روش مدیریت توزیع جریان‌ها در سیستم‌های مبتنی بر OpenFlow

مهشید صالحی و مسعودرضا هاشمی

چکیده: معماری فعلی شبکه‌های مرکز داده ترکیبی از سوئیچ‌های اترنت و روترها است اما این معماری نمی‌تواند نیازهای این شبکه‌ها را برآورده کند. سوئیچ‌های اترنت انعطاف‌پذیر و تنظیماتشان ساده است اما گسترش‌پذیر نیستند. روترها گسترش‌پذیری بهتر و استفاده کارامد از پهنای باند را فراهم می‌کنند اما هزینه آنها زیاد است. این معماری ترکیبی سربارهای تنظیمات و نگهداری قابل توجهی را به وجود می‌آورد و بنابراین اگر بتوان شبکه‌های لایه دو را بزرگ‌تر کرد تعداد روترها و در نتیجه هزینه‌ها کمتر می‌شود. برای این منظور روش‌های زیادی ارائه شده است. در این مقاله برخی از نیازهای اصلی شبکه‌های مرکز داده و ویژگی‌های تعدادی از روش‌های پیشنهادی بیان شده و از بین روش‌ها، OpenFlow ترجیح داده می‌شود اما سربار کنترلی OpenFlow زیاد است. یک روش برای کم‌کردن سربار کنترلی جداکردن جریان‌های بزرگ و کوچک است که مدیریت جریان‌های بزرگ به کنترل‌کننده داده می‌شود و برای مسیریابی جریان‌های کوچک از مسیریابی ECMP استفاده می‌شود. OpenFlow از ECMP پشتیبانی نمی‌کند. در این مقاله روشی مبتنی بر OpenFlow برای جایگزینی ECMP ارائه شده که کارایی آن معادل ECMP است و در مقابل خطا تحمل‌پذیر است.

کلید واژه: **سوئیچ، شبکه، مرکز داده، OpenFlow، ECMP.**

۱- مقدمه

مراکز داده مدرن نیاز به یک شبکه گسترش‌پذیر با پهنای باند بالا، تأخیر کم، امنیت بالا، پشتیبانی از مجازی‌سازی و مدیریت ساده دارند. اترنت به صورت وسیع در شبکه‌های مرکز داده استفاده شده و چون عملیات سوئیچ‌های اترنت ساده و قابل انعطاف است، سوئیچ‌های اترنت به تنظیمات دستی نیاز ندارند و سوئیچ‌ها به طور اتوماتیک یاد می‌گیرند که چگونه بسته‌ها را ارسال کنند. ترمیم خرابی‌ها با پروتکل درخت پوشای سریع کاملاً اتوماتیک است و با هر توپولوژی دلخواهی می‌توانند به یکدیگر متصل شوند ولی اترنت نمی‌تواند این نیازمندی‌ها را در یک مقیاس بزرگ برآورده کند.

چهار دلیل اصلی برای گسترش‌پذیرنبودن اترنت وجود دارد: وابسته‌بودن به ارسال سیل‌آسا، استفاده از انتشار سراسری برای پروتکل‌هایی مانند ARP و DHCP، محدودبودن جدول ارسال و عدم توانایی در انجام مهندسی ترافیک.

بهترین راه حل عملی فعلی برای محدودیت‌های گسترش‌پذیری اترنت، تقسیم‌کردن شبکه به چندین شبکه اترنت مستقل از هم و استفاده از روترهای IP برای اتصال آنها است. متأسفانه این روش به طراحی دقیق نیاز دارد و روترهای IP وسایل گرانی هستند که نیاز به پیکربندی و نگهداری پیچیده‌ای دارند. اگر شبکه‌های اترنت بتوانند بزرگ‌تر شوند، تعداد روترها می‌تواند کم شود.

به دلیل محدودیت‌های گسترش‌پذیری اترنت و مشکلات روترهای IP، به طراحی یک شبکه لایه دو که بتواند یک جایگزین مناسبی برای اترنت باشد و با هاست‌های[1] موجود سازگار باشد، نیاز است. در این مقاله چندین روش که در این زمینه ارائه شده، بررسی شده است.

SDN[2] یک معماری شبکه و مبتنی بر نرم‌افزار است و به یک کنترل‌کننده اجازه می‌دهد تا رفتار سوئیچ‌های شبکه را مدیریت کند. دید سراسری و کنترلی که توسط SDN فراهم می‌شود، اجازه فراهم‌شدن امنیت خوب و کارایی بالای مورد نیاز شبکه مرکز داده را می‌دهد. در معماری SDN بخش کنترل از بخش داده جدا شده و بخش کنترل به سمت یک کنترل‌کننده مرکزی هدایت و به صورت نرم‌افزاری پیاده‌سازی می‌شود. OpenFlow [۱] پروتکل معروفی است که برای پیاده‌سازی بیشتر معماری‌های SDN استفاده شده است.

معماری‌های مبتنی بر SDN و OpenFlow می‌توانند سربار ناشی از ارسال سراسری اترنت، محدودیت‌های توپولوژی و محدودیت‌های مسیریابی را حذف کنند. استفاده از یک معماری متمرکز مانند OpenFlow باعث می‌شود مدیر شبکه به راحتی سیاست‌های مورد نظرش را به جای تک‌تک سوئیچ‌ها فقط روی کنترل‌کننده تنظیم کند. مهندسی ترافیک و کیفیت سرویس می‌تواند به خوبی انجام شود. اضافه‌کردن ویژگی‌های جدید به شبکه راحت انجام می‌شود چون ویژگی‌ها به کنترل‌کننده اضافه می‌شوند و عملیات سوئیچ‌ها سبک هستند. OpenFlow برای جلوگیری از حملات امنیتی می‌تواند از مکانیزم‌های مختلف احراز هویت استفاده کند. سوئیچ‌های OpenFlow نسبتاً ساده هستند چون سیاست‌ها به جای سخت‌افزار یا نرم‌افزار سوئیچ به کنترل‌کننده تحمیل می‌شوند. سوئیچ‌های جدید شرکت‌های معروف مانند سیسکو، HP، Juniper، Brocade و غیره، OpenFlow را پشتیبانی می‌کنند ولی هنوز یک مشکل وجود دارد و آن این که پیاده‌سازی فعلی OpenFlow برای شبکه‌های بزرگ گسترش‌پذیر نیست.

یکی از دلایل مهم گسترش‌پذیر نبودن OpenFlow سربار ناشی از نصب جریان است. وقتی به ازای یک جریان در جدول جریان مدخل نباشد، اولین بسته آن جریان به بخش کنترل سوئیچ ارسال شده و بخش کنترل سوئیچ آن را بسته‌بندی کرده و به کنترل‌کننده مرکزی ارسال می‌کند. متأسفانه پردازنده‌های اکثر سوئیچ‌ها ضعیف هستند و نمی‌توانند عملیات را به ازای هر جریان انجام دهند و پهنای باند بین بخش کنترل و بخش داده سوئیچ محدود است [۲] و [۳]. این عوامل، نرخ داده را بین سوئیچ و کنترل‌کننده مرکزی محدود می‌کند. از طرفی ارسال اولین بسته هر جریان به کنترل‌کننده و ارسال یک قانون برای آن جریان از طرف کنترل‌کننده، باعث تأخیر بیش از دو RTT[3] جریان می‌شود.

این مقاله در تاریخ ۳ دی ماه ۱۳۹۲ دریافت و در تاریخ ۱۴ تیر ماه ۱۳۹۳ بازنگری شد.

مهشید صالحی، دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان، اصفهان، (email: mahshid.salehi@ec.iut.ac.ir).

مسعودرضا هاشمی، دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان، اصفهان، (email: hashemim@cc.iut.ac.ir).

1. Host
2. Software Defined Networking
3. Round Trip Time

Archive of SID

www.SID.ir

آنالیزهای ترافیک مرکز داده [۴] و [۵] نشان می‌دهد بیشتر حجم ترافیک در تعداد کمی جریان حمل می‌شوند. نویسنده‌های این مقاله‌ها [۴] و [۵] گزارش داده‌اند که نود درصد جریان‌ها کمتر از ۱ MB داده حمل می‌کنند و بیشتر از نود درصد بایت‌ها در جریان‌های بزرگ‌تر از ۱۰۰ MB منتقل می‌شوند. در مقاله‌های [۳]، [۶] و [۷] اثبات شده که با محول‌کردن کنترل جریان‌های کوچک به خود سوئیچ و کنترل‌کردن جریان‌های بزرگ توسط کنترل‌کننده، می‌توان به مدیریت کارای جریان‌ها رسید. تکنیک‌های ارسال جریان بر اساس Hash مانند مسیریابی ECMP[1] [۸] و [۹] فقط برای تعداد زیادی جریان‌های کوچک خوب کار می‌کند و برای جریان‌های بزرگ (جریان‌هایی که مقدار قابل توجهی داده منتقل می‌کنند) خوب کار نمی‌کند. Hedera [۶] نشان داد که ارسال جریان‌های کوچک توسط مسیریابی ECMP و مدیریت جریان‌های بزرگ توسط کنترل‌کننده می‌تواند ۱۱۳ درصد گذردهی[2] بیشتری نسبت به ECMP به دست آورد. این مطلب در مقاله [۷] بررسی و تأیید شده است.

اما پیاده‌سازی فعلی OpenFlow از ECMP پشتیبانی نمی‌کند و به کنترل‌کننده هم اجازه دسترسی به جدول ECMP را نمی‌دهد. در این مقاله روشی مبتنی بر OpenFlow برای جایگزینی ECMP ارائه شده است که گذردهی آن معادل با ECMP است.

در بخش ۲ نیازهای شبکه‌های مرکز داده بیان می‌شود و سپس در بخش ۳ بر اساس این نیازها بعضی از روش‌های ارائه‌شده برای ایجاد یک شبکه لایه ۲ اترنت بررسی می‌شود. در بخش ۴ روش پیشنهادی معادل ECMP بیان شده و در بخش ۵ بر اساس شبیه‌سازی انجام‌شده، این روش ارزیابی می‌شود. نهایتاً در بخش ۶ نتیجه‌گیری بیان می‌شود.

۲- نیازهای شبکه‌های مرکز داده

همان طور که در قسمت‌های قبل ذکر شد، اترنت در محیط‌های مرکز داده حضور دارد، نیاز به تنظیمات دستی ندارد، با همه توپولوژی‌ها سازگار است اما گسترش‌پذیر نیست و پهنای باند در دسترس را هدر می‌دهد. شبکه‌های ترکیبی لایه دو و سه می‌توانند گسترش‌پذیری و پهنای باند بهتری را فراهم کنند ولی مدیریت و پیکربندی آنها مشکل است. اپراتورهای شبکه مزیت‌های هر دو روش را می‌خواهند در حالی که ترجیح می‌دهند برای کاهش هزینه از سخت‌افزار موجود استفاده شود. بنابراین چالش اصلی فراهم‌کردن سادگی اترنت و مقیاس‌پذیری و کارایی IP در حالی که از سخت‌افزار موجود استفاده می‌شود، است.

چند مورد از نیازهای شبکه‌های مرکز داده که باید برای طراحی معماری این شبکه‌ها به آنها توجه شود عبارتند از:

۱) گسترش‌پذیری: اصلی‌ترین معیارهای گسترش‌پذیری عبارتند از:
- حذف ارسال سیل‌آسا
- سربار کنترلی کم
- نیاز به اندازه جدول ارسال کم

۲) استفاده کارامد از پهنای باند: در معماری ارائه‌شده برای یک شبکه مرکز داده باید همه لینک‌ها فعال باشند و توازن بار انجام گیرد.

۳) بدون حلقه بودن: برای ترافیک‌ها حلقه ایجاد نشود.

۴) نیازنداشتن به تنظیمات دستی: برای ارسال ترافیک نیاز به تنظیمات دستی جداول ارسال نباشد، چون در مقیاس‌های بزرگ تنظیمات دستی سخت است و مدیریت را سخت می‌کند.

۵) هزینه کم: ما دو معیار زیر را برای هزینه کم در نظر گرفتیم:
- معماری نباید سخت‌افزار سوئیچ را تغییر دهد و باید بتواند با سخت‌افزار موجود کار کند.
- معماری نباید نرم‌افزار هاست را تغییر دهد.

۶) مستقل‌بودن از توپولوژی: معماری باید با هر توپولوژی دلخواهی کار کند و وابسته به یک یا چند نوع توپولوژی خاص نباشد. وابسته‌بودن معماری به یک توپولوژی باعث می‌شود اپراتورهای شبکه نتوانند توپولوژی جدیدی که کارایی بهتر، هزینه کمتر و یا هر دو را ارائه می‌دهد را جایگزین توپولوژی فعلی شبکه کنند.

۷) جابه‌جایی هاست[3]: هاست‌ها باید بتوانند بدون قطع‌شدن ارتباطاتشان در شبکه جابه‌جا شوند.

۸) زمان Failover کم: وقتی یک خطایی در شبکه رخ می‌دهد، مدت زمانی که طول می‌کشد تا تغییرات در شبکه اعمال شود، باید کم باشد.

۳- مقایسه روش‌های ارائه‌شده

این روش‌ها را به سه گروه تقسیم کردیم: روش‌های توزیع‌شده، روش‌های مبتنی بر کنترل‌کننده مرکزی و روش‌هایی که توپولوژی را تعریف می‌کنند.

۳-۱ روش‌های توزیع‌شده

TRILL [۱۰] و [۱۱] از پروتکل IS-IS که یک مکانیزم مبتنی بر ارسال سیل‌آسا است برای شناسایی توپولوژی استفاده کرده و متحمل سربار کنترلی ناشی از اجرای این پروتکل می‌شود. بنابراین زمان همگرایی و زمان Failover این روش به پروتکل IS-IS وابسته است. این روش به فریم‌ها[4] یک فرمت هدر جدید اضافه کرده و از یک فیلد TTL برای جلوگیری از حلقه استفاده می‌کند و بنابراین نیاز به تغییر سخت‌افزار سوئیچ دارد. در صورت وجود ناسازگاری در اطلاعاتی که سوئیچ‌ها از توپولوژی دارند، حلقه‌های موقت می‌توانند ایجاد شوند ولی این شرایط خیلی زودگذر بوده و هیچ وقت حلقه‌های بی‌نهایت ایجاد نمی‌شود. همچنین Rbridgeها آدرس‌های فیزیکی را از بسته‌هایی که از آنها عبور می‌کند، یاد می‌گیرند. به دلیل این که سوئیچ‌ها توپولوژی را می‌شناسند به درخواست‌های ARP به صورت محلی پاسخ داده می‌شود ولی درخواست‌های پروتکل DHCP همچنان ارسال سیل‌آسا می‌شوند. چون این روش انتشار سراسری را حذف نمی‌کند هنوز مسایل امنیتی مربوط به انتشار سراسری برای آن مطرح است. در این روش حداکثر تعداد مدخل‌های سوئیچ‌ها برابر تعداد هاست‌ها است. این روش مسیریابی را محدود به کوتاه‌ترین مسیرها می‌کند و اگر چند مسیر کوتاه وجود داشت، برای انتخاب مسیر از مسیریابی ECMP استفاده می‌کند.

FabricPath [۱۲] ورژنی از TRILL است ولی سریع‌تر و عملیات آن ساده‌تر است. روش TRILL، ۱۹۲ بیت ولی FabricPath، ۱۲۸ بیت هدر به فریم اصلی اضافه می‌کند. کارایی این روش نسبت به TRILL بیشتر است و از پهنای باند در دسترس به طور مؤثر استفاده می‌کند. در این روش آدرس‌های فیزیکی فقط در سوئیچ‌های لبه[5] یاد گرفته می‌شوند و سوئیچ‌های هسته آدرس‌های فیزیکی را یاد نمی‌گیرند. سوئیچ‌های لبه

1. Equal Cost Multi Path
2. Aggregate Throughput
3. Host Mobility
4. Frame
5. Edge

Archive of SID

www.SID.ir

آدرس فیزیکی هاست‌های محلی و هاست‌هایی که ارتباط دوطرفه با یکی از هاست‌های محلی را دارند، یاد می‌گیرند. بنابراین حداکثر تعداد مدخل‌های سوئیچ‌های لبه برابر تعداد هاست‌ها و حداکثر تعداد مدخل‌های بقیه سوئیچ‌ها برابر تعداد سوئیچ‌ها است. همچنین جابه‌جایی هاست‌ها به راحتی قابل انجام است چون نیاز نیست مدخل‌های سوئیچ‌های هسته تغییر کند. این روش مانند TRILL یک فرمت هدر جدید معرفی می‌کند و بنابراین نیاز به تغییر سخت‌افزار سوئیچ دارد.

SEATTLE [۱۳] مشابه روش TRILL است. این روش ارسال سیل‌آسا را برای درخواست‌های ARP و DHCP به ترتیب با استفاده از جدول Hashing توزیع‌شده و سرور DHCP حذف می‌کند ولی همچنان برای اجرای پروتکل مسیریابی به ارسال سیل‌آسا نیاز دارد و توازن بار را به کوتاه‌ترین مسیرها محدود می‌کند. حداکثر تعداد مدخل‌های سوئیچ‌ها برابر تعداد هاست‌ها است. اگر به هر دلیلی مدخل مربوط به نگاشت آدرس IP به آدرس فیزیکی هاستی از بین برود، چون بسته‌های ARP انتشار سراسری نمی‌شوند، تا زمانی که تغییرات اعمال شود هاست در دسترس نخواهد بود ولی در صورت بروز خطا فرایند اعمال تغییرات حدوداً چهار ثانیه طول می‌کشد. این روش چون انتشار سراسری بسته‌های ARP را حذف کرده، از این نظر امنیت بهتری نسبت به اترنت دارد.

روش MOOSE [۱۴] مستقل از توپولوژی است و از یک سرویس دایرکتوری مرکزی برای رسیدگی به درخواست‌های ARP و DHCP استفاده می‌کند و بنابراین سربار کنترلی آن کمتر از اترنت است. اما هنوز برای انجام عملیاتش به ارسال سیل‌آسا و انتشار سراسری وابسته است و مسایل امنیتی مربوط به انتشار سراسری در مورد آن صادق است. حداکثر تعداد مدخل‌های یک سوئیچ در این روش برابر تعداد سوئیچ‌ها است و سوئیچ‌ها می‌توانند یک پروتکل مسیریابی مانند انواع پروتکل OSPF را اجرا کنند. پروتکل OSPF-OMP مناسب است چون می‌تواند ECMP را اجرا کند اما برای مهندسی ترافیک کاری انجام نمی‌دهد. جابه‌جایی هاست‌ها در این روش قابل انجام است و وقتی یک هاست جابه‌جا می‌شود از یک بسته ARP برای اطلاع‌دادن به بقیه سوئیچ‌ها استفاده می‌شود و در نتیجه باعث ترافیک انتشار سراسری اضافی می‌شود.

VL۲ [۴] مبتنی بر توپولوژی Clos [۱۵] است. این روش ارسال سیل‌آسا را برای یادگیری آدرس فیزیکی و درخواست‌های ARP با استفاده از یک سرویس دایرکتوری و ارسال سیل‌آسای درخواست‌های DHCP را با سرور DHCP حذف می‌کند. اما در این روش یک پروتکل مسیریابی Link State اجرا می‌شود که برای اجرای آن به ارسال سیل‌آسا نیاز است و متحمل سربار کنترلی ناشی از اجرای این پروتکل می‌شود. زمان Failover این روش به پروتکل Link State وابسته است. سوئیچ‌ها نیازی به یادگیری آدرس‌های فیزیکی ندارند. حداکثر تعداد مدخل‌های جدول ارسال برای سوئیچ‌های لبه برابر تعداد هاست‌ها و برای بقیه سوئیچ‌ها برابر تعداد سوئیچ‌ها است. این روش مهندسی ترافیک را با استفاده از ECMP و VLB [۱۶] و [۱۷] انجام می‌دهد. هدف هر دو مکانیزم یکسان است. VLB ترافیک را در بین نودهای میانی و ECMP در بین مسیرها با هزینه یکسان توزیع می‌کند. VL۲ جریان‌ها را به جای بسته‌ها توزیع می‌کند و این باعث حفظ ترتیب بسته‌ها می‌شود. کارایی VLB به ازای جریان‌ها تقریباً معادل ECMP است.

۳-۲ روش‌های مبتنی بر کنترل‌کننده مرکزی

ما این دسته از روش‌ها را به دو گروه تقسیم کردیم: آنهایی که از مسیریابی Hop-by-Hop و آنهایی که از مسیریابی مبدأ استفاده می‌کنند.

۳-۲-۱ روش‌های مبتنی بر مسیریابی Hop-by-Hop

سوئیچ OpenFlow [۱] مبتنی بر معماری SDN است. در این سوئیچ وقتی اولین بسته یک جریان جدید وارد سوئیچ می‌شود، سوئیچ آن بسته را به کنترل‌کننده ارسال می‌کند. کنترل‌کننده با استفاده از Pull آمار بقیه جریان‌ها را به دست می‌آورد و یک مسیر مناسب را برای جریان جدید پیدا و روی سوئیچ‌ها نصب می‌کند. در این روش ارسال سیل‌آسای ناشی از ARP و DHCP حذف می‌شود. سوئیچ‌ها یادگیری آدرس‌های فیزیکی را انجام نمی‌دهند. سربار کنترلی این روش به دلیل جمع‌آوری آمار و نصب مسیرها به ازای هر جریان، زیاد و تعداد مدخل‌های یک سوئیچ برابر تعداد جریان‌ها است. مهندسی ترافیک و توازن بار در این روش می‌تواند به صورت بهینه انجام شود. این روش توسط سازنده‌های معروف مثل HP، سیسکو، Juniper، Brocade و ... روی سوئیچ‌ها پیاده‌سازی شده است.

Hedera [۶] روشی است که طراحان آن را برای توپولوژی درختی با چند ریشه طراحی و آزمایش کرده‌اند و برای این توپولوژی، مهندسی ترافیک نزدیک به بهینه را ارائه داده‌اند [۱۸]. در این روش ارسال سیل‌آسای ناشی از درخواست‌های ARP و DHCP با استفاده از کنترل‌کننده مرکزی حذف می‌شود و هنگامی که فریم‌ها از سوئیچ‌ها عبور می‌کنند یادگیری آدرس‌های فیزیکی انجام نمی‌شود. کنترل‌کننده مرکزی آمار جریان‌ها را با استفاده از Pullکردن سوئیچ‌ها، استخراج می‌کند و بنابراین سربار کنترلی این روش به دلیل جمع‌آوری آمار، بالا است. Hedera اندازه جدول را درست نمی‌کند و حداکثر تعداد مدخل‌های یک سوئیچ برابر تعداد جریان‌ها است.

Mahout [۷] مشابه روش Hedera است اما در Mahout شناسایی جریان‌های بزرگ توسط هاست‌ها انجام می‌شود و بنابراین نیاز به تغییر نرم‌افزار هاست‌ها دارد که باعث شده Mahout با وسایل مهم شبکه ناسازگار باشد اما سربار کنترلی آن خیلی کمتر از Pull (استفاده‌شده در Hedera) و نمونه‌گیری (استفاده‌شده در DevoFlow) است.

ایده روش DevoFlow [۳] هم مشابه روش Hedera است ولی برای توپولوژی خاصی طراحی نشده است. سربار کنترلی جمع‌آوری آمار این روش خیلی کمتر از Hedera و یک مقدار بیشتر از Mahout است. این روش برای کم‌کردن سربار کنترلی، سخت‌افزار سوئیچ را تغییر می‌دهد و برای توازن بار به جای استفاده از ECMP از توزیع احتمال یا راند-رابین استفاده شده و ظرفیت لینک در نظر گرفته می‌شود که این باعث بهینه‌ترشدن توزیع ترافیک و مهندسی ترافیک می‌شود و نتیجه آن بهتر از ECMP است.

روش PAST [۱۹] برای توپولوژی خاصی طراحی نشده است. در این روش ارسال سیل‌آسای ناشی از درخواست‌های ARP و DHCP با استفاده از کنترل‌کننده مرکزی حذف می‌شود و هنگامی که فریم‌ها از سوئیچ‌ها عبور می‌کنند یادگیری آدرس‌های فیزیکی انجام نمی‌شود. سربار کنترلی این روش نسبت به Hedera، Mahout و DevoFlow کمتر است. حداکثر تعداد مدخل‌های یک سوئیچ برابر تعداد هاست‌ها است ولی مدخل‌ها در داخل جداول ارسال قرار می‌گیرند و در جدول جریان قرار نمی‌گیرند. در جدول ارسال نسبت به جدول جریان مدخل‌های بیشتری جا می‌شود. گذردهی این روش مساوی مسیریابی ECMP است ولی در این روش نیز مانند مسیریابی ECMP ترافیک لینک‌ها برای انتخاب مسیر در نظر گرفته نمی‌شود. با خراب‌شدن یک سوئیچ باید همه درخت‌ها به روز شوند و در صورت بروز خطا حدود یک دقیقه طول می‌کشد تا K ۱۰۰ درخت به روز شود.

Archive of SID

www.SID.ir

Portland [۲۰] توپولوژی را به توپولوژی درخت با چند ریشه محدود می‌کند و ارسال سیل‌آسای ناشی از ARP را حذف نمی‌کند ولی از یک مدیر مرکزی برای کاهش سربار سیل‌آسای ناشی از ARP استفاده می‌کند. چون این روش فقط از یک مدیر مرکزی استفاده می‌کند در مقابل حملاتی که به مدیر مرکزی می‌شود، آسیب‌پذیر است و رخنه‌گر[1] می‌تواند با ارسال درخواست‌های زیاد ARP، مدیر مرکزی را از کار بیندازد. این روش فقط شامل سربار کنترلی ناشی از عملیات استاندارد شبکه بوده و سربار آن کم است. چون در این روش ارسال فریم‌ها بر اساس آدرس شبه MAC است، جداول ارسال خیلی کوچک هستند. همچنین جداول ارسال بر اساس انطباق با طولانی‌ترین پیشوند روی آدرس شبه MAC مقصد، جستجو می‌شوند. حداکثر تعداد مدخل‌های جدول ارسال یک سوئیچ برابر تعداد پورت‌های آن سوئیچ است. Portland برای توازن بار از ECMP استفاده می‌کند. زمان Failover این روش بین ۶۰ تا ۸۰ میلی‌ثانیه است.

روش SPAIN [۲۱] از VLANها برای توزیع ترافیک روی چند مسیر در توپولوژی‌های دلخواه استفاده می‌کند و یک Agent روی هر هاست نصب می‌کند تا هاست‌ها توزیع ترافیک را انجام دهند. بنابراین توزیع ترافیک بهبود می‌یابد، بار ترافیکی لینک‌ها برای انتخاب مسیر مناسب در نظر گرفته نمی‌شود و سربار کنترلی افزایش می‌یابد چون هر هاست برای این که بتواند جریان‌ها را از طریق VLANهای مختلف پخش کند، نیاز دارد درخت‌ها را از کنترل‌کننده مرکزی دانلود کند. SPAIN هنوز برای انجام عملیاتش به ارسال سیل‌آسا و انتشار سراسری وابسته است. این روش تعداد مدخل‌های یک سوئیچ را افزایش می‌دهد. حداکثر تعداد مدخل‌های یک سوئیچ برابر (تعداد هاست‌ها × تعداد VLANها) است. در این روش هاستی که جابه‌جا می‌شود مسئول است هاست‌های دیگر را از مکان جدیدش باخبر کند. وقتی یک هاست جابه‌جا می‌شود یک پیغام را ارسال سراسری می‌کند که باعث می‌شود هاست‌ها و سوئیچ‌ها خودشان را به روز کنند. پس این روش از جابه‌جایی هاست‌ها پشتیبانی می‌کند ولی نیاز به ارسال سراسری اضافی دارد. SPAIN در برابر خطاها تحمل‌پذیری بالایی دارد، چندین مسیر را محاسبه می‌کند و وقتی یک مسیر قطع شد بلافاصله می‌تواند جریان‌ها را از مسیر جایگزین ارسال کند. چون این روش هنوز به انتشار سراسری و ارسال سیل‌آسا وابسته است، مسایل امنیتی وابسته به انتشار سراسری در مورد آن صادق است. همچنین اگر فقط از یک کنترل‌کننده استفاده شود، کنترل‌کننده می‌تواند یک ضعف امنیتی محسوب شود. این روش از VLAN اترنت استفاده مجدد می‌کند و بنابراین VLAN ها نمی‌توانند برای امنیت و جداکردن ترافیک همانند اترنت سنتی استفاده شوند.

روش NetLord [۲۲] توسعه‌یافته روش SPAIN است. در این روش فقط تعداد مدخل‌های جدول ارسال نسبت به SPAIN بهبود یافته است و حداکثر تعداد مدخل‌های یک سوئیچ برابر (تعداد سوئیچ‌ها × تعداد VLANها) است.

۳-۲-۲ روش‌های مبتنی بر مسیریابی مبدأ

بعضی از روش‌ها مانند Axon و SecondNet از مسیریابی مبدأ[2] استفاده می‌کنند. ابتدا بعضی از مزایا و معایب مسیریابی مبدأ را بیان می‌کنیم و سپس به بررسی روش‌های Axon و SecondNet می‌پردازیم.

مسیریابی مبدأ تعداد مدخل‌های جدول ارسال را کاهش می‌دهد و سوئیچ‌ها آدرس‌های فیزیکی را یاد نمی‌گیرند و بنابراین گسترش‌پذیری آن نسبت به مسیریابی Hop-by-Hop بهتر است. در مقابل، پهنای باند در دسترس را کاهش می‌دهد اما این سربار برای مسیرهایی با ۱ تا ۱۰۰ هاپ ناچیز است. مسیریابی Hop-by-Hop در مقابل خرابی‌ها سریع‌تر از مسیریابی مبدأ عمل می‌کند زیرا در مسیریابی Hop-by-Hop سوئیچ‌های میانی می‌توانند به صورت مستقل تصمیمات مسیریابی را انجام دهند، در حالی که در مسیریابی مبدأ باید خرابی‌ها به سوئیچ مبدأ اطلاع داده شود تا سوئیچ مبدأ از مسیر جایگزین استفاده کند و این زمان پاسخ به خرابی را در صورتی که مسیر جایگزین محاسبه شده باشد به یک RTT محدود می‌کند و در غیر این صورت زمان محاسبه مسیر جدید نیز به آن اضافه می‌شود [۱۸].

Axon [۲۳] یک سوئیچ سخت‌افزاری لایه دو جدید است که از مسیریابی مبدأ برای بهبود گسترش‌پذیری اترنت استفاده می‌کند. Axon از یک کنترل‌کننده مرکزی برای رجیسترکردن همه سرویس‌ها و نصب مسیرها استفاده می‌کند و ارسال سیل‌آسای ناشی از ARP و DHCP و یادگیری آدرس‌های فیزیکی را حذف می‌کند. در این روش سوئیچ‌های میانی Stateless هستند و مدخل‌ها در سوئیچ‌های لبه نگهداری می‌شود. تعداد مدخل‌های یک سوئیچ لبه برابر تعداد جریان‌های هاست‌های متصل به سوئیچ است. در مقاله مربوط به این روش نشان داده شده که سربار کنترلی Axon کم است. اگرچه Axon مکانیزم توازن بار خاصی را ارائه نمی‌دهد ولی چون از کنترل‌کننده مرکزی برای انتخاب مسیر استفاده می‌کند، اجازه انجام مهندسی ترافیک را می‌دهد. در این روش در فرایند یادگیری توپولوژی از مکانیزم Pre-Shared Key استفاده می‌شود ولی چون فقط از یک کنترل‌کننده متمرکز استفاده می‌شود، کنترل‌کننده می‌تواند مورد حمله واقع شود و با از کار افتادن کنترل‌کننده کل شبکه از کار می‌افتد.

SecondNet [۲۴] نیز از مسیریابی مبدأ برای بهبود گسترش‌پذیری اترنت استفاده می‌کند و از یک کنترل‌کننده مرکزی برای نصب مسیرها و حذف ارسال سیل‌آسای ناشی از ARP و DHCP استفاده می‌کند. کنترل‌کننده مرکزی نیاز دارد مسیرها را کشف و نصب نماید، در نتیجه سربار کنترلی آن بیشتر از روش‌های BCube و DCell است. این روش با استفاده از MPLS قابل اجراست. در این روش همه سوئیچ‌ها Stateless هستند چون از مسیریابی مبدأ استفاده می‌کند و مدخل‌ها در سرورها قرار می‌گیرند. بنابراین SecondNet، نرم‌افزار هاست‌ها را تغییر می‌دهد. این روش توازن بار را روی بار ترافیکی فعلی شبکه انجام نمی‌دهد ولی امکان تضمین‌کردن پهنای باند[3] را فراهم می‌کند. SecondNet روی هر توپولوژی دلخواهی قابل اجرا است ولی کارایی آن به توپولوژی وابسته است مثلاً با BCube بهره‌وری بالایی به دست می‌آورد در حالی که با شبکه‌های VL۲ و Fat-Tree نمی‌تواند این بهره‌وری را به دست آورد. در این روش از یک درخت پوشا که ریشه آن کنترل‌کننده است، برای سیگنالینگ و اطلاع‌دادن خرابی‌ها استفاده می‌شود. در واقع یک پهنای باندی برای این منظور رزرو می‌شود و بنابراین خطاها یا تغییرات در شبکه به سرعت به کنترل‌کننده اطلاع داده می‌شود. برای ایجاد یک کنترل‌کننده تحمل‌پذیر در برابر خرابی‌ها نیز از تکرارکردن استفاده می‌شود.

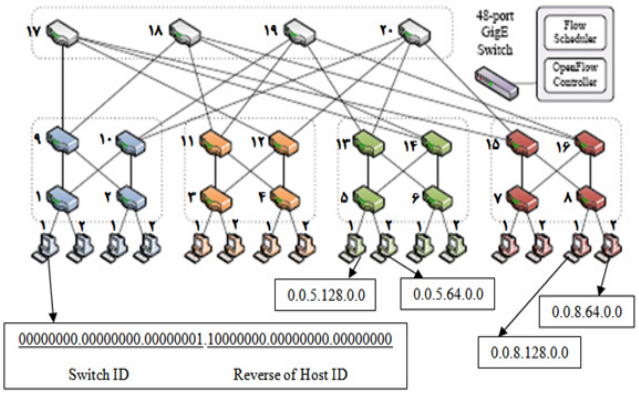
۳-۳ روش‌هایی که توپولوژی را تعریف می‌کنند

DCell [۲۵] یک توپولوژی منظم برای شبکه مرکز داده ارائه می‌دهد و عملیات ارسال فریم‌ها هم توسط سوئیچ و هم توسط سرور طبق

1. Hacker
2. Source Routing
3. Bandwidth Guarantee

Archive of SID

www.SID.ir



شکل ۱: تخصیص آدرس شبه MAC به هاست‌ها.

شناختی که از توپولوژی هست، انجام می‌شود. این روش ارسال سیل‌آسای ناشی از ARP و DHCP را حذف نمی‌کند و سربار کنترلی این روش کم است. اندازه جدول ارسال به تعداد سطوح شبکه وابسته است. DCell توزیع ترافیک را بهبود می‌دهد ولی مهندسی ترافیک را انجام نمی‌دهد و در برابر خرابی‌های لینک، سوئیچ و سرور مقاوم است اما اگر هم‌زمان چندین خرابی لینک اتفاق بیفتد ممکن است حلقه ایجاد شود.

BCube [۲۶] توسعه‌یافته DCell است. این روش از مسیریابی مبدأ برای بهبود گسترش‌پذیری استفاده می‌کند اما از کنترل‌کننده مرکزی استفاده نمی‌کند و مبدأ از طریق Probing مسیر بسته را انتخاب و در هدر بسته قرار می‌دهد. بنابراین توازن بار به خوبی انجام می‌شود و سوئیچ‌ها Stateless هستند. BCube یک توپولوژی منظم برای شبکه مرکز داده ارائه می‌دهد و عملیات ارسال فریم‌ها هم توسط سوئیچ و هم توسط سرور طبق شناختی که از توپولوژی هست، انجام می‌شود. این روش ارسال سیل‌آسای ناشی از ARP و DHCP را حذف نمی‌کند اما چون توپولوژی منظم است از ارسال سیل‌آسا برای کشف مکان هاست‌ها استفاده نمی‌کند. BCube در برابر خرابی‌های لینک، سوئیچ و سرور مقاوم است اما چون از مسیریابی مبدأ استفاده می‌کند زمان Failover آن بیشتر از روش‌های مبتنی بر مسیریابی Hop-by-Hop است.

۳-٤ خلاصه

مقایسه انجام‌شده نشان می‌دهد که راه حل ایده‌آلی برای همه ویژگی‌هایی که باید برای یک شبکه لایه دو مرکز داده وجود داشته باشد، وجود ندارد، علت این است که هر روش روی یکی از جنبه‌های یک شبکه لایه دو مرکز داده تمرکز می‌کند. ما معتقدیم که ترکیب‌کردن ویژگی‌های کلیدی بعضی از روش‌ها برای به دست آوردن روشی بهتر، امکان‌پذیر است. برای مثال اگر بتوان گسترش‌پذیری OpenFlow را بهبود داد، این روش می‌تواند در آینده برای ایجاد شبکه‌های لایه دو مرکز داده استفاده شود.

٤- روش پیشنهادی

یکی از دلایل گسترش‌پذیر نبودن روش‌های مبتنی بر SDN که در آن از OpenFlow استفاده می‌شود، درگیرشدن کنترل‌کننده به ازای هر جریان است. سوئیچ‌ها در فرایند نصب جریان‌ها خود یک مشکل گسترش‌پذیری بزرگ‌تری هستند. پهنای باند بخش داده تقریباً چهار برابر پهنای باند بخش کنترل است [۳] و این باعث تأخیر در نصب جریان‌ها می‌شود. همچنین اجازه فراهم‌شدن به موقع آمار برای انجام مدیریت ترافیک مانند توازن بار را نمی‌دهد. نگهداری دید کامل به ازای هر جریان در یک شبکه بزرگ نیاز به صدها هزار مدخل جدول جریان در هر سوئیچ دارد که سوئیچ‌ها چنین جدول جریان بزرگی ندارند.

راه حل ارائه‌شده در مقاله‌های [۳]، [۶] و [۷] این است که برای مسیریابی جریان‌های کوچک از مسیریابی‌های معمولی استفاده شود و فقط کنترل جریان‌های بزرگ به کنترل‌کننده داده شود و کنترل‌کننده فقط جریان‌هایی را که بیشترین حجم داده را انتقال می‌دهند، مدیریت می‌کند. با این روش توانسته‌اند به مدیریت کارای جریان‌ها برسند. در روش Hedera برای ارسال جریان‌های کوچک از تکنیک ECMP استفاده شده است.

ECMP [۹] تکنیکی است که امکان استفاده از چندین مسیر با هزینه یکسان (مثلاً کمترین تعداد هاب) را فراهم می‌کند. وقتی یک بسته می‌رسد و چندین مسیر با هزینه یکسان برای ارسال آن وجود دارد، روی مسیری متناظر با Hash فیلدهایی در هدر بسته به پیمانه تعداد مسیرها (معمولاً پنج فیلد آدرس IP مبدأ و مقصد، پروتکل، پورت‌های مبدأ و مقصد) ارسال می‌شود [۸]. با استفاده از این تکنیک ترافیک روی چندین مسیر توزیع می‌شود، بسته‌های یک جریان از مسیر یکسانی عبور می‌کنند و ترتیب آنها حفظ می‌شود، وقتی لینکی قطع می‌شود ترافیک بین مسیرهای باقیمانده توزیع می‌شود.

اما پیاده‌سازی فعلی OpenFlow از ECMP پشتیبانی نمی‌کند و به کنترل‌کننده هم اجازه دسترسی به جدول ECMP را نمی‌دهد.

در روش PAST [۱۹] به ازای هر هاست یک درخت پوشا که ریشه آن همان هاست است، محاسبه می‌شود و مدخل‌های مربوطه در جدول لایه ۲ سوئیچ نصب می‌شوند و برای ارسال ترافیک‌هایی که مقصدشان این هاست هست از این درخت پوشا استفاده می‌شود. گذردهی این روش معادل ECMP است اما اگر یکی از سوئیچ‌ها قطع شود باید مدخل‌های مربوط به تمام درخت‌هایی که تحت تأثیر قرار گرفته‌اند (در توپولوژی‌هایی مانند HyperX [۲۷] و JellyFish [۲۸] که به همه سوئیچ‌ها هاست متصل است، این تعداد معادل کل درخت‌ها است) به روز رسانی شوند. بنابراین همچنان مشکل گسترش‌پذیر نبودن نصب و به روز رسانی تعداد زیادی مدخل که در بالا به آن اشاره شد، وجود دارد. چون در این روش از جدول لایه ۲ (ارسال بر اساس آدرس فیزیکی و VLAN) برای ارسال ترافیک استفاده می‌شود، وقتی مدخلی متناظر با بسته در جدول لایه ۲ نباشد (مثلاً وقتی لینک یا سوئیچی در شبکه قطع می‌شود) سوئیچ طبق استاندارد اترنت بسته را ارسال سیل‌آسا می‌کند و این مانند اترنت باعث هدررفتن پهنای باند می‌شود.

در این مقاله روشی مبتنی بر OpenFlow برای جایگزینی ECMP ارائه شده که گذردهی آن معادل با ECMP است. همچنین مشکل ذکرشده برای روش PAST در این روش برطرف شده است.

٤-۱ مسیریابی روش پیشنهادی

اساس روش ما مبتنی بر آدرس شبه MAC است. کنترل‌کننده به هر سوئیچ یک شناسه می‌دهد و آدرس شبه MAC از کنار هم قرارگیری شناسه سوئیچی که هاست به آن متصل شده و معکوس شناسه هاست، تشکیل شده است. به این صورت که سه بایت اول آدرس شبه MAC شامل شناسه سوئیچ و سه بایت آخر شامل معکوس شناسه هاست است. شکل ۱ آدرس شبه MAC تخصیص داده شده به هاست‌ها را در یک توپولوژی Fat-Tree [۲۹] نشان می‌دهد.

هر هاست به‌جای آدرس MAC، آدرس شبه MAC هاست‌های دیگر را دارد. وقتی هاست یک درخواست ARP برای به دست آوردن آدرس

Archive of SID
www.SID.ir

```
n = تعداد سوئیچ‌ها;
D [n][n] // ماتریسی که طول کوتاه‌ترین مسیرها بین هر دو نود شبکه را نگه می‌دارد
All_shortest_path [n][n]// ماتریسی شامل همه‌ی پورت‌های نود مبدا که کوتاه‌ترین فاصله را تا نود مقصد دارند
for (i = 1; i < = n; i++)
   { for (j = 1; j < = n; i++)
      {D [i][j] = ∞;
if (لینکی بین i و j وجود داشته باشد)
          {D [i][j]=1;
           All_shortest_path [i][j].Add(j); }
      }
   }
for ( k = 1; k < = n; k++)
  for ( i = 1; i < = n; i++)
    for ( j = 1; j < = n; i++)
      if ( i ≠ j)
        if (D[i][j] > = D[i][k] + D[k][j])
          if (D[i][j] = = D[i][k] + D[k][j])
            foreach ( "port" in All_shortest_path[i][k])
              if ( All_shortest_path [i][j] does not contain "port" )
                All_shortest_path[i][j].Add ("port");
          else
            { All_shortest_path [i][j].Clear();
              D[i][j] = D[i][k] + D[k][j];
              foreach ( "port" in All_shortest_path [i][k])
                  All_shortest_path[i][j].Add ("port"); }
```

شکل ۲: شبه‌کد محاسبه همه کوتاه‌ترین مسیرها بین هر دو سوئیچ.

MAC یک هاست دیگر ارسال می‌کند، سوئیچ محلی (سوئیچی که مستقیماً به هاست متصل است) این درخواست را به کنترل‌کننده ارسال می‌کند. کنترل‌کننده، آدرس شبه MAC همه هاست‌ها را ذخیره و پاسخ این درخواست را ایجاد کرده و آدرس شبه MAC را در آن قرار می‌دهد. این پاسخ برای هاست درخواست‌کننده ارسال می‌شود و هاست هنگام ارسال ترافیک آدرس شبه MAC مقصد را در فیلد MAC مقصد هدر فریم‌ها قرار می‌دهد.

هر سوئیچ آدرس شبه MAC هاست‌های متصل به خود را دارد. وقتی سوئیچ فریمی را از یک هاست محلی دریافت می‌کند، آدرس شبه MAC هاست مربوطه را در فیلد MAC مبدأ هدر فریم قرار می‌دهد. اگر سوئیچ فریمی را از یک هاست غیر محلی دریافت کند، آدرس MAC اصلی هاست محلی (هاست مقصد) را در فیلد MAC مقصد هدر فریم قرار می‌دهد.

کنترل‌کننده همه کوتاه‌ترین مسیرهای بین هر دو سوئیچ را محاسبه می‌کند. ما برای محاسبه همه کوتاه‌ترین مسیرها بین هر دو سوئیچ، الگوریتم فلوید را تغییر داده‌ایم که شبه‌کد آن در شکل ۲ آمده است.

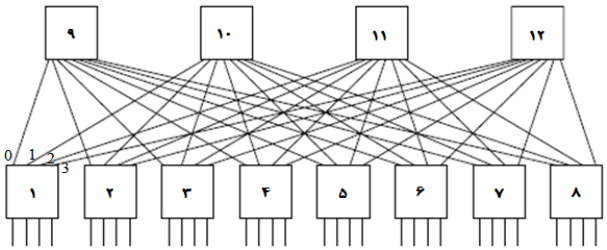
برای مسیر پشتیبان کنترل‌کننده سعی می‌کند در هر سوئیچ به ازای هر سوئیچ لبه (سوئیچ‌هایی که به آنها هاست متصل است) پورتی را انتخاب کند که فاصله آن تا سوئیچ لبه یک هاب بیشتر از کوتاه‌ترین مسیر است و در ضمن با پورت‌هایی که کوتاه‌ترین فاصله را تا مقصد دارند، مشترک نباشد. اگر چنین پورتی پیدا نشد و تعداد کوتاه‌ترین مسیرها بیشتر از یک بود، کنترل‌کننده از بین پورت‌هایی که کوتاه‌ترین فاصله را تا مقصد دارند، یکی را به صورت تصادفی به عنوان مسیر (یا پورت) پشتیبان انتخاب می‌کند.

۴-۲ توزیع ترافیک

کنترل‌کننده باید ترافیک ارسالی به هاست‌های متصل به یک سوئیچ را بین کوتاه‌ترین مسیرها به آن سوئیچ، تقسیم کند، در حالی که فقط مجاز است از ویژگی‌های در دسترس OpenFlow استفاده کند. ما قوانین ارسال را بر اساس آدرس شبه MAC مقصد ایجاد می‌کنیم و این باعث می‌شود همه بسته‌های یک جریان از یک مسیر عبور کنند و ترتیبشان حفظ شود. هدف این است تا آنجا که می‌شود تعداد قوانین را کم کنیم.

```
1. for ( i = 1 ; i < = تعداد سوئیچ‌ها ; i++)
2. for ( j = 1 ; j < = تعداد سوئیچ‌ها ; i++)
3.    { N = تعداد پورت‌های i که کوتاهترین فاصله را تا j دارند;
4.      Power = ⌈log2 N⌉
5.      if ( 2 Power == N)
6.           for ( k = 0 ; k < N ; k++)
7.             { dst_PMAC = معکوس ( بیت "Power" با k معادل باینری عدد );
8.               Port = انتخاب تصادفی از بین پورت‌های i که کوتاهترین فاصله را تا j دارند;
9.               Rules.Add(dst_PMAC , Port) // این قانون به قوانین سوئیچ i اضافه می‌شود
10.            }
11.     else
12.        {for ( k = 0 ; k < 2 Power -1 ; k++)
13.          { dst_PMAC = معکوس ( بیت "Power – 1" با k معادل باینری عدد );
14.            Port = انتخاب تصادفی از بین پورت‌های i که کوتاهترین فاصله را تا j دارند;
15.            Rules.Add(dst_PMAC , Port) // این قانون به قوانین سوئیچ i اضافه می‌شود
16.          }
17.        for ( k = 2 Power -1 ; k < N ; k++)
18.          { dst_PMAC = معکوس ( بیت "Power" با k معادل باینری عدد );
19.            Port = انتخاب تصادفی از بین پورت‌های i که کوتاهترین فاصله را تا j دارند;
20.            Rules.Add(dst_PMAC , Port) // این قانون به قوانین سوئیچ i اضافه می‌شود
21.          } } }
```

شکل ۳: شبه‌کد ایجاد قوانین مربوط به توزیع ترافیک.



شکل ۴: توپولوژی Clos.

کنترل‌کننده برای هر سوئیچ، قوانین لازم برای تقسیم ترافیک هر سوئیچ لبه بین کوتاه‌ترین مسیرها را ایجاد کرده و در سوئیچ مربوطه نصب می‌کند.

در شکل ۳ الگوریتم مربوط به ایجاد قوانین نشان داده شده است. این الگوریتم ابتدا عددی را پیدا می‌کند که دو به توان این عدد بزرگ‌تر یا مساوی تعداد کوتاه‌ترین مسیرها باشد (خط چهارم الگوریتم). اگر تعداد کوتاه‌ترین مسیرها توانی از دو بود، الگوریتم هر یک از اعداد صفر تا تعداد کوتاه‌ترین مسیرها را به عدد باینری تبدیل می‌کند که تعداد بیت‌های این عدد باینری برابر عددی است که الگوریتم در خط چهارم به دست آورده است. سپس عدد باینری به دست آمده را معکوس می‌کند (خط هفتم). الگوریتم به ازای هر یک از این اعداد باینری یکی از کوتاه‌ترین مسیرها را به طور تصادفی انتخاب می‌کند و قانون را ایجاد می‌کند (خطوط شش تا نه). اگر تعداد کوتاه‌ترین مسیرها توانی از دو نباشد، الگوریتم اعداد صفر تا دو به توان عدد به دست آمده در خط چهارم منهای یک را به باینری تبدیل می‌کند که تعداد بیت‌های این عدد باینری برابر عدد به دست آمده در خط چهارم منهای یک است. هر یک از اعداد باینری معکوس می‌شوند. به ازای هر یک از این اعداد باینری یکی از کوتاه‌ترین مسیرها به طور تصادفی انتخاب می‌شود و قانون ایجاد می‌شود (خطوط دوازده تا پانزده). در خطوط هفده تا بیست به ازای اعداد دو به توان عدد خط چهارم منهای یک تا تعداد کوتاه‌ترین مسیرها یک عدد باینری با تعداد بیت برابر عدد به دست آمده در خط چهارم حاصل شده و سپس عدد باینری به دست آمده معکوس می‌شود. به ازای هر یک از این اعداد باینری یکی از کوتاه‌ترین مسیرها به طور تصادفی انتخاب و قانون ایجاد می‌شود.

در شکل ۴ قوانین مربوط به سوئیچ یک توپولوژی Clos [۱۵] شکل ۵ نشان داده شده است. در این توپولوژی بین هر دو سوئیچ لبه چهار مسیر کوتاه یکسان وجود دارد ولی هیچ مسیری با یک هاب بیشتر وجود ندارد.

Archive of SID
www.SID.ir

Ingress Port	MAC src	MAC dst	Eth type	...	IP src	IP dst	IP Port	TCP src port	TCP dst port	Action	
*	*	0.0.2.01*	*		*	*	*	*	*	Port 0	
*	*	0.0.2.0*	*		*	*	*	*	*	Port 1	
*	*	0.0.2.1*	*		*	*	*	*	*	Port 2	
*	*	0.0.2.*	*		*	*	*	*	*	Port 3	مسیر پشتیبان
*	*	0.0.3.01*	*		*	*	*	*	*	Port 3	
*	*	0.0.3.0*	*		*	*	*	*	*	Port 1	
*	*	0.0.3.1*	*		*	*	*	*	*	Port 0	
*	*	0.0.3.*	*		*	*	*	*	*	Port 2	مسیر پشتیبان
...											
*	*	0.0.8.01*	*	...	*	*	*	*	*	Port 2	
*	*	0.0.8.0*	*		*	*	*	*	*	Port 0	
*	*	0.0.8.1*	*		*	*	*	*	*	Port 3	
*	*	0.0.8.*	*		*	*	*	*	*	Port 1	مسیر پشتیبان

شکل ۵: قوانین مربوط به سوئیچ یک توپولوژی Clos شکل ۴.

بنابراین کنترل‌کننده از بین این چهار مسیر یکی را به طور تصادفی انتخاب کرده و آن را به عنوان مسیر پشتیبان قرار می‌دهد. پس تعداد مسیرهای باقیمانده سه می‌شود که توانی از دو نیست و بنابراین خطوط یازده تا بیست و یک الگوریتم شکل ۳ اجرا می‌شود.

همان طور که در شکل ۴ مشاهده می‌شود قوانینی که طول پیشوند آدرس MAC مقصدشان بزرگ‌تر است اولویت بیشتری دارند. وقتی بسته‌ای وارد سوئیچ می‌شود با قوانین موجود در جدول چک می‌شود، اگر چندین قانون در جدول جریان وجود داشته باشد که منطبق با بسته ورودی باشد قانونی که اولویت بیشتری دارد انتخاب می‌شود.

مسیرهای پشتیبان از کمترین اولویت برخوردار هستند. اگر لینکی قطع شود، سوئیچ بلافاصله متوجه می‌شود. قانون مربوط به مسیر پشتیبان انطباق را فقط روی شناسه سوئیچ (سه بایت اول) انجام می‌دهد. پس همه بسته‌هایی که به هاست‌های متصل به یک سوئیچ می‌روند با قانون مسیر پشتیبان مربوط به آن سوئیچ منطبق می‌شوند و در صورت قطع‌شدن لینک اصلی سوئیچ به جای ارسال بسته به کنترل‌کننده و منتظر جواب ماندن، می‌تواند بلافاصله بسته را از مسیر پشتیبان ارسال کند.

۳-٤ مسیریابی روش پیشنهادی

اکنون به بررسی ویژگی‌های روش پیشنهادی بر اساس نیازهای شبکه‌های مرکز داده که در فصل دوم توضیح داده شدند، می‌پردازیم.

۱) گسترش‌پذیری:

- حذف ارسال سیل‌آسا: روش پیشنهادی مبتنی بر OpenFlow است و همان طور که توضیح داده شد، در این روش با استفاده از کنترل‌کننده مرکزی ارسال سیل‌آسای ناشی از ARP و DHCP حذف می‌شود. همچنین دیگر سوئیچ‌ها نیازی به یادگیری آدرس‌های MAC ندارند.
- سربار کنترلی کم: در روش پیشنهادی نیازی به اجرای پروتکل‌های Link State نیست. روش پیشنهادی فقط جایگزینی برای ECMP است که از ویژگی‌های در دسترس OpenFlow و سوئیچ‌ها استفاده می‌کند و می‌توان از آن برای مسیریابی جریان‌های کوچک استفاده کرد.
- نیاز به اندازه جدول کم: در روش پیشنهادی برای کاهش تعداد مدخل‌های جدول سوئیچ‌ها از تجمیع و انطباق با طولانی‌ترین پیشوند استفاده شده است. همچنین اندازه جدول در این روش به تعداد سوئیچ‌ها وابسته است و به تعداد هاست‌ها وابسته نیست که تعداد آنها خیلی کمتر از هاست‌ها است.

۲) استفاده کارامد از پهنای باند: روش پیشنهادی از نظر استفاده از پهنای باند معادل ECMP است.

۳) بدون حلقه بودن: در این روش برای ترافیک‌ها حلقه ایجاد نمی‌شود.

۴) Self - Configuration: روش پیشنهادی برای ارسال ترافیک نیازی به تنظیمات دستی ندارد.

۵) هزینه کم:

- تغییرنکردن سخت‌افزار سوئیچ: روش پیشنهادی نیازی به تغییر سخت‌افزار سوئیچ‌ها ندارد.
- تغییرنکردن نرم‌افزار هاست: در روش پیشنهادی هاست‌ها بدون تغییر باقی می‌مانند.

۶) مستقل از توپولوژی: روش پیشنهادی می‌تواند با هر توپولوژی دلخواهی کار کند.

۷) جابه‌جایی هاست: مدخل‌های جدول سوئیچ‌ها بر اساس شناسه سوئیچ هستند و بنابراین با جابه‌جایی هاست‌ها نیازی به تغییر مدخل‌های سوئیچ‌ها نیست. با جابه‌جایی یک هاست، کنترل‌کننده آدرس شبه MAC جدیدی را برای آن هاست ایجاد کرده و به همه هاست‌ها اطلاع می‌دهد تا حافظه کش ARP خودشان را به روز کنند.

۸) زمان Failover کم: اگر در یک زمان فقط پورت اصلی قطع شود سوئیچ می‌تواند فریم‌ها را بلافاصله از پورت پشتیبان ارسال کند ولی اگر هم‌زمان پورت اصلی و پورت پشتیبان خراب شوند، سوئیچ نیاز دارد منتظر تصمیم و پاسخ کنترل‌کننده بماند.

۵- ارزیابی

در این قسمت پروتکل درخت پوشا و مسیریابی ECMP و سپس گذردهی روش پیشنهادی و ECMP مقایسه می‌شوند.

۵-۱ ابزار شبیه‌سازی

ابزارهای شبیه‌سازی موجود مانند NS ۲- برای شبیه‌سازی شبکه‌هایی با مقیاس بزرگ مناسب نیستند چون این ابزارها به ازای هر بسته پردازش انجام می‌دهند و این باعث می‌شود در شبکه‌های با مقیاس بزرگ که تعداد جریان‌ها و در نتیجه تعداد بسته‌ها زیاد است، مدت زمان پردازش آنها زیاد باشد. بنابراین یک شبیه‌ساز مبتنی بر جریان با استفاده از زبان برنامه‌نویسی سی‌شارپ برای پیاده‌سازی استفاده شد اما با هر زبان برنامه‌نویسی می‌توان چنین شبیه‌سازی را پیاده‌سازی کرد.

این شبیه‌ساز توپولوژی شبکه مرکز داده را به عنوان یک گراف جهت‌دار که هر لینک آن یک ظرفیت ثابت دارد، مدل می‌کند. شبیه‌ساز همچنین الگوی ترافیکی میان هاست‌ها را به عنوان ورودی دریافت کرده و از آن استفاده می‌کند و به جای بسته‌ها جریان‌ها را مدل می‌کند و جزئیات TCP و بافرینگ سوئیچ را حذف می‌کند. شبیه‌ساز ایجادشده مبتنی بر رویداد[1] است. وقتی جریانی شروع یا خاتمه پیدا می‌کند، نرخ همه جریان‌ها با استفاده از الگوریتم شکل ۶ مجدداً محاسبه می‌شود. الگوریتم شکل ۶ اولین بار توسط مقاله [۳] ارائه و استفاده شد و سپس توسط مقاله [۱۹] نیز مورد استفاده قرار گرفت. این الگوریتم به این صورت کار می‌کند که ابتدا لینکی که بیشترین تعداد جریان از آن عبور می‌کند را پیدا کرده و به جریان‌های عبوری از آن یک نرخ اختصاص می‌دهد. سپس لینک بعدی که بیشترین تعداد جریان را دارد، انتخاب می‌شود و این روند ادامه پیدا می‌کند تا نرخ همه جریان‌ها مشخص شود.

1. Event - Based

Archive of SID

www.SID.ir

Input: set of flows F and a set of ports $\mathcal{P}$
Output: a rate $r(f)$ of each flow $f \in F$

begin
Initialize: $F_a = \emptyset; \forall f, r(f) = 0$
while $\mathcal{P} \neq \emptyset$ **do**
  Sort $\mathcal{P}$ in descending order of the number of unassigned flows per port
  $P = \mathcal{P}[0]$
  used $= \sum_{f \in F_a \cap P} r(f)$
  for each $f \in P$ AND $f \notin F_a$ **do**
    $r(f) = (P.\text{rate} - \text{used})/|P - (P \cap F_a)|$
    $F_a = F_a \cup \{f\}$
end

شکل ۶: الگوریتم تخصیص نرخ به جریان‌ها.

۵-۲ الگوهای ترافیکی

دو نوع الگوی ترافیکی برای شبیه‌سازی در نظر گرفته شد. ۱) ترافیک MapReduce [۳۰]: این الگوی ترافیکی با انتخاب n سرور به عنوان اجزای Shuffle مدل می‌شود. هر یک از این سرورها ۱۲۸ MB را به سرورهای دیگر ارسال می‌کند و هر سرور هم‌زمان با k سرور دیگر در ارتباط است. ترتیب ارتباط هر سرور با سرورهای دیگر به طور تصادفی است. همه اندازه‌گیری‌های ما برای این الگوی ترافیکی برای یک دوره یک دقیقه‌ای بوده است. ۲) الگوی ترافیکی دوم بر پایه آنالیزهای ترافیک شبکه مرکز داده است که در مقاله [۵] انجام گرفته است. این الگوی ترافیکی

بر اساس اندازه جریان‌ها، میانگین زمان رسیدن آنها و میانگین تعداد جریان‌های هر سرور که در مقاله بیان شده، ایجاد شده است. زمان اتمام این شبیه‌سازی وقتی است که همه جریان‌ها تمام بایت‌های خود را ارسال کنند.

۵-۳ توپولوژی‌های استفاده‌شده

برای ارزیابی از سه نوع توپولوژی مختلف استفاده کرده‌ایم: EGFT [۳۱]، HyperX [۲۷] و JellyFish [۲۸].

توپولوژی EGFT توسعه‌یافته Fat-Tree است و مدت طولانی است که در شبکه‌ها از آن استفاده می‌شود. این توپولوژی در سال‌های اخیر برای استفاده در شبکه‌های مرکز داده لایه دو پیشنهاد شده [۲۰]، [۲۹] و [۳۲] که در این توپولوژی، درختی با چند ریشه ایجاد می‌شود. h تعداد سطوح این درخت را نشان می‌دهد و در سطح iام تعداد فرزندان هر نود با m_i، تعداد والد هر نود با w_i و پهنای باند لینک‌های بین سطوح i و $i+1$ با k_i نشان داده می‌شود.

توپولوژی HyperX عمومیت کمتری نسبت به EGFT دارد. این توپولوژی هم سال‌های اخیر وارد مباحث شبکه مرکز داده شده است [۳۳]. در یک توپولوژی HyperX به هر سوئیچ T تا ترمینال متصل شده و توپولوژی L بعد دارد و S_k تعداد نودها در بعد k را نشان می‌دهد. هر سوئیچ به $\sum_{i=1}^{L}(S_i - 1)$ سوئیچ دیگر متصل بوده و تعداد کل سوئیچ‌ها برابر $\prod_{i=1}^{L} S_i$ است. $K \equiv K_1,\ldots,K_L$ پهنای باند لینک‌های هر بعد را نشان می‌دهد [۲۷].

اخیراً توپولوژی JellyFish [۲۸] برای شبکه‌های مرکز داده ارائه شده است. در این توپولوژی هر سوئیچ i، k_i پورت دارد که r_i تا از آن به سوئیچ‌های دیگر و بقیه پورت‌ها به سرورها متصل می‌شود. در ساده‌ترین حالت هر سوئیچ k پورت دارد که r تا از آن به سوئیچ‌های دیگر متصل شده و با N سوئیچ، $N(k-r)$ سرور پشتیبانی می‌شود. b نشان‌دهنده پهنای باند لینک‌ها است و برای ایجاد لینک‌ها، به طور تصادفی ۲ سوئیچ که پورت‌های آزاد دارند و همسایه نیستند، انتخاب می‌شود و با یک لینک به یکدیگر متصل می‌شوند. این کار ادامه پیدا می‌کند تا دیگر نتوان لینکی اضافه کرد. اگر در آخر فقط یک سوئیچ با دو پورت آزاد باقی بماند، دو سوئیچ دیگر که با یکدیگر همسایه هستند ولی با این سوئیچ همسایه نیستند، به طور تصادفی انتخاب شده و لینک بین آنها قطع شده و به پورت‌های آزاد این سوئیچ متصل می‌شوند. یک ویژگی این توپولوژی انعطاف‌پذیری آن است و چون اتصال بین سوئیچ‌ها به صورت تصادفی است به راحتی می‌توان به این توپولوژی سوئیچ اضافه کرد.

برای همه توپولوژی‌ها فرض شد که تعداد پورت‌های هر سوئیچ ۶۴ و Oversubscribtion [۲۹] ۱:۴ است. به عبارت دیگر پهنای باند دوبخشی[1] هر توپولوژی یک‌چهارم یک شبکه با پهنای باند کامل است.

نرخ پهنای باند دوبخشی یک گراف به صورت زیر محاسبه می‌شود. گراف به دو قسمت مساوی تقسیم می‌شود و نرخ پهنای باند دوبخشی برابر است با نسبت مجموع پهنای باند لینک‌هایی که از قسمت اول به قسمت دوم شبکه می‌روند به مجموع پهنای باند هاست‌های یک قسمت شبکه [۱۹].

۵-۴ مقایسه پروتکل درخت پوشا و ECMP

مقایسه‌های انجام‌شده در این مقاله بر اساس معیار گذردهی است. برای محاسبه گذردهی مجموع نرخ ارسال سرورها در نظر گرفته شده است. ۳ توپولوژی مختلف ایجاد شده و مشخصات مربوط به هر توپولوژی به صورت زیر می‌باشد

EGFT :
$\#\text{hosts} = ۱۶۰۰$, $h = ۲$, $T = m_۱ = ۵۰$, $m_۲ = ۳۲$, $w_۰ = ۱$,
$w_۱ = ۷$, $k_۰ = ۱$ Gbps , $k_۱ = ۲$ Gbps , $\#\text{switchs} = ۹۰$

که $h+1$ تعداد سطوح، m_i تعداد فرزندان هر نود سطح iام، T تعداد هاست‌های متصل به هر نود سطح یک، w_i تعداد والدهای هر نود سطح iام و k_i پهنای باند لینک‌های بین سطوح i و $i+1$ است

HyperX :
$\#\text{hosts} = ۱۶۶۴$, $L = ۲$, $S_۱ = ۴$, $S_۲ = ۱۳$, $T = ۳۲$,
$k_۱ = ۴$ Gbps , $k_۲ = ۱$ Gbps , $\#\text{switchs} = ۵۲$

که L تعداد ابعاد، S_i تعداد نودهای بعد iام، T تعداد هاست‌های متصل به هر سوئیچ و k_i پهنای باند لینک‌های بعد iام است

JellyFish :
$\#\text{hosts} = ۱۵۳۶$, $N = \#\text{switchs} = ۴۸$,
$k = ۶۴$, $r = ۳۲$, $T = ۳۲$, $b = ۱$ Gbps

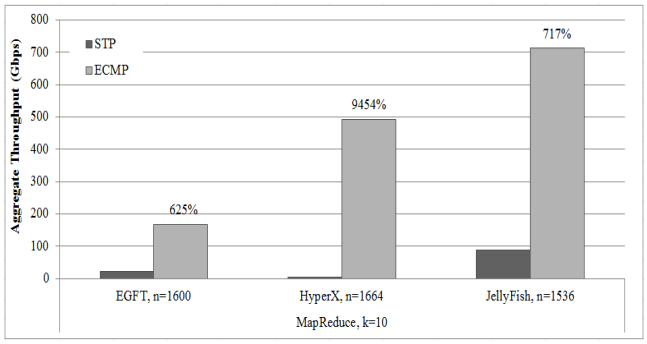
که در آن N تعداد کل سوئیچ‌ها، k تعداد پورت‌های هر سوئیچ، r تعداد سوئیچ‌های متصل به هر سوئیچ، T تعداد هاست‌های متصل به هر سوئیچ و b پهنای باند لینک‌ها است.

شکل ۷ مقایسه پروتکل درخت پوشا و مسیریابی ECMP را برای توپولوژی‌های مختلف و الگوی ترافیکی MapReduce با $k = ۱۰$ برای هر توپولوژی نشان می‌دهد. همان طور که در شکل آمده است، گذردهی با استفاده از مسیریابی ECMP و این الگوی ترافیکی برای توپولوژی EGFT ۷ برابر، برای توپولوژی HyperX ۹۶ برابر و برای توپولوژی JellyFish ۸ برابر گذردهی پروتکل درخت پوشا شده است.

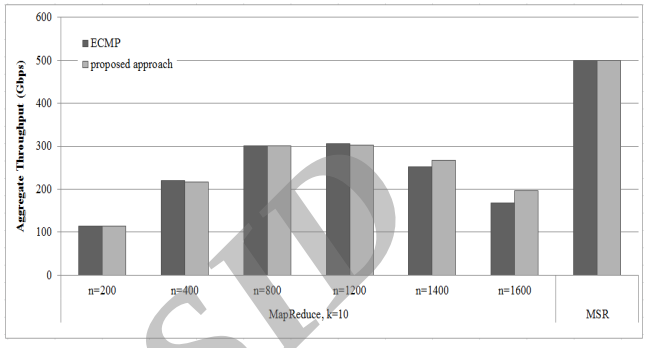
1. Bisection Bandwidth

Archive of SID

www.SID.ir



شکل ۷: مقایسه STP و ECMP.



شکل ۸: مقایسه ECMP و روش پیشنهادی برای توپولوژی EGFT.

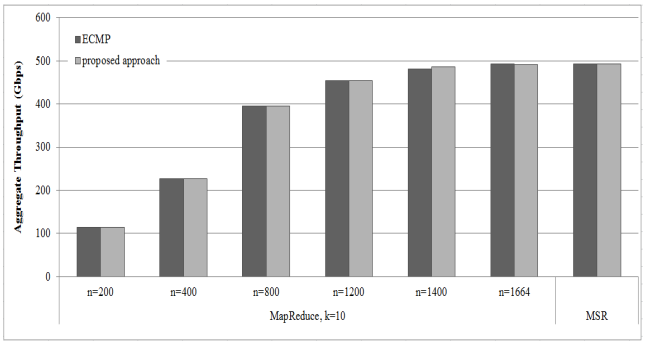
۵-۵ مقایسه روش پیشنهادی با ECMP

برای مقایسه روش پیشنهادی و مسیریابی ECMP نیز از معیار گذردهی و توپولوژی‌های مورد استفاده در مقایسه‌ی ECMP و پروتکل درخت پوشا، استفاده کردیم.

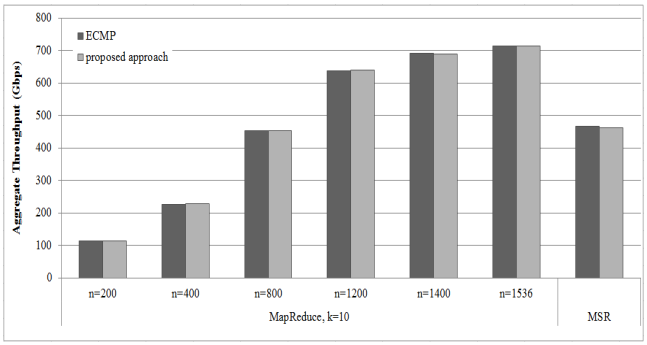
شکل ۸ مقایسه مسیریابی ECMP با روش پیشنهادی را برای توپولوژی EGFT نشان می‌دهد و MapReduce به الگوی ترافیکی اول و MSR به الگوی ترافیکی دوم اشاره می‌کند. الگوی ترافیکی اول با $k=10$ و nهای ۲۰۰، ۴۰۰، ۸۰۰، ۱۲۰۰، ۱۴۰۰ و ۱۶۰۰ شبیه‌سازی شده و هر یک از شبیه‌سازی‌ها با چهار Seed مختلف انجام شد. چون اختلاف بین نتایج حاصل از Seed های مختلف کم بود، میانگین آنها محاسبه شده و در شکل نشان داده شده است. همان طور که از شکل مشخص است برای این توپولوژی گذردهی روش پیشنهادی مساوی یا بیشتر از گذردهی مسیریابی ECMP است.

کاهش گذردهی شبکه در nهای ۱۴۰۰ و ۱۶۰۰ به خاطر این است که ECMP بر اساس Hashای که از فیلدهای هدر بسته می‌گیرد مسیر را انتخاب می‌کند و ممکن است Hash هدر بسته‌های چند جریان یکسان شود و از یک مسیر عبور کنند در حالی که مسیر دیگری موجود باشد که پهنای باند خالی داشته باشد.

شکل ۹ مقایسه مسیریابی ECMP با روش پیشنهادی را برای توپولوژی HyperX نشان می‌دهد. در این شکل نیز MapReduce به الگوی ترافیکی اول و MSR به الگوی ترافیکی دوم اشاره می‌کند. الگوی ترافیکی اول با $k=10$ و nهای ۲۰۰، ۴۰۰، ۸۰۰، ۱۲۰۰، ۱۴۰۰ و ۱۶۶۴ شبیه‌سازی شد و هر شبیه‌سازی با چهار Seed مختلف انجام گردید. در شبیه‌سازی این توپولوژی هم اختلاف بین نتایج حاصل از Seed های مختلف کم بود، بنابراین میانگین آنها محاسبه و در شکل نشان داده شده است. همان طور که از شکل مشخص است برای این توپولوژی گذردهی روش پیشنهادی با مسیریابی ECMP یکسان است.



شکل ۹: مقایسه ECMP و روش پیشنهادی برای توپولوژی HyperX.



شکل ۱۰: مقایسه ECMP و روش پیشنهادی برای توپولوژی JellyFish.

شکل ۱۰ مقایسه مسیریابی ECMP با روش پیشنهادی را برای توپولوژی JellyFish نشان می‌دهد. در این شکل نیز MapReduce به الگوی ترافیکی اول و MSR به الگوی ترافیکی دوم اشاره می‌کند. الگوی ترافیکی اول با $k=10$ و nهای ۲۰۰، ۴۰۰، ۸۰۰، ۱۲۰۰، ۱۴۰۰ و ۱۵۳۶ شبیه‌سازی شد و هر شبیه‌سازی با چهار Seed مختلف انجام گردید. اختلاف بین نتایج حاصل از Seed های مختلف در این شبیه‌سازی‌ها نیز کم بود، بنابراین میانگین آنها محاسبه و در شکل نشان داده شده است. همان طور که از شکل مشخص است برای این توپولوژی گذردهی روش پیشنهادی با مسیریابی ECMP یکسان است.

۶- نتیجه‌گیری

روش‌های زیادی برای جایگزینی اترنت ارائه شده است. در این مقاله تعدادی از این روش‌ها بر اساس نیازهای شبکه‌های مرکز داده بررسی گردید و بررسی انجام‌شده نشان داد که تا کنون راه حل ایده‌آلی برای جایگزینی اترنت که همه نیازهای شبکه‌های مرکز داده را برآورده کند، ارائه نشده است. از بین این روش‌ها، روش‌های مبتنی بر OpenFlow مطلوب‌تر هستند اما OpenFlow بی‌عیب نیست. چون OpenFlow کنترل‌کننده را به ازای هر جریان درگیر می‌کند، سربار آن زیاد است.

برای کاهش سربار OpenFlow چندین روش ارائه شده که در این روش‌ها برای جریان‌های کوچک از یک مسیریابی معمولی مانند ECMP استفاده می‌شود و مسیریابی جریان‌های بزرگ توسط کنترل‌کننده انجام می‌شود. اما OpenFlow از این ECMP پشتیبانی نمی‌کند و بنابراین در این مقاله روشی ارائه شده که مبتنی بر OpenFlow بوده و گذردهی آن معادل مسیریابی ECMP است و همچنین نسبت به خطا تحمل‌پذیر است. نتایج حاصل از شبیه‌سازی نشان می‌دهد که روش پیشنهادی با ECMP برابری می‌کند.

Archive of SID

www.SID.ir

مراجع

[1] N. McKeown, et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, Apr. 2008.

[2] M. Casado, T. Koponen, D. Moon, and S. Shenker, "Rethinking packet forwarding hardware," in *Proc. 7th ACM SIGCOMM HotNets Workshop*, 6 pp., 6-7 Oct. 2008.

[3] A. R. Curtis, et al., "DevoFlow: scaling flow management for high-performance networks," *SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254-265, Aug. 2011.

[4] A. Greenberg, et al., "VL2: a scalable and flexible data center network," *SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 51-62, Oct. 2009.

[5] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proc. of the 9th ACM SIGCOMM Conf. on Internet Measurement Conf.*, pp. 202-208, Chicago, 4-6 Nov. 2009.

[6] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *Proc. of the 7th USENIX Conf. on Networked Systems Design and Implementation, NSDI'10*, p. 19, 2010.

[7] A. R. Curtis, K. Wonho, and P. Yalagandula, "Mahout: low-overhead datacenter traffic management using end-host-based elephant detection," in *Proc IEEE INFOCOM*, pp. 1629-1637, 10-15 Apr. 2011.

[8] C. E. Hopps, *Analysis of an Equal-Cost Multi-Path Algorithm*, RFC 2992, 2000.

[9] *Cisco Data Center Infrastructure 2.5 Design Guide*, 2007, Available from: www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/ccmigration_09186a008073377d.pdf.

[10] R. Perlman, "Rbridges: transparent routing," in *Proc IEEE INFOCOM*, vol. 2, pp. 1211-1218, 2004.

[11] J. Touch and R. Perlman, *Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement*, RFC 5556, 2009.

[12] Cisco, *Scaling Data Centers with FabricPath and the Cisco FabricPath Switching System*, 2010.

[13] C. Kim, M. Caesar, and J. Rexford, "Floodless in seattle: a scalable ethernet architecture for large enterprises," *SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 3-14, Feb. 2008.

[14] M. Scott, A. Moore, and J. Crowcroft, "Addressing the scalability of ethernet with MOOSE," in *Proc. of DC CAVES Workshop*, 8 pp., 17-18 Mar. 2009.

[15] C. Clos, "A study of non-blocking switching networks," *Bell System Technical J.*, vol. 32, no. 2, pp. 406-424, Mar. 1953.

[16] R. Zhang-Shen and N. McKeown, "Designing a predictable Internet backbone network," in *Proc. of 3rd Workshop on Hot Topics in Networks, HotNets-III*, pp. 58-64, Nov. 2004.

[17] M. Kodialam, T. Lakshman, and S. Sengupta, "Efficient and robust routing of highly variable traffic," in *Proc. of 3rd Workshop on Hot Topics in Networks, HotNets-III*, 15-16 Nov. 2004.

[18] B. Stephens, A. L. Cox, S. Rixner, and T. S. E. Ng, "A scalability study of enterprise network architectures," in *Proc. of the 2011 ACM/IEEE 7th Symp. on Architectures for Networking and Communications Systems*, pp. 111-121, Oct. 2011.

[19] B. Stephens, et al., "PAST: scalable ethernet for data centers," in *Proc. of the 8th Int. Conf. on Emerging Networking Experiments and Technologies*, pp. 49-60, Nice, Dec. 2012.

[20] R. N. Mysore, et al., "PortLand: a scalable fault-tolerant layer 2 data center network fabric," *SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 39-50, Aug. 2009.

[21] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS data-center ethernet for multipathing over arbitrary topologies," in *Proc. of the 7th USENIX Conf. on Networked Systems Design and Implementation, NSDI'10*, pp. 265-280, Apr. 2010.

[22] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, "NetLord: a scalable multi-tenant network architecture for virtualized datacenters," *SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 62-73, Aug. 2011.

[23] J. Shafer, B. Stephens, M. Foss, S. Rixner, and A. L. Cox, "Axon: a flexible substrate for source-routed ethernet," in *Proc. of the 6th ACM/IEEE Symp. on Architectures for Networking and Communications Systems*, 11 pp., California, Oct. 2010.

[24] C. Guo, et al., "SecondNet: a data center network virtualization architecture with bandwidth guarantees," in *Proc. of the 6th Int. Conf., ACM*, 12 pp., Philadelphia, Nov. 2010.

[25] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75-86, Oct. 2008.

[26] C. Guo, et al., "BCube: a high performance, server-centric network architecture for modular data centers," *SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63-74, Oct. 2009.

[27] A. Jung Ho, et al., "HyperX: topology, routing, and packaging of efficient large-scale networks," in *Proc. of the Conf. on High Performance Computing Networking, Storage, and Analysis*, 11 pp., 14-20 Nov. 2009. 2009.

[28] A. Singla, C. Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: networking data centers randomly," in *Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation*, pp. 17-17, 25-27 Apr. 2012.

[29] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63-74, Oct. 2008.

[30] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107-113, Jan. 2008.

[31] S. R. Ohring, M. Ibel, S. K. Das, and M. J. Kumar, "On generalized fat trees," in *Proc. IEEE 9th Int. Parallel Processing Symp.*, pp. 37-44, Apr. 1995.

[32] N. Farrington, E. Rubow, and A. Vahdat, "Data center switch architecture in the age of merchant silicon," in *Proc. IEEE 17th Symp. on High Performance Interconnects*, pp. 93-102, Aug. 2009.

[33] D. Abts and J. Kim, High Performance *Datacenter Networks: Architectures, Algorithms, and Opportunities*, Morgan & Claypool Publishers, 2011.

مهشید صالحی مدرک کارشناسی مهندسی کامپیوتر خود را در سال ۱۳۸۴ از دانشگاه آزاد اسلامی واحد نجف آباد و مدرک کارشناسی ارشد مهندسی کامپیوتر خود را در سال ۱۳۹۲ از دانشگاه صنعتی اصفهان دریافت نمود. زمینه‌های علمی مورد علاقه نام‌برده عبارتند از: شبکه‌های مرکز داده، معماری سوئیچ و روتر، مدیریت شبکه‌های کامپیوتری.

مسعودرضا هاشمی تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد مهندسی برق و کامپیوتر به‌ترتیب در سال‌های ۱۳۶۴ و ۱۳۶۶ در دانشگاه صنعتی اصفهان و در مقطع دکتری مهندسی برق و کامپیوتر در سال ۱۳۷۷ در دانشگاه تورنتو کانادا به پایان رسانده است. دکتر هاشمی یکی از اعضای مؤسس شرکت Accelight Networks در سال ۱۳۷۸ بود. این شرکت در شهر اتاوا به ثبت رسید و تا سال ۱۳۸۲ موفق به طراحی و ساخت اولین روتر با ظرفیت بالای یک ترابایت با تکنولوژی سوئیچینگ نوری شد. نام‌برده در سال ۱۳۸۲ به دانشگاه صنعتی اصفهان ملحق شد و بعد از یکسال کار تحقیقاتی در دانشگاه تورنتو طی سال ۱۳۸۳، در سال ۱۳۸۴ همکاری خود با دانشگاه صنعتی اصفهان را از سر گرفت. دکتر هاشمی در حال حاضر با مرتبه دانشیاری عضو هیأت علمی دانشکده مهندسی برق و کامپیوتر دانشگاه صنعتی اصفهان است. وی در گروه‌های سخت‌افزار، فناوری اطلاعات و گروه شبکه همکاری می‌کند. وی همزمان رئیس مرکز فناوری دانشگاه نیز است. زمینه‌های تحقیقاتی وی عبارتند از: معماری سوئیچ و روتر، شبکه‌های مبتنی بر نرم‌افزار، شبکه‌های داده محور و مدیریت مصرف در شبکه‌های هوشمند برق.

Archive of SID

www.SID.ir