادث منجر به خطای با توقف اعدام در یک نقطه انفصال و یا، با استفاده از بسته backtrace اشاره کنند. STS [37] یک شبکه با تعریف نرم‌افزار با شبیه‌ساز عیب‌یابی است. که در پایتون نوشته شده است و به POX بستگی دارد. که دستگاه‌های موجود در

یک شبکه را برای تست موارد و شناسایی مجموعه‌ای از ورودی‌ها که منجر به تولید یک خطای داده می‌شود شبیه‌سازی می‌کند.

5. نرم‌افزار SDN

شبکه با تعریف نرم‌افزار، برنامه‌های کاربردی با طیف گسترده‌ای از محیط‌های شبکه‌ای را دربر دارد. توسط افتراق کنترل و داده‌ها، شبکه قادر به برنامه‌ریزی سفارشی کنترل است، که فرصتی برای از بین بردن middleboxes و همچنین توسعه ساده و استقرار شبکه‌های جدید خدمات و پروتکل‌ها را دارد. در زیر، ما محیط‌های مختلف راه‌حل SDN را که پیشنهاد شده است مورد بررسی قرار می‌دهیم.

A. شبکه‌های سازمانی

شرکت‌ها درحال اجرای شبکه‌های بزرگی هستند، درحالی‌که نیازمند داشتن امنیت و عملکرد شدیدی هستند. از این گذشته، محیط شرکت‌های مختلف می‌تواند شرایط بسیار متفاوتی داشته باشد، ویژگی‌ها و تعداد کاربران، به‌عنوان مثال، شبکه دانشگاه می‌تواند یک مورد خاص از شبکه‌های سازمانی در نظر گرفته شود: در چنین محیطی، بسیاری از دستگاه‌های اتصال موقت هستند و توسط دانشگاه کنترل نمی‌شوند، و بیشتر برای به چالش کشیدن امنیت و تخصیص منابع هستند. علاوه بر این، اغلب دانشگاه‌ها باید برای پشتیبانی testbeds تحقیق و پروتکل‌های تجربی ارائه کنند. مدیریت کافی یک انتقاد مهم در محیط‌های شرکت است و SDN می‌تواند برای اجرای برنامه‌نویسی و تنظیم سیاست‌های شبکه و همچنین کمک به نظارت بر فعالیت‌های شبکه و عملکرد شبکه استفاده شود.

علاوه بر این، SDN می‌تواند برای ساده‌سازی شبکه‌های رهایی از middleboxes و یکپارچه‌سازی قابلیت‌های آن‌ها در کنترل شبکه مورد استفاده قرار گیرد. برخی از نمونه‌های قابل توجه قابلیت middlebox که با استفاده از SDN پیاده‌سازی شده است شامل NAT، فایروال‌ها، توازن بار [74] [75] و کنترل دسترسی به شبکه [76] است. در مورد

middleboxes پیچیده‌تر با ویژگی‌هایی که نمی‌توانند به‌طور مستقیم بدون تخریب عملکرد (به‌عنوان مثال، بسته عمیق بازرسی) اجرا شوند، SDN می‌تواند برای ارائه کنترل یکپارچه و مدیریت مورد استفاده قرار گیرد [77].

کار ارائه شده در [78] مربوط به مسائل به‌روزرسانی‌های شبکه سازگار است. تغییرات پیکربندی، یک منبع مشترک بی‌ثبات در شبکه‌ها هستند و می‌تواند منجر به قطع، نقص‌های امنیتی و اختلالات عملکرد شوند. در[78]، مجموعه‌ای از انتزاع سطح بالا ارائه شده که اجازه می‌دهد مدیران شبکه، کل شبکه را برای تضمین این‌که هر بسته در طول پردازش در شبکه دقیقا با پیکربندی شبکه جهانی سازگار است به‌روز رسانی کنند. برای حمایت از این انتزاع، چند به‌روز رسانی بر اساس به مکانیسم OpenFlow توسعه داده شده است.

همانطور که در بخش قبلی بحث شد، OpenFlow از Ethane تکامل یافته است [20]، یک معماری شبکه برای رسیدگی به مسائل مواجه شده شبکه‌های سازمانی طراحی شده است.

B. مراکز داده

مراکز داده با یک سرعت شگفت انگیز در سال‌های اخیر به جلو می‌رود، و به‌طور مداوم در تلاش برای رسیدن به سرعت بالاتر و در حال تغییر تقاضا است. مدیریت ترافیک دقیق و اجرای سیاست، زمانی که در چنین مقیاس بزرگی کنار هم آیند، به‌ویژه هنگامی‌که هر گونه اختلال و یا خدمات اضافی تاخیر ممکن است به بهره‌وری گسترده و/یا از دست دادن سود منجر شود.

زمانی‌که چالش‌های شبکه‌های مهندسی در این مقیاس و پیچیدگی پویا با الزامات برنامه منطبق می‌شود، اغلب مراکز داده‌ها در اوج تقاضا قرار می‌گیرند؛ به‌عنوان یک نتیجه، آنها به خوبی تحت ظرفیت زیاد زمان اجرا می‌شوند اما به سرعت آماده‌یسرویس‌دهی به حجم کار بالاتر می‌باشند.

TABLE III
CURRENT CONTROLLER IMPLEMENTATIONS COMPLIANT WITH THE OPENFLOW STANDARD.

Controller	Implementation	Open Source	Developer	Overview
POX [59]	Python	Yes	Nicira	General, open-source SDN controller written in Python.
NOX [17]	Python/C++	Yes	Nicira	The first OpenFlow controller written in Python and C++.
MUL [60]	C	Yes	Kulcloud	OpenFlow controller that has a C-based multi-threaded infrastructure at its core. It supports a multi-level north-bound interface (see Section III-E) for application development.
Maestro [21]	Java	Yes	Rice University	A network operating system based on Java; it provides interfaces for implementing modular network control applications and for them to access and modify network state.
Trema [61]	Ruby/C	Yes	NEC	A framework for developing OpenFlow controllers written in Ruby and C.
Beacon [22]	Java	Yes	Stanford	A cross-platform, modular, Java-based OpenFlow controller that supports event-based and threaded operations.
Jaxon [62]	Java	Yes	Independent Developers	a Java-based OpenFlow controller based on NOX.
Helios [24]	C	No	NEC	An extensible C-based OpenFlow controller that provides a programmatic shell for performing integrated experiments.
Floodlight [38]	Java	Yes	BigSwitch	A Java-based OpenFlow controller (supports v1.3), based on the Beacon implementation, that works with physical- and virtual- OpenFlow switches.
SNAC [23]	C++	No	Nicira	An OpenFlow controller based on NOX-0.4, which uses a web-based, user-friendly policy manager to manage the network, configure devices, and monitor events.
Ryu [63]	Python	Yes	NTT, OSRG group	An SDN operating system that aims to provide logically centralized control and APIs to create new network management and control applications. Ryu fully supports OpenFlow v1.0, v1.2, v1.3, and the Nicira Extensions.
NodeFlow [64]	JavaScript	Yes	Independent Developers	An OpenFlow controller written in JavaScript for Node.JS [65].
ovs-controller [55]	C	Yes	Independent Developers	A simple OpenFlow controller reference implementation with Open vSwitch for managing any number of remote switches through the OpenFlow protocol; as a result the switches function as L2 MAC-learning switches or hubs.
Flowvisor [48]	C	Yes	Stanford/Nicira	Special purpose controller implementation.
RouteFlow [66]	C++	Yes	CPqD	Special purpose controller implementation.

موضوع بسیارمهم مصرف انرژی است، که یک هزینه‌ی غیر بدیهی در مرکز داده‌ی بزرگ مقیاس دارد. هلر و همکاران [79] نشان می‌دهند که تحقیقات زیادی بر روی سرور و خنک‌کننده (70٪ از کل بهبود انرژی) از طریق سخت‌افزار بهتر و یا مدیریت نرم‌افزار انجام شده است، اما زیرساخت شبکه مرکز داده (که برای10-20٪ از کل هزینه انرژی حساب می‌شود) هنوز هم مصرف 3 میلیارد کیلووات ساعت در سال 2006را دارد. آنها ElasticTree را پیشنهاد دادند که از SDN برای پیدا کردن حداقل توان شبکه که شرایط ترافیک موجود و خاموش کردن سوئیچ‌هایی که مورد نیاز نیست را تحمل می‌کند بهره می‌برد. به عنوان یک نتیجه، آنها صرفه‌جویی انرژی را بین 25-62٪ تحت تغییر شرایط ترافیک نشان می‌دهند. می‌توان تصور کرد که این صرفه‌جویی را می‌توان بیشتر افزایش داد اگر به صورت موازی با مدیریت سرور و مجازی‌سازی استفاده شود ؛ یک امکان Honeyguide است که بهینه‌سازی مصرف انرژی با استفاده از مهاجرت ماشین مجازی برای افزایش تعداد ماشین و سوئیچ که می‌تواند خاموش شوند را در پی دارد.

با این حال، تمام راه‌حل‌های SDN ممکن است در شبکه‌های با عملکرد بالا مناسب نباشند. در حالی که مدیریت ترافیک ساده است و باید به‌طور معقول و متعادل با مقیاس‌پذیری و سربار عملکرد باشد. کورتیس و همکارانش [34] بر این باورند که زوج‌های OpenFlow کنترل مرکزی و با دید کامل هستند، زمانی‌که تنها جریان "مهم" نیاز به مدیریت دارد. هر چند استفاده تهاجمی از سیاست‌های فعال و قوانین ممکن است این مسئله را حل کند، اما ممکن

است توانایی کنترل تضعیف دانه دانه و مدیریت ترافیک و جمع‌آوری آمار موثر را نداشته باشد. چارچوب آنها، DevoFlow، برخی از تغییرات طراحی برای نگه داشتن جریان در داده پیشنهاد می‌کند تا آنجا که ممکن است به‌اندازه کافی برای مدیریت جریان موثر نباشد. بنابراین با هل دادن مسئولیت جریان برگشت به سوئیچ‌ها و اضافه کردن آمار مکانیزم جمع‌آوری کارآمدتر می‌شود. در یک شبیه‌سازی تعادل بار، راه حل آن‌ها 10-53 جریان بار کمتر در جدول بود و 10-42 بار پیام کنترل بیش از OpenFlow بود.

یک مثال عملی از یک برنامه واقعی از مفهوم و معماری SDN در زمینه مراکز داده توسط گوگل در اوایل سال 2012 توسط این شرکت در اجلاس شبکه باز [81] ارائه شد که یک پیاده‌سازی شبکه اتصال مراکز داده مبتنی بر SDN در مقیاس بزرگ بود. کار در [82] با جزئیات بیشتر در طراحی، اجرا و ارزیابی B4 ارائه شده است. این کار توصیف یکی از اولین و بزرگترین استقرار SDN را ارئه می‌کند. انگیزه‌ی اصلی نیاز به مسیریابی سفارشی و مهندسی ترافیک بود و این واقعیت که سطح مقیاس‌پذیری، تحمل خطا، بازده هزینه و کنترل مورد نیاز، نمی‌تواند با استفاده از یک معماری WAN سنتی به‌دست آید. یک راه‌حل پیشنهاد شد و معماری SDN مبتنی بر OpenFlow برای کنترل سوئیچ‌های فردی ساخته شد. پس از سه سال از تولید، نشان داده شد که B4 کارآمد است. علاوه بر این، تجربه نشان می‌دهد که تنگنای ناشی از ارتباطات کنترل به داده و سربار در برنامه‌نویسی سخت‌افزار از مسائل مهم در کارهای آینده در نظر گرفته شود.

C. شبکه‌های دردسترس بی‌سیم مبتنی بر زیرساخت

تلاش‌های زیادی بر زمینه شبکه‌های دردسترس بی‌سیم مبتنی بر زیرساخت‌ها، مانند تلفن همراه و WiFi تمرکز داشته است.

به‌عنوان مثال، پروژه OpenRoads [83]، [84] به دنبال یک جهانی است که در آن کاربران می‌توانند آزادانه و یکپارچه در سراسر زیرساخت‌های مختلف بی‌سیم که ممکن است توسط ارائه‌دهندگان مختلف مدیریت شوند حرکت کنند. آنها یک معماری بی‌سیم مبتنی بر SDN پیشنهاد دادند که سازگار و در عین حال باز و قابل اشتراک بین

ارائه‌دهندگان خدمات مختلف بود. آنها یک بستر آزمایشی با استفاده از دستگاه‌های بی‌سیم- OpenFlow فعال مانند AP های وای‌فای و ایستگاه‌های پایه وایمکس توسط کنترل -NOX و کنترل Flowvisor به‌کار گرفتند و بهبود عملکرد در تحویل را نشان دادند.

اودین [86] محیط LAN برنامه‌ریزی بی‌سیم معرفی کرد. که به‌طور خاص، یک نقطه دسترسی انتزاع بر روی کنترل از نقطه دسترسی فیزیکی مهیا می‌سازد، که قادر به تحرک فعال مدیریت و توازن بار بدون تغییر به مشتری است.

OpenRadio [87] بر استقرار داده‌های بی‌سیم تمرکز دارد که برنامه‌ریزی انعطاف‌پذیری در لایه PHY و MAC (برخلاف لایه3 SDN) فراهم می‌کند در حالی که با عملکرد و زمان مهلت سختی مواجه می‌شود. سیستم برای ارائه یک رابط طراحی شده است که قادر به پردازش زیرمجموعه‌های ترافیک با استفاده از پروتکل‌های مختلف از جمله وای‌فای، وایمکس، GPP LTE3 پیشرفته، و غیره است.

D. شبکه‌های نوری

سیستم‌های انتقال داده به‌عنوان جریان، اجازه می‌دهند تا شبکه با تعریف نرم‌افزار و شبکه‌های OpenFlow به طور خاص، برای حمایت فن‌آوری‌های شبکه‌های متعدد ادغام شوند. به‌عنوان نتیجه، ممکن است ارائه کنترل فن‌آوری اگنوستیک واحد برای شبکه‌های حمل و نقل نوری و تسهیل تعامل بین هر دو بسته و شبکه مدار سوئیچ باشد. بر اساس گروه کاری حمل و نقل نوری (OTW) که در 2013 توسط بنیاد گسترش شبکه (ON)، ایجاد شده است، از مزایای اعمال SDN و استاندارد OpenFlow به‌ویژه به شبکه حمل و نقل نوری عبارتند از: بهبود کنترل شبکه حمل و نقل نوری و انعطاف‌پذیری مدیریت، استقرار مدیریت و کنترل سیستم‌های شخص ثالث و استقرار خدمات جدید با اعمال نفوذ مجازی‌سازی و [88SDN].

تلاش‌ها و طرح‌های پیشنهادی زیادی برای کنترل هر دو مدار سوئیچ و شبکه سوئیچ بسته با استفاده از پروتکل OpenFlow مطرح شده است. در [89] یک پلت‌فرم NetFPGA برای سوئیچینگ بسته و مدار سوئیچ معماری شبکه‌های مبتنی بر سوئیچینگ انتخابی طول موج (WSS)، با استفاده از پروتکل OpenFlow ارائه شده است.

معماری کنترل دیگری در OpenFlow برای فعال کردن SDN بر اساس عملیات در شبکه‌های نوری در [91] پیشنهاد شده است، که شرایط خاصی برای اجرای پسوند پروتکل OpenFlow برای حمایت از شبکه‌های حمل و نقل نوری را توصیف می‌کند.

در این کار، رابط اترنت مجازی (veths) معرفی می‌شود. این veths، به رابط فیزیکی از یک گره نوری (به‌عنوان مثال فوتونیک اتصال متقابل PXC) نگاشت می‌شود و قادر به فعال کردن کنترلر SDN (به‌عنوان مثال کنترلر NOX در این مورد) برای به‌کار بردن هایلایت نوری (به‌عنوان مثال، از طریق پروتکلOpenFlow) است. در راه‌اندازی آزمایش، معیارهای عملکرد شبکه، مانند زمان تاخیر راه‌اندازی و امکان‌سنجی مسیریابی و تخصیص طول موج، کنترل پویای گره نوری در شبکه‌های مبتنی بر OpenFlow تشکیل شده توسط چهار گره PXCs در یک توپولوژی مش مورد ارزیابی قرار گرفت.

معماری شبکه‌ی نوری تعریف شده با نرم‌افزار (SDON) در [93] معرفی شد و پروتکل کنترل QoS برای تعویض پشت سر هم نوری در SDON بر اساس OpenFlow توسعه یافته است. عملکرد پروتکل پیشنهادی با پروتکل مبتنی بر GMPLS معمولی مورد بررسی قرار گرفت و توزیع نتایج نشان داد که SDON یک زیرساخت برای پشتیبانی از پروتکل‌های کنترل یکپارچه برای بهینه‌سازی بهتر عملکرد شبکه و بهبود ظرفیت ارائه می‌دهد.

E. کسب وکار کوچک و خانگی

پروژه‌های زیادی به بررسی چگونگی استفاده‌ی SDN در شبکه‌های کوچکتر، مانند کسانی که در خانه و یا کسب‌وکار کوچک یافت می‌شوند پرداخته است. همانطور که این محیط به‌طور فزاینده‌ای پیچیده و در دسترس است، نیاز به مدیریت شبکه بیشتر و امنیت تنگ‌تر دارد. شبکه‌های امن ضعیف ممکن است به اهدافی نانوشته و یا میزبان برای نرم‌افزارهای مخرب تبدیل شوند، درحالی که با توجه به پیکربندی شبکه ممکن است موجب سرخوردگی و یا از دست دادن کسب‌وکار شود. متاسفانه، تخصیص مدیر به هر شبکه در خانه و دفتر کار میسر نیست.

کالورت و همکارانش [94] ادعا می‌کنند که اولین قدم در مدیریت شبکه‌ی خانه، دانستن آنچه که واقعا رخ داده است. به‌این ترتیب آنها یک شبکه/کنترل دروازه‌ای به‌عنوان یک " شبکه خانگی ضبط داده" برای ایجاد سیاهه‌های مربوط ممکن است برای عیب‌یابی و یا اهداف دیگر استفاده شود، پیشنهاد. کردند.

Feamster [95] پیشنهاد می‌کند که چنین شبکه‌ای باید با "به برق وصل شدن و خارج شدن" عمل کند، یعنی با برون‌سپاری مدیریت کارشناسان شخص ثالث، و این که می‌تواند از طریق برنامه‌ریزی قابل کنترل از راه دور سوئیچ ها اتفاق بیافتد و از شبکه توزیع الگوریتم‌های نظارت برای شناسایی مشکلات امنیتی ممکن استفاده کند.

در مقابل، Mortier و همکارانش [96] بر این باورند که کاربران تمایل درک و کنترل بر رفتار شبکه‌های خود را بیشتر دارند. به‌جای سیاست‌های سنتی، یک شبکه خانگی ممکن است توسط کاربران بهتر خود را مدیریت کند و به درک بهتر و نیازهای محیط خود دست یابد. نسبت به این اهداف، آنها یک شبکه نمونه که در آن SDN برای ایجاد یک دیدگاه که چگونه شبکه درحال استفاده‌ی آنان یک نقطه از یک کنترل است به کاربران استفاده شده است، ایجغاد کردند.

مهدی و همکارانش [97] استدلال می‌کنند که یک سیستم تشخیص ناهنجاری(ADS) که در یک شبکه خانگی قابل برنامه‌ریزی اجرا شده است، شناسایی دقیق فعالیت‌های مخرب را در مقایسه با یک ISP مستقر فراهم می‌کند. الگوریتم ADS می‌تواند در کنار کنترل‌کننده دیگر، مانند HomeOS عمل کند که ممکن است به فعالیت مشکوک و گزارش ناهنجاری ISP یا مدیر محلی واکنش نشان دهد.

6. چالش‌های پژوهش و دستورالعمل‌های آینده

همان‌گونه که SDN به‌طور گسترده پذیرفته شده است و پروتکل‌هایی مانند OpenFlow بیشتر تعریف شده است، راه‌حل‌های جدید ارائه شده و چالش‌های جدید بوجود می‌آیند. در این بخش در مورد چالش‌های مختلف ناشی از SDN و همچنین تحقیقات آینده بحث می‌کنیم: (1) کنترل و طراحی سوئیچ، (2) مقیاس‌پذیری و عملکرد در SDNs، (3)

واسط کنترلر-سرویس ، (4) مجازی‌سازی و برنامه‌های کاربردی سرویس ابری، (5) اطلاعات شبکه محور و (6) شبکه‌های ناهمگن با SDN.

A. طراحی کنترلر و سوئیچ

SDN به‌طور قابل توجهی مقیاس‌پذیری، عملکرد، نیرومندی و چالش‌های امنیتی را افزایش می‌دهد. در زیر به بررسی تعدادی از تحقیقات متمرکز در پرداختن به این مسائل در سطح طراحی سوئیچ و کنترلر می‌پردازیم.

Devoflow رسیدگی به جریان "کوتاه مدت" در سوئیچ ها و جریان "زندگی طولانی" در کنترلرها را برای کاهش جریان راه‌اندازی و تاخیر سربار کنترلر پیشنهاد داده است. کار ارائه شده در [28] بر خلاف ASIC طرفدار جایگزینی توسط یک جریان از قوانین تطبیق و پردازش آنها درCPU تا دسترسی کارآمد بود. FLARE یک مدل جدید گره شبکه با تمرکز بر" شبکه با برنامه‌ریزی عمیق" بود که برنامه‌ریزی برای داده‌ها، کنترل و همچنین به‌عنوان رابط بین آنها را به‌کار می‌برد. کار ارائه شده در [99] از جنبه‌های مهم در طراحی کنترلر از جمله کنترل سلسله مراتبی، مدل داده‌ها، مقیاس‌پذیری و توسعه مورد بحث قرار گرفته است.

با توجه به عملکرد و مقیاس‌پذیری، ارائه شده در مطالعه [100] یک کنترل‌کننده تنها می‌تواند تا 6 میلیون جریان در هر ثانیه را اداره کند. یک مطالعه اخیر [101]، بر روی کنترل‌کننده چراغ تمرکز داشت و نشان داد که یک کنترل‌کننده می تواند 12.8 میلیون جریان جدید در هر ثانیه در یک ماشین 12 هسته‌ای، با تاخیر میانگین 24.7 برای هر جریان اداره کند. بااین‌حال، برای افزایش مقیاس‌پذیری و قابلیت اطمینان به‌ویژه برای اهداف نیرومندی، کنترل منطقی متمرکز باید به‌صورت فیزیکی توزیع شده باشد. اونیکس [44]، Kando [47]، و HyperFlow [45] از این روش برای رسیدن به کنترل قوی و مقیاس‌پذیر استفاده کرده‌اند. در [46]، مبادلات مربوط به توزیع کنترل، مانند staleness در مقابل بهینگی و پیچیدگی منطقی نرم‌افزار در مقابل نیرومندی شناسایی و تعیین شده است. در [41]، مشکل قرار دادن کنترلر به‌معنی تعداد مورد نیاز کنترلر در شبکه مورد بحث قرار گرفته است. در بیشتر کارهای اخیر در کنترل توزیع شده، نیاز به انتساب پویای سوئیچ به کنترل بود که در [102]، که یک الگوریتم برای افزایش یا کاهش پیشنهاد

استخری از کنترلرها بر اساس تخمین بار کنترل است، نشان داده شده است. آنها همچنین یک مکانیسم برای تحویل پویای سوئیچ از یک کنترلر به دیگری پیشنهاد دادند.

در [103] یک نوع SDN با الهام از MPLS همراه با مفاهیم کنترلر لبه و کنترل‌کننده‌های اصلی ارائه شد: ورود و خروج کنترل سوئیچ سابق و رسیدگی به رابط میزبان شبکه، در حالی که دسته دوم سوئیچ اصلی و رابط اپراتور شبکه می‌باشد.

اگر چه کنترل و اندازه‌گیری دو مولفه مهم در مدیریت شبکه هستند، کمی فکر برای طراحی رابط‌های برنامه کاربردی برای اندازه‌گیری مورد نیاز است. کار ارائه شده در [104] یک معماری اندازه‌گیری ترافیک با تعریف نرم‌افزار پیشنهاد می‌کند، که در آن اندازه‌گیری داده‌ها از کنترل جدا هستند.

B. ارتباط شبکه با تعریف نرم‌افزار

اینترنت انقلابی در راه ما به‌عنوان افراد و به‌عنوان یک جامعه، زندگی، کار، رفتار کسب‌وکار، معاشرت، دریافت سرگرمی و غیره بود. در نتیجه، اینترنت به‌عنوان بخشی از زیرساخت‌های حیاتی جامعه ما بسیار شبیه به قدرت، آب و شبکه حمل و نقل در نظر گرفته می‌شود.

مقیاس‌پذیری و عملکرد مورد نیاز از برنامه‌های کاربردی پیچیده و افزاینده‌ای انواع چالش‌ها را با معماری اینترنت فعلی مطرح می‌کند. این مسئله باعث شده است که جامعه به بررسی راه‌حل‌ها بیاندیشد [105]. همان‌گونه که اینترنت به‌عنوان "پرچم روز" رشد می‌کند، مانند استفاده برای "ارتقاء" آرپانت با پروتکل TCP / IP، یکی دیگر از چالش‌های قابل توجه زیرساخت‌های فیزیکی و پروتکل‌ها در حال تکامل است. یک مثال قابل توجه استقرار IPv6 است: با وجود بیش از یک دهه در پیگیری استانداردها و دو رویداد استقرار در سراسر جهان، IPv4 هنوز هم باعث اکثر ترافیک اینترنت است.

بسیاری از کارهای در حال حاضر در SDN، بررسی و یا پیشنهاد راه‌حل در چارچوب یک دامنه اداری منطبق برای کنترل منطقی متمرکز بر مدلSDN است. بااین حال، محیط‌هایی که مدیریت غیرمتمرکز دارند، مانند اینترنت،

پاسخی برای کنترل است که به‌صورت منطقی توزیع شده است. این موضوع به همکاری سیستم‌های مستقل (AS ها) اجازه خواهد داد تا به‌طور مستقل توسط کنترلر خود کنترل شوند (تمرکز منطقی و احتمالا از لحاظ فیزیکی توزیع‌شده). چندین تلاش، ایده‌ی اینترنت با تعریف نرم‌افزار را بررسی کرده‌اند. به‌عنوان مثال، [106] یک معماری اینترنت با تعریف نرم‌افزار که از MPLS جهت تمایز بین لبه‌ی شبکه و هسته برای تقسیم وظایف بین دامنه و درون اجزاء دامنه بهره گرفته است، ارائه کرده است. به‌عنوان تنها روتر مرزی و کنترلر مرتبط به هر حوزه در انجام وظایف بین دامنه، تغییرات مدل‌های سرویس بین دامنه درگیر خواهد بود و به تغییرات نرم‌افزار در کنترل بین دامنه نه از کل زیرساخت‌ها محدود است. نمونه‌هایی از این معماری می‌تواند برای تحقق بخشیدن به خدمات اینترنت جدید مانند شبکه‌های اطلاعات محور و خدمات middlebox به‌اشتراک‌گذاری شده برای کاوش مورد استفاده قرار گیرد.

رویکرد دیگر مسیریابی به درون AS [107] از NOX و OpenFlow برای پیاده‌سازی قابلیتBGP استفاده می‌کند. روش دیگر، پروتکل جلسه توسعه یافته است که [108] از پیکربندی مشتق شده از برنامه‌های کاربردی از منابع سراسر شبکه پشتیبانی می‌کند.

C. تعامل کنترلر-سرویس

در حالی که تعامل کنترلر سوئیچ ("به‌سمت جنوب") نسبتا در پروتکل‌هایی مانند OpenFlow و ForCES به‌خوبی تعریف شده است، هیچ استانداردی برای تعامل بین کنترل‌کننده و خدمات شبکه و یا برنامه‌های کاربردی ("به‌سمت شمال") وجود ندارد. یک ممکن توضیح این است که رابط به‌سمت شمال به‌طور کامل در نرم‌افزار تعریف شده است، در حالی‌که تعاملات کنترلر سوئیچ باید در پیاده‌سازی سخت‌افزاری فعال شود.

اگر ما در مورد کنترلر به‌عنوان یک " سیستم عامل شبکه" فکر کنیم، پس باید یک رابط واضح تعریف کنیم که توسط آن بتوان به برنامه‌های کاربردی سخت‌افزار (سوئیچ‌ها)، همکاری و برقراری ارتباط با برنامه‌های کاربردی دیگر و استفاده از خدمات سیستم (به‌عنوان مثال کشف توپولوژی، حمل‌ونقل)، بدون نیاز به دانستن اجرای جزئیات کنترل از

توسعه‌دهنده‌ی نرم‌افزار دسترسی یافت. در حالی‌که چند کنترلر وجود دارد و رابط برنامه آن‌ها هنوز هم در مراحل اولیه و مستقل از یکدیگر است.

برخی از پیشنهادات (به‌عنوان مثال، Procera [109]، Frenetic [110]، FML [111]، Nettle [112]) طرفدار استفاده از یک زبان پیکربندی شبکه برای بیان سیاست‌ها هستند. به‌عنوان مثال، Procera [109]یک لایه سیاست در بالای کنترلرهای موجود برای رابط کاربری با فایل‌های پیکربندی، GUI ها و سنسورهای خارجی ایجاد کرد؛ لایه سیاست پیشنهادی مسئول تبدیل سیاست سطح بالا به جریان محدودیت‌های داده شده برای استفاده توسط کنترلر است. در [113]، تنظیمات شبکه و مدیریت مکانیسم‌ها پیشنهاد شده‌اند که به فعال کردن تغییرات شرایط شبکه و دولت، حمایت از پیکربندی شبکه و تعاریف سیاست، و ارائه دید و کنترل وظایف برای تشخیص شبکه و عیب‌یابی تمرکز دارند.

علاوه بر این، API به‌سمت شمال باید به برنامه‌ها اجازه‌ی اعمال سیاست‌های مختلف به جریان یکسان را بدهد (مثلا حمل توسط مقصد و نظارت توسط منبع IP). کار ارائه شده در [114] یک مدولار را برای اطمینان از قوانین نصب شده برای اجرای یک وظیفه که قوانین دیگر را باطل نمی‌کند پیشنهاد می‌کند.

تا زمانی که یک رابط استاندارد به‌سمت شمال واضح است، SDN برنامه‌های کاربردی تا توسعه‌ی "ad hoc" و مفهوم انعطاف‌پذیری و قابل حمل بودن " برنامه‌های شبکه " ممکن است صبر کند.

D. مجازی‌سازی و سرویس‌های ابر

تقاضا برای مجازی‌سازی و سرویس‌های ابر به‌سرعت در حال رشد و جذب قابل توجه صنعت و دانشگاه است. چالش‌های ارائه‌شده‌ی آن شامل تأمین سریع، مدیریت کارآمد منابع و مقیاس‌پذیری است که می‌تواند با استفاده از مدل کنترل SDN مورد خطاب قرار گیرد.

به‌عنوان مثال، FlowVisor [48] و AutoSlice [115] برش‌های متنوعی از منابع شبکه ایجاد می‌کنند (به‌عنوان مثال، پهنای باند، توپولوژی، CPU، جدول حمل‌ونقل) و آنها را به کنترلرهای مختلف و اجرای انزوا بین برش‌ها واگذار

می‌کنند. کنترلر SDN دیگری که می‌توان به‌عنوان یک شبکه استفاده کرد برای حمایت از مجازی‌سازی در سیستم عامل‌های ابر مانند Floodlight برای OpenStack [38] و NOX برای Mirage [116] است. FlowN [117] به‌دنبال ارائه یک راه‌حل مقیاس‌پذیر برای مجازی‌سازی شبکه‌ها با ارائه‌ی یک نقشه کارآمد بین شبکه‌های مجازی و فیزیکی با اعمال نفوذ سیستم‌های پایگاه داده مقیاس‌پذیر است.

در [118] یک الگوریتم برای مهاجرت کارآمد با تضمین پهنای باند با استفاده از OpenFlow پیشنهاد شده است. LIME [119]یک راهحل مبتنی بر SDN برای مهاجرت از ماشین‌های مجازی است، که در طول مهاجرت، وضعیت شبکه و تنظیم خودکار دستگاه‌های شبکه در مکان‌های جدید را بر عهده دارد. NetGraph [120]مجموعه‌ای از API ها را برای مشتریان برای دسترسی توابع شبکه مجازی خود مانند نظارت بر زمان واقعی و تشخیص را فراهم می‌کند.

در مراکز داده‌های ابر، زیرساخت به‌عنوان یک سرویس (IaaS)، [121] یک چارچوب مدیریت برای منابع در مراکز داده ابری و مسائل متعدد مربوط به مدیریت را ارائه می‌کند. در این مقاله، نویسندگان یک مرکز داده و معماری رویداد محور با رابط مدیریت باز را پیشنهاد می‌کنند، که تکنیک SDN موجب ادغام منابع شبکه با مرکز داده و تأمین خدمات با هدف بهبود خدمات در سطح موافقت و ارائه خدمات سریع‌تر می‌شود.

E. شبکه‌ی اطلاعات محور

شبکه اطلاعات محور (ICN) یک پارادایم جدید است که برای معماری آینده‌ی اینترنت با هدف افزایش بهره‌وری از تحویل محتوا و در دسترس بودن محتوا پیشنهاد شده است. این مفهوم جدید به‌تازگی توسط تعدادی از پیشنهادات معماری، مانند شبکه‌ی محتوا محور (CCN) محبوبیت یافته است، همچنین با نام پروژه شبکه داده‌ها (NDN)[122] شناخته شده است. انگیزه‌ی آنها این است که اینترنت فعلی اطلاعات محور است و تکنولوژی شبکه هنوز هم در ایده تمرکز مبتنی بر مکان و میزبان‌به‌میزبان است. با پیشنهاد یک معماری که داده را به جای میزبان نشان می‌دهد، توزیع محتوا به‌طور مستقیم در علی‌رغم پیچیدگی، در دسترس بودن، و مکانیسم‌های امنیتی در شبکه اجرا می‌شود.

جدایی بین پردازش اطلاعات و حمل‌ونقل در ICN با جدایی تراز داده و کنترل در SDN متفاوت است. سوالی که در اینجا پرسیده می‌شود چگونگی ترکیب ICN با SDN نسبت به " شبکه با تعریف نرم‌افزار اطلاعات محور" است. تعدادی از پروژه‌های [123]، استفاده از SDN[124]، [125]، [126]، [127]، [128] برای مفاهیم پیاده‌سازی ICNS را پیشنهاد کرده‌اند. همانگونه که OpenFlow برای حمایت از تطابق هدر سفارشی گسترش می‌یابد، SDN می‌تواند به‌عنوان یک کلید فن‌آوری برای ICNS به‌کار رود.

F. پشتیبانی ناهمگن شبکه

شبکه‌های آینده به‌طور فزاینده‌ای جهت اتصال کاربران و برنامه‌های کاربردی بر روی شبکه‌ها اعم از سیمی، بی‌سیم مبتنی بر زیرساخت (به‌عنوان مثال، شبکه‌های تلفن همراه بر اساس شبکه‌های مش بی‌سیم)، به شبکه‌های بی سیم (به‌عنوان مثال شبکه‌های ad-hoc تلفن همراه، شبکه‌های فضایی) ناهمگن خواهند بود. در این میان، ترافیک تلفن‌همراه در طول چند سال گذشته افزایش نمایی داشته و انتظار می‌رود 18 برابر تا سال 2016، با دستگاه‌های متصل تلفن همراه از جمعیت جهان افزایش یابد، این مسئله درحال‌حاضر یک واقعیت است [129]. دستگاه‌های تلفن همراه با چند رابط شبکه به امری عادی تبدیل شده است، کاربران تقاضا خدمات ارتباطی با کیفیت بالا بدون در نظر گرفتن محل و یا نوع دسترسی به شبکه را خواهند داشت. شبکه‌های خود سازمانده (به‌عنوان مثال، شبکه‌های ad-hoc بی‌سیم چند هاپ) ممکن است دامنه شبکه‌های مبتنی بر زیرساخت و یا رسیدگی به اختلالات اتصال اپیزودیک را گسترش دهند. در نتیجه شبکه‌های خود سازمانده ممکن است انواع برنامه‌های کاربردی جدید مانند خدمات مبتنی بر ابر، ارتباطات فضایی، خدمات اجتماعی، ارائه مراقبت‌های بهداشتی، واکنش‌های اضطراری و نظارت بر محیط زیست را فعال کنند. تحویل کارآمد محتوای شبکه‌های در دسترس بی‌سیم به امری ضروری تبدیل خواهد شد، و شبکه‌های خود-سازمانده ممکن است تبدیل به بخشی شایع در اینترنت ترکیبی آینده شوند.

چالش عمده در مواجه با شبکه‌های آینده استفاده‌ی کارآمد از منابع است ؛ این امر به‌ویژه در مورد شبکه‌های چند هاپ بی‌سیم موقت، به‌عنوان ظرفیت بی‌سیم در دسترس، ذاتا محدود است. این امر به دلیل تعدادی از عوامل از جمله

استفاده از رسانه‌های فیزیکی به‌اشتراک گذاشته‌ی مرکب، اختلالات کانال‌های بی‌سیم و عدم وجود مدیریت زیرساخت است. هر چند این شبکه خود سازمانده می‌تواند برای تکمیل و یا "پر کردن شکاف" در زیرساخت [130] مورد استفاده قرار گیرد. ویژگی‌های ناهمگن شبکه (مانند محیط فیزیکی، توپولوژی، ثبات) و گره (به‌عنوان مثال، اندازه بافر، محدودیت قدرت، تحرک) نیز یکی دیگر از عوامل مهم با توجه به مسیریابی و تخصیص منابع است.

SDN پتانسیل تسهیل استقرار و مدیریت برنامه‌های کاربردی شبکه و خدمات با بهره‌وری بیشتر را دارد. بااین‌حال، تکنیک‌های SDN به‌روز، مانند جریان باز، تا حد زیادی زیرساخت مبتنی بر شبکه را مورد هدف قرار می‌دهد. آنها یک مکانیسم کنترل متمرکز را که برای عدم تمرکز و اختلال نامناسب است ترویج می‌دهند درحال‌حاضر تاخیر در محیط زیرساخت کمتر است.

درحالی‌که کارهای قبلی استفاده از SDN در محیط‌های بی‌سیم را مورد آزمایش قرار داده‌اند، در درجه اول بر استقرار مبتنی بر زیرساخت‌ها تمرکز کردیم (به‌عنوان مثال، وایمکس، دسترسی Wi-Fi). یک مثال قابل توجه این پروژه OpenRoads [83]است، که در آن کاربران می‌توانند آزادانه بین پیش‌بینی زیرساخت‌های بی‌سیم حرکت کنند درحالی‌که از ارائه‌دهنده‌ی شبکه حمایت می‌کنند. مطالعات دیگر مانند [128]، [131]، [132] OpenFlow در محیط‌های بی‌سیم مش بررسی کرده‌اند.

7. نتیجه‌گیری

در این مقاله، یک نمای کلی از برنامه‌ریزی شبکه ارائه شده و در این زمینه، به بررسی شبکه نوظهور با تعریف نرم‌افزار (SDN) پرداختیم، باا نگاه به شبکه‌های قابل برنامه‌ریزی، مشاهده می‌کنیم این مسئله از تحولات ایده‌های اولیه تا اخیر است. به‌طور خاص معماری SDN با جزئیات و همچنین استاندارد OpenFlow [2]توصیف شد. پیاده‌سازی SDN فعلی و تست سیستم‌عامل و خدمات شبکه و برنامه‌های کاربردی ارائه و مورد بررسی قرار گرفتند که بر اساس پارادایم SDN توسعه یافته بودند. با بحث در مورد آینده‌ی SDN به پشتیبانی از شبکه‌های ناهمگن به شبکه اطلاعات محور (ICN) رسیدیم.

REFERENCES

[1] A. Doria, J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810 (Proposed Standard), March 2010.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Commun. Review, 38(2):69–74, 2008.

[3] Open networking foundation. https://www.opennetworking.org/about.

[4] Open Networking Research Center (ONRC). http://onrc.net.

[5] Thomas A. Limoncelli. Openflow: a radical new idea in networking. Commun. ACM, 55(8):42–47, August 2012.

[6] A.T. Campbell, I. Katzela, K. Miki, and J. Vicente. Open signaling for atm, internet and mobile networks (opensig'98). ACM SIGCOMM Computer Commun. Review, 29(1):97–108, 1999.

[7] A. Doria, F. Hellstrand, K. Sundell, and T. Worster. General Switch Management Protocol (GSMP) V3. RFC 3292 (Proposed Standard), June 2002.

[8] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden. A survey of active network research. IEEE Commun. Mag., 35(1):80–86, 1997.

[9] D.L. Tennenhouse and D.J. Wetherall. Towards an active network architecture. In DARPA Active NEtworks Conf. and Exposition, 2002. Proc., pages 2–15. IEEE, 2002.

[10] J.T. Moore and S.M. Nettles. Towards practical programmable packets. In Proc. 20th Conf. on Computer Commun. (INFOCOM). Citeseer, 2001.

[11] Devolved Control of ATM Networks. http://www.cl.cam.ac.uk/research/srg/netos/old-projects/dcan/#pub.

[12] J. E. Van Der Merwe and I. M. Leslie. Switchlets and dynamic virtual atm networks. In Proc Integrated Network Management V, pages 355– 368. Chapman and Hall, 1997.

[13] J.E. Van der Merwe, S. Rooney, I. Leslie, and S. Crosby. The tempesta practical framework for network programmability. IEEE Netw., 12(3):20–28, 1998.

[14] J. Rexford, A. Greenberg, G. Hjalmtysson, D.A. Maltz, A. Myers, G. Xie, J. Zhan, and H. Zhang. Network-wide decision making: Toward a wafer-thin control plane. In Proc. HotNets, pages 59–64. Citeseer, 2004.

[15] A. Greenberg, G. Hjalmtysson, D.A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4d approach to network control and management. ACM SIGCOMM Computer Commun. Review, 35(5):41–54, 2005.

[16] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In Proc. 2nd conf. on Symp. on Networked Systems Design & Implementation-Volume 2, pages 15–28. USENIX Association, 2005.

[17] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. ACM SIGCOMM Computer Commun. Review, 38(3):105–110, 2008.

[18] R. Enns. NETCONF Configuration Protocol. RFC 4741 (Proposed Standard), December 2006. Obsoleted by RFC 6241.

[19] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin. Simple network management protocol (snmp), rfc1157, 1990.

[20] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. ACM SIGCOMM Computer Commun. Review, 37(4):1–12, 2007.

[21] Z. Cai, AL Cox, and TSE Ng. Maestro: A system for scalable openflow control. Technical Report TR10-08, Rice University, December 2010.

[22] Beacon. https://openflow.stanford.edu/display/Beacon/Home.

[23] Simple Network Access Control (SNAC). http://www.openflow.org/wp/snac/.

[24] Helios by nec. http://www.nec.com/.

[25] Andrea Passarella. Review: A survey on content-centric technologies for the current internet: Cdn and p2p solutions. Comput. Commun., 35(1):1–32, January 2012.
[26] Zhiliang Wang, Tina Tsou, Jing Huang, Xingang Shi, and Xia Yin. Analysis of Comparisons between OpenFlow and ForCES, March 2012.
[27] Guohan Lu, Rui Miao, Yongqiang Xiong, and Chuanxiong Guo. Using cpu as a traffic co-processing unit in commodity switches. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 31–36, New York, NY, USA, 2012. ACM
[28] Jeffrey C. Mogul and Paul Congdon. Hey, you darned counters!: get off my asic! In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 25–30, New York, NY, USA, 2012. ACM.
[29] Yan Luo, Pablo Cascon, Eric Murray, and Julio Ortega. Accelerating openflow switching with network processors. In Proc. 5th ACM/IEEE Symp. on Architectures for Networking and Commun. Syst., ANCS '09, pages 70–71, New York, NY, USA, 2009. ACM.
[30] Voravit Tanyingyong, Markus Hidell, and Peter Sj¨odin. Improving pcbased openflow switching performance. In Proc. 6th ACM/IEEE Symp. on Architectures for Networking and Commun. Syst., ANCS '10, pages 13:1–13:2, New York, NY, USA, 2010. ACM.
[31] A. Bianco, R. Birke, L. Giraudo, and M. Palacin. Openflow switching: Data plane performance. In 2010 IEEE Int. Conf. Commun. (ICC), pages 1–5, May.
[32] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. TranGia. Modeling and performance evaluation of an openflow architecture. In Teletraffic Congress (ITC), 2011 23rd Int., pages 1–7, Sept.
[33] Brent Stephens, Alan Cox, Wes Felter, Colin Dixon, and John Carter. Past: scalable ethernet for data centers. In Proc. 8th int. conf. on Emerging networking experiments and technologies, CoNEXT '12, pages 49–60, New York, NY, USA, 2012. ACM.
[34] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. Devoflow: scaling flow management for high-performance networks. SIGCOMM Comput. Commun. Rev., 41(4):254–265, August 2011.
[35] M. Yu, J. Rexford, M.J. Freedman, and J. Wang. Scalable flow-based networking with difane. In Proc. ACM SIGCOMM 2010 conf. on SIGCOMM, pages 351–362. ACM, 2010.
[36] Yossi Kanizo, David Hay, and Isaac Keslassy. Palette: Distributing tables in software-defined networks. In INFOCOM, pages 545–549, 2013.
[37] Nanxi Kang, Zhenming Liu, Jennifer Rexford, and David Walker. Optimizing the one big switch abstraction in software-defined networks.
[38] Floodlight, an open sdn controller. http://floodlight.openflowhub.org/.
[39] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood. On controller performance in software-defined networks. In USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), 2012.
[40] Andreas Voellmy and Junchang Wang. Scalable software defined network controllers. In Proc. ACM SIGCOMM 2012 conf. on Applications, technologies, architectures, and protocols for computer commun., SIGCOMM '12, pages 289–290, New York, NY, USA, 2012. ACM.
[41] Brandon Heller, Rob Sherwood, and Nick McKeown. The controller placement problem. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 7–12, New York, NY, USA, 2012. ACM.
[42] K. Phemius and M. Bouet. Openflow: Why latency does matter. In Integrated Network Management (IM 2013), 2013 IFIP/IEEE Int. Symp. on, pages 680–683, 2013.
[43] S.H. Yeganeh, A. Tootoonchian, and Y. Ganjali. On scalability of software-defined networking. IEEE Commun. Mag., 51(2):136–141, February.
[44] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al. Onix: A distributed control platform for large-scale production networks. OSDI, Oct, 2010.
[45] A. Tootoonchian and Y. Ganjali. Hyperflow: A distributed control plane for openflow. In Proc. 2010 internet network management conf. on Research on enterprise netw., pages 3–3. USENIX Association, 2010.

[46] Dan Levin, Andreas Wundsam, Brandon Heller, Nikhil Handigol, and Anja Feldmann. Logically centralized?: state distribution trade-offs in software defined networks. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 1–6, New York, NY, USA, 2012. ACM.
[47] Soheil Hassas Yeganeh and Yashar Ganjali. Kandoo: a framework for efficient and scalable offloading of control applications. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 19–24, New York, NY, USA, 2012. ACM.
[48] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.Y. Huang, P. Kazemian, M. Kobayashi, J. Naous, et al. Carving research slices out of your production networks with openflow. ACM SIGCOMM Computer Commun. Review, 40(1):129–130, 2010.
[49] http://www.itu.int/en/ITU T/sdn/Pages/default.aspx. Itu telecommunication standardization sector's sdn portal, 2013.
[50] http://irtf.org/sdnrg. Software-defined networking research group - sdnrg at irtf, 2013.
[51] http://www.openstack.org/. Openstack, 2013.
[52] http://www.opendaylight.org/. Opendaylight, 2013.
[53] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Netw., 2010.
[54] T.R. Henderson, M. Lacage, G.F. Riley, C. Dowell, and JB Kopena. Network simulations with the ns-3 simulator. SIGCOMM demonstration, 2008.
[55] Open vswitch and ovs-controller. http://openvswitch.org/.
[56] Pantou: Openflow 1.0 for openwrt. http://www.openflow.org/wk/index.php/ OpenFlow 1.0 for OpenWRT.
[57] ofsoftswitch13 - cpqd. https://github.com/CPqD/ofsoftswitch13.
[58] Indigo: Open source openflow switches. http://www.openflowhub.org/display/Indigo/. [59] Pox. http://www.noxrepo.org/pox/about-pox/.
[60] Mul. http://sourceforge.net/p/mul/wiki/Home/.
[61] Trema openflow controller framework. https://github.com/trema/trema.
[62] Jaxon:java-based openflow controller. http://jaxon.onuos.org/.
[63] Ryu. http://osrg.github.com/ryu/.
[64] The nodeflow openflow controller. http://garyberger.net/?p=537.
[65] Node.js. http://nodejs.org/.
[66] Marcelo R. Nascimento, Christian E. Rothenberg, Marcos R. Salvador, Carlos N. A. Corrˆea, Sidney C. de Lucena, and Maur´ıcio F. Magalh˜aes. Virtual routers as a service: the routeflow approach leveraging software-defined networks. In Proc. 6th Int. Conf. on Future Internet Technol., CFI '11, pages 34–37, New York, NY, USA, 2011. ACM.
[67] Christian Esteve Rothenberg, Marcelo Ribeiro Nascimento, Marcos Rogerio Salvador, Carlos Nilton Araujo Corrˆea, Sidney Cunha de Lucena, and Robert Raszuk. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 13–18, New York, NY, USA, 2012. ACM.
[68] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford. A nice way to test openflow applications. NSDI, Apr, 2012.
[69] Haohui Mai, Ahmed Khurshid, Rachit Agarwal, Matthew Caesar, P. Brighten Godfrey, and Samuel Talmadge King. Debugging the data plane with anteater. In Proc. ACM SIGCOMM 2011 conf., SIGCOMM '11, pages 290–301, New York, NY, USA, 2011. ACM.
[70] Ahmed Khurshid, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. Veriflow: verifying network-wide invariants in real time. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 49–54, New York, NY, USA, 2012. ACM.
[71] Andreas Wundsam, Dan Levin, Srini Seetharaman, and Anja Feldmann. Ofrewind: enabling record and replay troubleshooting for networks. In Proc. 2011 USENIX conf. on USENIX annu. technical conf., USENIXATC'11, pages 29–29, Berkeley, CA, USA, 2011. USENIX Association.

[72] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazi´eres, and Nick McKeown. Where is the debugger for my softwaredefined network? In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 55–60, New York, NY, USA, 2012. ACM.
[73] Sdn troubleshooting simulator. http://ucb-sts.github.com/sts/.
[74] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari. Plug-n-serve: Load-balancing web traffic using openflow. ACM SIGCOMM Demo, 2009.
[75] R. Wang, D. Butnariu, and J. Rexford. Openflow-based server load balancing gone wild. In Workshop of HotICE, volume 11, 2011.
[76] A.K. Nayak, A. Reimers, N. Feamster, and R. Clark. Resonance: Dynamic access control for enterprise networks. In Proc. 1st ACM workshop on Research on enterprise networking, pages 11–18. ACM, 2009.
[77] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella. Toward softwaredefined middlebox networking. 2012.
[78] Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. In Proc. ACM SIGCOMM 2012 conf. on Applications, technologies, architectures, and protocols for computer commun., SIGCOMM '12, pages 323–334, New York, NY, USA, 2012. ACM.
[79] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In Proc. 7th USENIX conf. on Networked systems design and implementation, pages 17–17. USENIX Association, 2010.
[80] H. Shirayanagi, H. Yamada, and K. Kono. Honeyguide: A vm migration-aware network topology for saving energy consumption in data center networks. In 2012 IEEE Symp. Computers and Commun. (ISCC), pages 000460–000467. IEEE, 2012.
[81] Inter-datacenter wan with centralized te using sdn and openflow. In Open Networking Summit, April 2012.
[82] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. In Proc. ACM SIGCOMM 2013 conf. on SIGCOMM, pages 3–14. ACM, 2013.
[83] K.K. Yap, R. Sherwood, M. Kobayashi, T.Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar. Blueprint for introducing innovation into wireless mobile networks. In Proc. 2nd ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, pages 25–32. ACM, 2010.
[84] K.K. Yap, M. Kobayashi, R. Sherwood, T.Y. Huang, M. Chan, N. Handigol, and N. McKeown. Openroads: Empowering research in mobile networks. ACM SIGCOMM Computer Commun. Review, 40(1):125–126, 2010.
[85] L.E. Li, Z.M. Mao, and J. Rexford. Toward software-defined cellular networks. In Software Defined Networking (EWSDN), 2012 European Workshop on, pages 7–12, 2012.
[86] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. Towards programmable enterprise wlans with odin. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 115–120, New York, NY, USA, 2012. ACM.
[87] M. Bansal, J. Mehlman, S. Katti, and P. Levis. Openradio: a programmable wireless dataplane. In Proc. 1st workshop on Hot topics in software defined networks, pages 109–114. ACM, 2012.
[88] Optical transport working group otwg. In Open Networking Foundation ONF, 2013.
[89] V. Gudla, S. Das, A. Shastri, G. Parulkar, N. McKeown, L. Kazovsky, and S. Yamashita. Experimental demonstration of openflow control of packet and circuit switches. In Optical Fiber Commun. (OFC), collocated National Fiber Optic Engineers Conf., 2010 Conf. on (OFC/NFOEC), pages 1–3, 2010.
[90] Netfpga platform. http://netfpga.org.
[91] Dimitra E. Simeonidou, Reza Nejabati, and Mayur Channegowda. Software defined optical networks technology and infrastructure: Enabling software-defined optical network operations. In Optical Fiber Commun. Conf./National Fiber Optic Engineers Conf. 2013, page OTh1H.3. Optical Society of America, 2013.
[92] Lei Liu, T. Tsuritani, I. Morita, Hongxiang Guo, and Jian Wu. Openflow-based wavelength path control in transparent optical networks: A proof-of-concept demonstration. In Optical Commun. (ECOC), 2011 37th European Conf. and Exhibition on, pages 1–3, 2011.

[93] A.N. Patel, P.N. Ji, and Ting Wang. Qos-aware optical burst switching in openflow based software-defined optical networks. In Optical Netw. Design and Modeling (ONDM), 2013 17th Int. Conf. on, pages 275– 280, 2013.
[94] K.L. Calvert, W.K. Edwards, N. Feamster, R.E. Grinter, Y. Deng, and X. Zhou. Instrumenting home networks. ACM SIGCOMM Computer Commun. Review, 41(1):84–89, 2011.
[95] N. Feamster. Outsourcing home network security. In Proc. 2010 ACM SIGCOMM workshop on Home networks, pages 37–42. ACM, 2010.
[96] R. Mortier, T. Rodden, T. Lodge, D. McAuley, C. Rotsos, AW Moore, A. Koliousis, and J. Sventek. Control and understanding: Owning your home network. In 2012 4th Int. Conf. on Commun. Syst. and Netw. (COMSNETS), , pages 1–10. IEEE, 2012.
[97] S. Mehdi, J. Khalid, and S. Khayam. Revisiting traffic anomaly detection using software defined networking. In Recent Advances in Intrusion Detection, pages 161–180. Springer, 2011.
[98] Akihiro Nakao. Flare : Open deeply programmable network node architecture. http://netseminar.stanford.edu/10 18 12.html.
[99] R. Bifulco, R. Canonico, M. Brunner, P. Hasselmeyer, and F. Mir. A practical experience in designing an openflow controller. In Software Defined Networking (EWSDN), 2012 European Workshop on, pages 61–66, Oct.
[100] Controller performance comparisons. http://www.openflow.org/wk/ index.php/Controller Performance Comparisons. [101] David Erickson. The beacon openflow controller, 2012.
[102] Advait Dixit, Fang Hao, Sarit Mukherjee, T.V. Lakshman, and Ramana Kompella. Towards an elastic distributed sdn controller. In Proc. 2nd ACM SIGCOMM workshop on Hot topics in software defined networking, HotSDN '13, pages 7–12, New York, NY, USA, 2013. ACM.
[103] Martin Casado, Teemu Koponen, Scott Shenker, and Amin Tootoonchian. Fabric: a retrospective on evolving sdn. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 85–90, New York, NY, USA, 2012. ACM.
[104] Minlan Yu, Lavanya Jose, and Rui Miao. Software defined traffic measurement with opensketch. In Proc. 10th USENIX Symp. on Networked Systems Design and Implementation, NSDI'13, 2013.
[105] Anja Feldmann. Internet clean-slate design: what and why? SIGCOMM Comput. Commun. Rev., 37(3):59–64, July 2007.
[106] Barath Raghavan, Mart´ın Casado, Teemu Koponen, Sylvia Ratnasamy, Ali Ghodsi, and Scott Shenker. Software-defined internet architecture: decoupling architecture from infrastructure. In Proc. 11th ACM Workshop on Hot Topics in Networks, HotNets-XI, pages 43–48, New York, NY, USA, 2012. ACM.
[107] R. Bennesby, P. Fonseca, E. Mota, and A. Passito. An inter-as routing component for software-defined networks. In Network Operations and Management Symp. (NOMS), 2012 IEEE, pages 138–145, 2012.
[108] E. Kissel, G. Fernandes, M. Jaffee, M. Swany, and M. Zhang. Driving software defined networks with xsp. In SDN12: Workshop on Software Defined Networks, 2012.
[109] Andreas Voellmy, Hyojoon Kim, and Nick Feamster. Procera: a language for high-level reactive network control. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 43–48, New York, NY, USA, 2012. ACM.
[110] Nate Foster, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. Frenetic: a network programming language. In Proc. 16th ACM SIGPLAN int. conf. on Functional programming, ICFP '11, pages 279–291, New York, NY, USA, 2011. ACM.
[111] Timothy L. Hinrichs, Natasha S. Gude, Martin Casado, John C. Mitchell, and Scott Shenker. Practical declarative network management. In Proc. 1st ACM workshop on Research on enterprise networking, WREN '09, pages 1–10, New York, NY, USA, 2009. ACM.
[112] Andreas Voellmy and Paul Hudak. Nettle: taking the sting out of programming network routers. In Proc. 13th int. conf. on Practical aspects of declarative languages, PADL'11, pages 235–249, Berlin, Heidelberg, 2011. Springer-Verlag.

[113] Hyojoon Kim and N. Feamster. Improving network management with software defined networking. IEEE Commun. Mag., 51(2):114–119, February.
[114] Christopher Monsanto, Joshua Reich, Nate Foster, Jennifer Rexford, and David Walker. Composing software-defined networks. In Proc. 10th USENIX Symp. on Networked Systems Design and Implementation, NSDI'13, 2013.
[115] Zdravko Bozakov and Panagiotis Papadimitriou. Autoslice: automated and scalable slicing for software-defined networks. In Proc. 2012 ACM conf. on CoNEXT student workshop, CoNEXT Student '12, pages 3–4, New York, NY, USA, 2012. ACM.
[116] Connected cloud control: Openflow in mirage. http://www.openmirage.org/blog/announcing-mirage-openflow.
[117] D. Drutskoy, E. Keller, and J. Rexford. Scalable network virtualization in software-defined networks. IEEE Internet Comput., PP(99):1–1.
[118] Soudeh Ghorbani and Matthew Caesar. Walk the line: consistent network updates with bandwidth guarantees. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 67– 72, New York, NY, USA, 2012. ACM.
[119] Eric Keller, Soudeh Ghorbani, Matt Caesar, and Jennifer Rexford. Live migration of an entire network (and its hosts). In Proc. 11th ACM Workshop on Hot Topics in Networks, HotNets-XI, pages 109–114, New York, NY, USA, 2012. ACM.
[120] Ramya Raghavendra, Jorge Lobo, and Kang-Won Lee. Dynamic graph query primitives for sdn-based cloudnetwork management. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 97–102, New York, NY, USA, 2012. ACM.
[121] Xiang Wang, Zhi Liu, Yaxuan Qi, and Jun Li. Livecloud: A lucid orchestrator for cloud datacenters. In 2012 IEEE 4th Int. Conf. Cloud Computing Technology and Science (CloudCom), pages 341– 348, 2012.
[122] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In Proc. 5th int. conf. on Emerging networking experiments and technologies, pages 1–12. ACM, 2009.
[123] Xuan-Nam Nguyen, Damien Saucez, and Thierry Turletti. Efficient caching in Content-Centric Networks using OpenFlow. In 2013 IEEE Conf. Computer Commun. Workshops (INFOCOM WKSHPS), pages 67–68. IEEE, April 2013.
[124] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi, and A. Detti. Supporting information-centric functionality in software defined networks. IEEE ICC Workshop on Software Defined Netw., June 2012.
[125] N. Blefari-Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, and L. Veltri. An openflow-based testbed for information centric networking. Future Network & Mobile Summit, pages 4–6, 2012.
[126] Junho Suh, Hyogi Jung, Taekyoung Kwon, and Yanghee Choi. Cflow: Content-oriented networking over openflow. In Open Networking Summit, April 2012.
[127] D. Syrivelis, G. Parisis, D. Trossen, P. Flegkas, V. Sourlas, T. Korakis, and L. Tassiulas. Pursuing a software defined information-centric network. In Software Defined Networking (EWSDN), 2012 European Workshop on, pages 103–108, Oct.
[128] X.N. Nguyen. Software defined networking in wireless mesh network. Msc. thesis, INRIA, UNSA, August 2012.
[129] Cisco visual networking index: Global mobile data traffic forecast update, 2011–2016. Technical report, Cisco, February 2012.
[130] B. Rais, M. Mendonca, T. Turletti, and K. Obraczka. Towards truly heterogeneous internets: Bridging infrastructure-based and infrastructureless networks. In 2011 3rd Int. Conf. Commun. Syst. and Netw. (COMSNETS), pages 1–10. IEEE, 2011.
[131] A. Coyle and H. Nguyen. A frequency control algorithm for a mobile adhoc network. In Military Commun. and Information Syst. Conf. (MilCIS), Canberra, Australia, November 2010.
[132] P. Dely, A. Kassler, and N. Bayer. Openflow for wireless mesh networks. In Proc. 20th Int. Conf. on Computer Commun. and Netw. (ICCCN), pages 1–6. IEEE, 2011.