دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

معماری شبکه مراکزداده

و بررسی چند نمونه از طرح‌های موجود

دانشجو:

اسماعیل امیری

پروژه دات کام

www.Prozhe.com

بهار 1394

به نام خدا

فهرست

مقدمه

امروزه مراکز داده[1] نقش مهمی را در صنعت فناوری اطلاعات بر عهده دارند. به خصوص این که گسترش معماری پردازش ابری[2] موجب شده است حجم عظیمی از پردازش و ذخیره‌سازی اطلاعات بر روی سرورهای مراکز داده انجام شود. این سرورها شامل هزاران PC شخصی بوده که با همکاری یکدیگر پردازش‌های مربوطه را انجام می‌دهند. بهره‌گیری از چنین ساختاری، مرکزداده را به صورت ماژولار در آورده؛ ضمن این که آن را از لحاظ اقتصادی مقرون به صرفه می‌سازد. با این حال، مشکل بزرگ در این مراکز ارتباط میان گره‌های محاسباتی است و نه توان محاسباتی هر کدام از آن‌ها [1]. در واقع طراحی شبکه‌ای که بتواند نیاز این گره‌ها به ارتباط با یکدیگر را برآورده سازد، همواره یک چالش مهم برای پژوهشگران بوده است.

شبکه مراکز داده ویژگی‌هایی دارد که آن را از بسیاری از شبکه‌های دیگر متمایز می‌سازد. یکی از این ویژگی‌ها ثابت بودن توپولوژی آن است. در یک شبکه مرکزداده معمولاً گره‌ها جابجا نشده و جایگاه آنها در شبکه نیز ثابت است. این موضوع به طراحان کمک می‌کند تا پروتکل‌های مسیریابی بهینه‌تری طراحی نمایند. ویژگی دیگر تعداد زیاد گره‌های انتهایی است. برای مثال در [3] یک شبکه مرکزداده معمول این گونه معرفی شده است: 100 هزار سرور که هر کدام از آن‌ها 32 ماشین مجازی داشته باشند، بیش از سه میلیون گره انتهایی را پدید می‌آورد. همچنین اگر هر 25 سرور نیاز به یک سوییچ داشته باشد، 8000 سوییچ در مجموع مورد نیاز است. این اعداد و ارقام به خوبی ابعاد منحصربه فرد یک شبکه مرکزداده را نشان می‌دهند. پروتکل‌های معمول، از جمله اترنت، برای این شبکه‌ها مناسب نیستند.

در این پروژه چند مقاله را بررسی کرده‌ایم که در آنها به چالش‌های یک شبکه مرکزداده پاسخ داده شده است. در [1] طرحی برای معماری گسترش‌پذیر در شبکه مرکزداده معرفی شده است. در این طرح جداول جریانِ پروتکل IP دستکاری شده تا هدایت ترافیک موثرتر و بهینه‌تر انجام شود. در [2] برخی از جزییات پیاده‌سازی بررسی شده است. در طرح پورت‌لند [3] نیز از آدرس‌های فیزیکیِ صوری (موسوم به PMAC) که برخلاف آدرس‌های لایه فیزیکی معمول سلسله‌مراتبی است، بهره برده شده است.

در این پروژه ابتدا معماری شبکه‌های مرکز داده را معرفی نموده ایم. پس از آن طرح‌های معرفی شده در [1] و [3] را بررسی کرده ایم. در نهایت یک جمع‌بندی کلی بر روی مطالب ارایه شده انجام داده ایم.

---

[1] datacenters

[2] Cloud computing

فصل اول

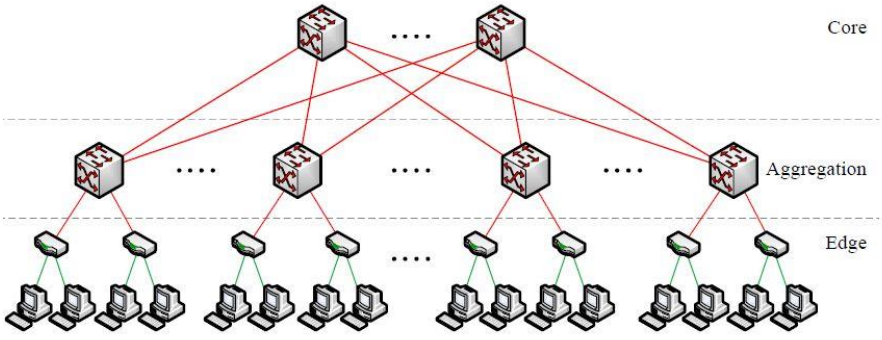
آشنایی با شبکه‌های مرکزداده

در این فصل به توصیف معماری شبکه در مراکز داده خواهیم پرداخت. این مطالب عمدتاً از [1] برداشته شده اند. با این حال تعاریف و مباحث ارایه شده در [2] و [3] را نیز مدنظر قرار داده ایم.

- توپولوژی

توپولوژی‌های معمول در مراکز داده شامل درخت های دو یا سه سطحی[1] از سوییچ‌ها و روترها است. یک طرح سه‌سطحی شامل سطح (یا لایه) هسته[2] در مرکز درخت، لایه توزیع[3] در وسط آن و لایه لبه[4] در برگ های درخت می‌باشد (ر.ک. تصویر 1). یک طرح دوسطحی لایه توزیع را نداشته و تنها لایه هسته و لبه را دارد. یک چنین طرحی معمولاً 5 تا 8 هزار میزبان را می‌تواند پشتیبانی نماید.

سوییچ های برگ درخت معمولاً تعداد قابل توجهی پورت GigE دارند (فرضاً 48 تا 288 عدد) و تعدادی پورت 10 GigE نیز برای اتصال به سوییچ های لایه بالاتر دارد. در واقع ترافیک میزبان‌ها توسط لینک های 10 GigE به سوییچ های لایه بالاتر منتقل می‌شود. در لایه های بالاتر سوییچ هایی با تعدادی پورت 10 GigE (معمولاً 32 تا 128 پورت) و ظرفیت مناسب برای جابجایی ترافیک بین قسمت های مختلف لایه دسترسی وجود دارد.



تصویر یک: معماری شبکه مرکز داده

[1] two- or three-level trees
[2] core
[3] aggregation
[4] edge

در [3] یک مرکزداده فرضی این گونه توصیف شده است: در این مرکزداده ماشین‌ها در رک[1]ها و ردیف هایی قرار می‌گیرند. این مرکز شامل 24 ردیف و 12 رک است. هر رک چهل ماشین دارد که توسط سوییچ Top of Rack (ToR) به هم متصل شده اند. این سوییچ یک پهنای‌باند دوطرفه non-blocking را فراهم می‌آورد. سوییچ های ToR در هر ردیف به یک سوییچ End of Row متصل می‌شوند. گاهی برای ایجاد قابلیت اطمینان هر سوییچ ToR به چند سوییچ EoR (مثلاً 4 تا) متصل می‌شود. یک سوییچ EoR معمولاً سوییچی 10 GigE است که برخی پورت های آن به سوییچ های هسته متصل شده است. در جدول زیر که از [2] برداشته شده است، برخی از ویژگی های سوییچ های ToR و EoR به طور خلاصه دیده می‌شود.

	TOR1G	TOR10G	EOR
GbE Ports	48	0	0
10GbE Ports	4	24	128
Power (W)	200	200	11,500
Size (RU)	1	1	33

توصیفات بالا به نوعی به آن چه در ابتدای متن گفته شد شباهت داشته و در هر صورت ساختار شبکه مراکز داده، یک شبکه درختی است که هر چه به ریشه (یا ریشه های) آن نزدیک تر می‌شویم کارایی بالاتر می‌رود و پهنای‌باند لینک‌ها نیز افزایش می‌یابد.

- هدایت بسته‌ها

دو روش کلی در مراکز داده برای مدیریت هدایت بسته‌ها وجود دارد. در روش اول از آدرس دهی لایه 3 استفاده می‌شود، در حالی که در روش دوم آدرس های لایه 2 مبنای کار است. در روش لایه 3 به تمامی میزبان‌ها آدرس های IP سلسله مراتبی بر اساس سوییچی که به آن متصل شده اند نسبت داده می‌شود. در مثال بخش قبلی، میزبان هایی که به یک سوییچ ToR وصل هستند در یک شبکه 26/ قرار می‌گیرند. (تعداد میزبان های هر رک 40 عدد است لذا شش بیت برای مشخص کردن آن‌ها لازم است.) همچنین تمامی رک‌ها در هر ردیف (که به یک سوییچ EoR متصل هستند) درون یک شبکه 22/ قرار می‌گیرند. این شیوه آدرس دهی حجم کم جداول جریان را در تمامی سوییچ‌ها تضمین می‌نماید. برای مثال سوییچ های هسته تنها لازم است که آدرس های ردیف‌ها را در خود ذخیره نمایند.

برای مسیریابی بین میزبان‌ها می‌توان از پروتکل های مسیریابی درون‌دامنه ای استفاده کرد (برای نمونه OSPF). این پروتکل‌ها معمولاً در صورت رخ دادن خطا (که یک امر متداول است) مسئله را شناسایی نموده و لینک یا سوییچ خراب را از مدار خارج می‌سازند. پدیدار شدن یک حلقه دائمی در لایه 3 تبعات منفی کم تری داشته چرا که TTL بسته‌ها از گردش دائمی آن‌ها جلوگیری می‌کند. با این حال جداول جریان دچار مشکل خواهند شد.

[1] rack

متاسفانه روش مبتنی بر آدرس دهی لایه 3 یک مشکل بزرگ دارد و آن وارد شدن فشار کاری بر روی مسوول شبکه است. (به عبارتی اتوماتیک نیست). اضافه شدن یک سوییچ به شبکه نیازمند بینش سطح بالا به شبکه و یک فرآیند پرخطا است. همچنین یک حالت ناسازگار بین اجزای شبکه (برای مثال یک DHCP که درست تنظیم نشده است) ممکن است منجر به این شود که بخش هایی از شبکه از دسترس خارج شود، با این توضیح که خطایابی نیز در این حالت سخت است. مسئله آخر این که تمایل به مجازی سازی در سرورها مطلوبیت راه حل های لایه 3 را کاهش داده است.

بر اساس آن چه گفته شد، بسیاری از مراکز داده به سوی آدرس دهی لایه 2 جذب شده اند، که در آن آدرس های تخت (و نه سلسله مراتبی) در لایه دو مبنای کار هستند. یک فابریک لایه دو مدیریت ساده تری دارد، و البته چالش های خود را نیز دارد. روش اترنت استاندارد در شبکه های با صدها هزار میزبان مقیاس پذیر نیست، چرا که لازم است همه‌پخشی در تمامی فابریک پشتیبانی شود. بدتر از همه این که وجود یک درخت پوشای[1] تکی (حتی اگر بهینه باشد)، در شبکه های با چند مسیر یکسان کارایی را به شدت تحت تاثیر قرار خواهد داد.

یک حد وسط بین دو روش ذکرشده می‌تواند به کارگیری VLAN باشد. VLAN اجازه می‌دهد یک فابریک لایه دو به بیش تر از مرز یک سوییچ گسترش یابد. VLANها نیز مشکلات خود را دارند؛ برای مثال لازم است که تنظیمات آن‌ها در تک تک سوییچ‌ها انجام شود، ضمن این که هر سوییچ باید اطلاعات کامل میزبان های درون خود را داشته باشد. مسئله آخر این که در این روش نیز یک درخت پوشا وجود دارد، و لذا کارایی محدود می‌شود.

- Oversubscription

بسیاری از مراکز داده برای کاستن از هزینه‌ها از مفهومی به نام Oversubscription بهره می‌برند، که عبارت است از نسبت *میزان پهنای‌باند تجمیع شده قابل دستیابی توسط گره های انتهایی در بدترین حالت* به *مجموع پهنای‌باند دو قسمتی*[2] *در یک توپولوژی معمول*. به زبان ساده تر oversubscription زمانی رخ می‌دهد که نرخ جریان های ورودی به سوییچ بیش تر از نرخ جریان خروجی آن باشد.

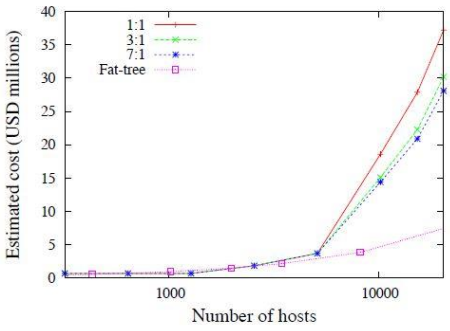
نسبت oversubscription برابر با 1:1 نشان می‌دهد که تمامی میزبان‌ها می‌توانند با هر کدام از میزبان‌ها و با تمام پهنای‌باند موجود در اینترفیس خود ارتباط برقرار سازند. از طرفی نسبت 5:1 نشان می‌دهد تنها 20 درصد از ظرفیت میزبان در اختیار است. در واقع اگر پهنای‌باند لینک متصل به سرور 1 Gbps باشد، در این حالت تنها 200 Mbps پهنای‌باند موجود است. طراحی های معمول از نسبت 2.5:1 (در شبکه یک گیگابیت‌درثانیه معادل با 400 Mbps) و یا 8:1 (برابر با 125 Mbps) بهره می‌برند.

[1] Spanning tree
[2] bisection

در تعاریف بالا از واژه دوقسمتی استفاده شده است. اگر شبکه را به دو قسمت کاملاً شبیه به هم تقسیم نماییم، مجموع ظرفیت ارتباطی میان این دو قسمت، پهنای‌باند آن bisection نام دارد. کمترین پهنای‌باند bisection برای تمامی دوقسمتی های موجود، پهنای‌باند دوقسمتی شبکه نام داشته و در تعریف oversubscription استفاده شده است.

- هزینه

هزینه هایی که برای شبکه میان کلاسترها در نظر گرفته می‌شود، بر روی طراحی تاثیر بالایی دارد. برای مثال نسبت oversubscription از جمله پارامترهای طراحی است. مقاله [1] بر اساس تجربه های معمول یک محاسبه سرانگشتی در زمینه هزینه های شبکه‌بندی مرکزداده انجام داده است. توجه شود که این ارقام مربوط به زمان چاپ مقاله (2008) می‌باشد. اگر فرض شود که یک سوییچ 48 پورت GigE 7000که در لبه شبکه استفاده می‌شود 7000 دلار و یک سوییچ 128 پورت GigE 10 که برای لایه توزیع مناسب است، 700 هزار دلار قیمت داشته باشد. تصویر 2 میزان هزینه‌ها را بر اساس دلار آمریکا و به صورت تابعی از تعداد میزبان های انتهایی نشان می‌دهد. هر کدام از منحنی‌ها مربوط به یک نرخ oversubscription است. برای مثال، شبکه ای که بتواند 200 هزار میزبان را با پهنای‌باند کامل بین تمام آن‌ها متصل سازد، حدوداً 37 میلیون دلار هزینه در بر دارد.



تصویر 2: هزینه بر حسب تعداد میزبان های مرکز داده

- مسیریابی چندمسیره

ارایه پهنای‌باند کامل بین هر دو میزبان دلخواه نیاز به درخت های با چند ریشه و با چند سوییچ هسته دارد. این معماری امکان ارتباط بین دو میزبان از طریق دو یا چند مسیر متفاوت را به ما می‌دهد. در نتیجه تکنیک های مسیریابی چند مسیره از جمله ECMP مورد نیاز است. بدون استفاده از ECMP، که پروتکل غالب برای این کاربرد است، بزرگ ترین کلاستری که می‌تواند با یک ریشه و با oversubscription برابر با 1:1 پشتیبانی شود، 1280 گره انتهایی خواهد داشت (با توجه به پهنای‌باند موجود در یک سوییچ 128 پورت GigE 10)

برای بهره بردن از امکان ارتباط چندمسیره ECMP پخش بار استاتیک را میان جریان‌ها انجام می‌دهد. در این روش پهنای‌باند جریان‌ها در محاسبات در نظر گرفته نمی شود. در نتیجه ممکن است در برخی الگوهای ارتباطی معمول نیز oversubscription داشته باشیم. علاوه بر این، پیاده‌سازی های معمول ECMP تعداد مسیرها را به 8 تا 16 مسیر محدود می‌سازد، که این رقم برای ارایه پهنای‌باند دوطرفه در مراکز داده بزرگ مناسب نیست. در نهایت باید به افزایش نمایی حجم جدول جریان در صورت استفاده از ECMP اشاره کرد، که هزینه‌ها و تاخیر در هدایت بسته‌ها را به دنبال دارد.

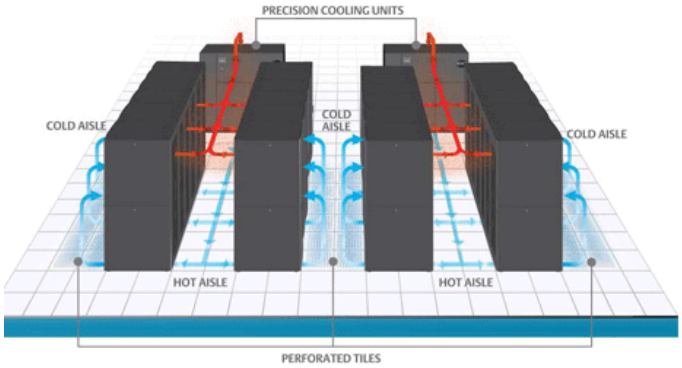
- مجازی سازی

محبوبیت روزافزون مجازی سازی در سرورهای مراکز داده الزامات خاصی را به شبکه تحمیل می‌نماید. سیستم های فعلی امکان اجرای ده‌ها ماشین مجازی بر روی یک سرور را می‌دهند. هر کدام از این ماشین های مجازی آدرس IP و MAC خود را خواهند داشت. چنین روندی در مراکز داده با صدها هزار سرور یک شیوه آدرس دهی کاملاً مقیاس پذیر را می‌طلبد، چرا که میلیون‌ها آدرس مجزا خواهیم داشت. مجازی سازی همچنین این امکان را می‌دهد که یک VM از طریق شبکه به سرور دیگری انتقال یابد. این جابجایی بنا به دلایل مختلفی ممکن است انجام شود. برای مثال اگر سروری تنها یک VM داشته باشد، آن VM را می‌توان به سرور دیگری منتقل کرده تا با خاموش نمودن سرور اولی، مصرف توان بهینه تر شود. از طرفی با پخش نمودن VMها می‌توان به توزیع بار دست یافت.

در هر دو روش آدرس دهی لایه دو و سه جابجایی VMها دردسرساز است. در لایه سه میزبان‌ها بر اساس سوییچی که به آن متصل هستند آدرس دریافت می‌کنند. در صورت جابجایی یک VM، آدرس آن باید تغییر یابد که این مطلوب نیست، چرا که تمامی ارتباطات باز TCP از بین رفته و حالت ذخیره شده در شبکه نیز بی اعتبار می‌شود. از طرفی لایه دو به تغییرات IP حساس نیست. با این حال گسترش مسیریابی به میلیون‌ها آدرس حتی اگر امکان پذیر باشد، قطعاً دشوار است.

- برخی جزییات پیاده‌سازی

در انتهای این فصل می‌خواهیم برخی از جزییات پیاده‌سازی یک شبکه مرکزداده را بیان نماییم. می‌دانیم که تمام پردازش داده در رک‌ها انجام می‌شود. رک‌ها در ردیف هایی به صورت کناربه‌کنار قرار می‌گیرند. ردیف‌ها نیز به صورت جلوبه‌جلو و عقب‌به‌عقب طراحی شده تا خنک سازی بهتر انجام شود. در واقع در این روش دو راهروی گرم و سرد ایجاد می‌شود. راهروی سرد حداقل 1.22 متر عرض داشته و راهروی گرم نیز حداقل 0.9 متر عرض دارد. بیش تر سیم‌ها از راهروی گرم عبور می‌نمایند. تصویر 3 این وضعیت را به تصویر کشیده است.



تصویر 3: راهروی گرم و سرد در مرکز داده

برای سیم کشی نیز به طور کلی سه روش وجود دارد. اگر رک‌ها درون یک ردیف باشند، ساده ترین راه این است که سیم‌ها را درون همان رک‌ها عبور دهیم. اما اگر رک‌ها در ردیف های جداگانه باشد، سیم‌ها از بالای رک‌ها عبور داده شده و به سقف آویزان هستند. همچنین برخی مراکز داده از کف‌پوش های قابل برداشتن برای پوشش سطح زمین استفاده کرده و از فضای خالی زیر آن‌ها برای سیم کشی می‌توان استفاده کرد.

جدول زیر لیستی از سیم های اترنت و فرستنده/گیرنده های موجود را نشان می‌دهد.

	Cat-6	Twinax	MMF120	Units
Cable cost	0.41	11.2	23.3	$/m
Cable weight	42	141	440	g/m
Cable diameter	5.5	7	25	mm
Gigabit Ethernet				
Standard	1000BASE-T	N/A	1000BASE-SX	
Range	100	N/A	550	m
Transceiver cost	10	N/A	36	$
Transceiver power	0.42	N/A	0.5	W
10 Gigabit Ethernet				
Standard	10GBASE-T	10GBASE-CX4	10GBASE-SR	
Range	30	15	300	m
Transceiver cost	100	100	250	$
Transceiver power	6	0.1	1	W
40 Gigabit Ethernet				
Standard	N/A	N/A	40GBASE-SR4	
Range	N/A	N/A	300	m
Transceiver cost	N/A	N/A	600	$
Transceiver power	N/A	N/A	2.4	W

واژه ی سیم اترنت معمولاً به سیم UTP (Unshielded Twisted Pair) اطلاق شده که در درجات مختلف موجود است. البته تمامی این درجات با استانداردهای IEEE منطبق نیست. فیبرهای نوری نیز در حال محبوب

شدن هستند (البته در زمان چاپ مقاله؛ تاکنون احتمالاً به نقطه مطلوبی رسیده اند) سیم های چند مدی (MMF) به علت ارزان تر بودن رواج بیشتری نسبت به سیم های تک مدی (SMF) دارند، با این حال کیفیت عملکرد آن‌ها پایین تر است. در سیم های چندین طول موج مختلف همزمان در سیم حضور دارند.

فصل دوم

معماری مقیاس پذیر برای شبکه مرکزداده

در این فصل مقاله [1] را مرور خواهیم کرد که طرحی برای معماری مقیاس پذیر در شبکه مرکزداده ارایه کرده است. اهداف این کار پژوهشی، طبق آن چه که در متن مقاله نوشته شده است، موارد زیر هستند:

1- هر میزبان باید بتواند با هر کاربر میزبان دیگری درون شبکه و با تمام پهنای‌باند کارت شبکه اش ارتباط برقرار نماید. این هدف احتمالاً به این علت انتخاب شده که امروزه ارتباطات میان سرورها در مراکز داده اهمیت بسیار زیادی یافته است.

2- همان گونه که کامپیوترهای شخصی مبنای گسترش قدرت محاسباتی مراکز داده به شیوه ای اقتصادی بوده اند، امید است بتوان با سوییچ های اترنت تجاری نیز شرایط مشابهی را در زمینه شبکه فراهم آورد. در واقع هدف این است که Economics of Scale را برای شبکه مرکزداده داشته باشیم.

3- کل سیستم باید با میزبان هایی که تنها IP و اترنت را پشتیبانی می‌نمایند سازگار باشد. تنها در این صورت امکان پیاده‌سازی طرح پیشنهادی در مراکز داده بهره‌برداری شده و در حال کار وجود دارد.

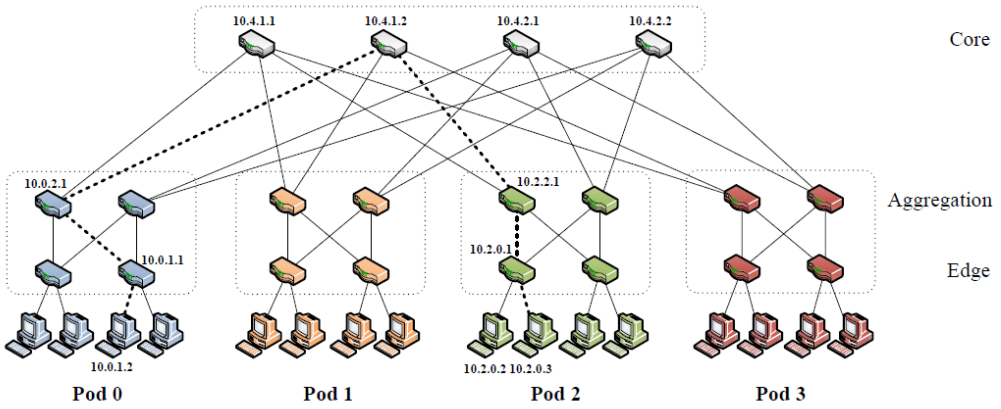
در ادامه این فصل معماری پیشنهادی مقاله را بررسی می‌نماییم. البته بخش زیادی از مقاله به بررسی ساختار شبکه مراکز داده اختصاص یافته که ما آن‌ها را در فصل یک شرح داده ایم.

- **معماری**

- آدرس دهی

تمامی آدرس های شبکه درون مجموعه 10.0.0.0/8 قرار دارد. شیوه آدرس دهی این گونه است که سوییچ‌ها آدرس هایی به فرم 10.pod.switch.1 دارند. جدول زیر این شیوه را نشان می‌دهد.

بخش	10	Pod	Position	ID
بازه	-	0 تا k-1	0 تا k-1	0 تا 255
توضیحات	عدد ثابت	pod مورد نظر	موقعیت درون pod ترتیب: از چپ به راست و از پایین به بالا	شناسه میزبان برای سوییچ‌ها این عدد یک است.



تصویر 4: معماری ارایه شده در [1]

برای مثال تمامی سوییچ های درون pod 0 به فرم 10.0.x.1 هستند. میزبان هایی که به این pod متصل هستند آدرسی به فرم 10.0.x.ID می‌گیرند. برای نمونه در تصویر 4 میزبان با آدرس 10.0.1.2 درون pod 0 قرار داشته و به سوییچی با موقعیت 1 متصل است. میزبان با آدرس 10.2.0.3 درون pod 2 قرار داشته و به سوییچی با موقعیت 2 متصل است.

لازم به ذکر است با این روش تا 2.4 میلیون میزبان می‌توان پشتیبانی کرد. البته اگر هر سرور تعدادی ماشین مجازی داشته باشد، ممکن است این روش مناسب نباشد. روش ارایه شده در فصل بعدی (پورت‌لند) چنین مسئله ای را پوشش داده است.

- جدول هدایت دومرحله ای

برای رسیدن به اهدافی که مدنظر بوده است، این طرح تغییراتی را در جدول هدایت بسته‌ها انجام داده است. به این صورت که هر مدخل در جدول اصلی یک اشاره گر به جدول فرعی دارد. ساختار اعضای جدول فرعی به فرم (suffix, port) می‌باشد. یک پیشوندِ مرحله اول از نوع terminating است اگر هیچ گونه پسوند مرحله دوم نداشته باشد. برای روشن شدن مطلب به مثال زیر توجه بفرمایید.

در شبکه تصویر 4 سوییچ 10.2.2.1 جدولی به فرم زیر دارد.

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

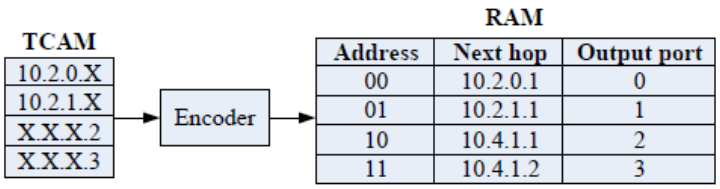
Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

بسته ای با مقصد 10.2.1.2 به پورت یک هدایت شده اما بسته ای با مقصد 10.3.0.3 بر اساس جدول ثانویه به پورت 3 هدایت خواهد شد. دقت شود که در جدول ثانویه، 8/ پسوند بوده و به هشت بیت آخر آدرس اشاره دارد.

- پیاده‌سازی lookup دومرحله ای

برای پیاده‌سازی از حافظه های CAM[1] که سرعت بالایی دارند، استفاده می‌شود. یک CAM می‌تواند جستجوی آرایه‌ها را در یک کلاک ساعت به طور موازی انجام دهد. یک نوع خاص از CAM موسوم به TCAM می‌تواند حالت سومی را علاوه بر 0 و یک، که don't care نامیده می‌شود، در جستجو لحاظ نماید. این ویژگی برای کاربرد آدرس‌یابی در شبکه کاملاً مناسب است، چرا که طول پیشوندها متغیر است. نقطه ضعف CAMها در مصرف انرژی زیاد و حجم کم حافظه‌شان است.

شکل 5 چگونگی ذخیره یک جدول دومرحله ای را نشان می‌دهد. این همان جدول صفحه قبل است که در TCAM بدین صورت ذخیره می‌شود.



تصویر 5: چگونگی پیاده‌سازی جدول دومرحله ای حافظه

چنان چه در تصویر دیده می‌شود پس از این که یک آدرس با جدول موجود در TCAM مطابقت داده شد، بر اساس یک اندیس به یک سطر در RAM هدایت می‌شود. در آن سطر آدرس گام بعدی و پورت مربوطه نوشته شده است. برای این که روش انجام کار دقیق تر مشخص شود به این مثال توجه بفرمایید. یک بسته با مقصد 10.2.0.3 در سطر اول و آخر جدول TCAM صدق می‌کند؛ با این حال بسته طبق سطر اول جدول RAM به پورت صفر (و نه پورت 3) هدایت می‌شود. دلیل این مسئله آن است که پیشوندها در خانه هایی بالاتر از پسوندها قرار می‌گیرند.

- الگوریتم مسیریابی

دو مرحله اول سوییچ‌ها (یعنی سوییچ های لبه و توزیع) آرایه های terminating (که به جدول ثانویه اشاره نمی کنند) دارند تا ارتباطات درون pod را پوشش دهند. برای مثال در سوییچ 10.2.2.1 دو سطر اول آن چنین هدفی دارند. برای خروج بسته‌ها از pod سوییچ‌ها یک آرایه به فرم 0/ دارند که به جدول ثانویه اشاره می‌کند. در جدول

[1] Content Addressable Memory

ثانویه بر اساس ID میزبان انتهایی بسته‌ها به سوییچ های هسته هدایت می‌شوند. این کار ترافیک را طور یکسان توزیع خواهد نمود. همچنین این کار کمک خواهد کرد که بسته های با یک آدرس مقصد به یک پورت هدایت شده و از به هم خوردن ترتیب بسته‌ها جلوگیری شود.

در متن مقاله به الگوریتم موجود در سوییچ های هسته نیز اشاره شده که در این جا برای اختصار از آن صرف نظر می‌کنیم. به طور خلاصه می‌توان گفت که آن‌ها از شماره pod برای هدایت ترافیک استفاده می‌کنند.

مسئله دیگری که به آن اشاره شده استفاده از یک نهاد مرکزی برای ساختن جداول جریان می‌باشد. به نظر می‌رسد که فرض شده است توپولوژی مرکزداده ثابت است (که البته فرض معقولی است.) با این حال در مقاله [3] خواهیم دید که در طرح پورت‌لند الگوریتم های توزیع‌شده مناسبی برای این کار طراحی شده است. حال می‌خواهیم به الگوریتم هایی که در این مقاله معرفی شده است بپردازیم.

در تصویر 6 الگوریتم مربوط به سوییچ های درون pod دیده می‌شود. سه حلقه اولیه نشان می‌دهد که تمامی ساب‌نت‌ها در تمامی podها برای تمامی سوییچ‌ها باید لحاظ شود. تابع addprefix(switch, prefix, port) در سطر چهارم سطرهای جریان را برای میزبان های درون همان pod می‌سازد. دقت شود که عدد مربوط به pod (در این جا x) یکسان بوده اما قسمت سوم آدرس متفاوت است. تابع addprefix دومی (در سطر 6 الگوریتم) سطرهای مربوط به ترافیکی را که باید به هسته هدایت شود مشخص می‌سازد.

```
1  foreach pod x in [0, k − 1] do
2      foreach switch z in [(k/2), k − 1] do
3          foreach subnet i in [0, (k/2) − 1] do
4              addPrefix(10.x.z.1, 10.x.i.0/24, i);
5          end
6          addPrefix(10.x.z.1, 0.0.0.0/0, 0);
7          foreach host ID i in [2, (k/2) + 1] do
8              addSuffix(10.x.z.1, 0.0.0.i/8,
               (i − 2 + z)mod(k/2) + (k/2));
9          end
10     end
11 end
```

تصویر 6: الگوریتم هدایت بسته ها

در پایان این بخش می‌خواهیم با ذکر یک مثال شیوه مسیریابی را کاملاً روشن سازیم. فرض کنید که در تصویر 4 بسته ای می‌خواهد از مبدا 10.0.1.2 به مقصد 10.2.0.3 برود. در ابتدا gateway (به آدرس 10.0.1.1) مبدا بر اساس سطر 0/ بسته را به جدول ثانویه خود می‌فرستد. در جدول ثانویه بر اساس ID مقصد (=3) بسته به پورت 2 فرستاده می‌شود. این مراحل در سوییچ بعدی تکرار شده و بسته به هسته شبکه می‌رسد. سوییچ های هسته بسته را

به pod شماره 2 هدایت کرده و در آن جا سوییچ‌ها بر اساس سطرهای terminating (10.2.0.0/24) بسته را به سمت مقصد هدایت خواهند کرد.

- کلاس‌بندی[1] جریان ها

علاوه بر مسیریابی دو مرحله ای گفته‌شده در بخش قبل، تکنیک هایی نیز برای کلاس‌بندی جریان‌ها در مقاله ارایه شده است. این تکنیک‌ها یک سربار اضافی را در هدایت بسته‌ها پدید می‌آورند با این حال در بهبود کیفیت عملکرد شبکه تاثیر مثبت دارند. در ابتدا بسته های مربوط به یک جریان شناسایی شده تا به یک پورت خروجی هدایت شوند. این کار از به هم خوردن ترتیب بسته‌ها جلوگیری خواهد کرد. در مرحله بعد باید جریان‌ها را به گونه ای روی پورت های خروجی پخش سازیم تا توازن برقرار گردد.

برای پیاده‌سازی تکنیک بالا از یک المان موسوم به FlowClassifier استفاده شده است. این المان یک ورودی و دو خروجی داشته و بر اساس آدرس IP مبدا و مقصد بسته‌ها را بر روی دو پورت خروجی خود پخش خواهد کرد. هدف اول این المان چنان چه در بند قبلی گفته شد این است که بسته های با مبدا و مقصد یکسان به پورت خروجی یکسانی هدایت شوند. هدف دوم آن است که تفاوت در اندازه بار روی پورت های خروجی به حداقل برسد.

ثابت می‌شود که این یک مسئله NP hard بوده و لذا غیرقابل حل دقیق است. الگوریتم شهودی زیر در FlowClassifier اجرا می‌شود. لازم به ذکر است که اجرای این الگوریتم (و در واقع پیاده‌سازی FlowClassifier) جایگزین جدول ثانویه در سوییچ های pod می‌گردد.

```
   // Call on every incoming packet
 1 IncomingPacket(packet)
 2 begin
 3     Hash source and destination IP fields of packet;
       // Have we seen this flow before?
 4     if seen(hash) then
 5         Lookup previously assigned port x;
 6         Send packet on port x;
 7     else
 8         Record the new flow f;
 9         Assign f to the least-loaded upward port x;
10         Send the packet on port x;
11     end
12 end
   // Call every t seconds
13 RearrangeFlows()
14 begin
15     for i=0 to 2 do
16         Find upward ports p_max and p_min with the largest and
           smallest aggregate outgoing traffic, respectively;
17         Calculate D, the difference between p_max and p_min;
18         Find the largest flow f assigned to port p_max whose size
           is smaller than D;
19         if such a flow exists then
20             Switch the output port of flow f to p_min;
21         end
22     end
23 end
```

[1] Classification

قسمت اول الگوریتم فوق که بر روی هر بسته اجرا می‌شود هدف اول را محقق می‌سازد. در واقع با استفاده از هش‌گیری جریان‌ها شناخته می‌شوند. قسمت دوم نیز به طور متناوب اجرا شده تا پورت خروجی با بار کم تر را مشخص نماید. اجرای متناوب این بخش موجب خواهد شد که ترافیک همواره به یک پورت هدایت نشود.

- زمان‌بندی[1] جریان

تحقیق‌ها نشان داده اند که بیش تر حجم ترافیک در اینترنت را جریان هایی با زمان طولانی و burstهای بزرگ تشکیل می‌دهد. در کنار این جریان‌ها تعداد زیادی جریان کوچک وجود دارد. در مقاله زمان بندی جریان بدین منظور استفاده شده تا جریان های بزرگ در مسیرشان با یکدیگر تلاقی نداشته باشند. یک زمان‌بند مرکزی آگاه به ترافیک شبکه این هدف را محقق خواهد ساخت. روش کلی کار بدین صورت است که یک المان به سوییچ های pod اضافه شده تا در صورت گذر نمودن یک ترافیک خاص از حدنصاب مشخص‌شده، آن ترافیک به عنوان یک ترافیک بزرگ در نظر گرفته می‌شود. در مرحله بعدی زمان‌بند مرکزی سعی خواهد کرد آن ترافیک را از لینک هایی که ترافیک بزرگ ندارند عبور دهد. وضعیت لینک‌ها با یک عدد باینری مشخص می‌شود. اگر یک لینک حامل ترافیک بزرگ باشد، عدد 1 به آن نسبت داده می‌شود.

- شناسایی خطا

وجود مسیرهای افزونه در توپولوژی استفاده‌شده آن را برای کاربردهای مقاوم در برابر خطا جذاب می‌سازد. در این طرح هر سوییچ یک ارتباط BFD (Bidirectional Forwarding Detection) با هر یک از همسایگانش برقرار خواهد ساخت. این ارتباط وجود هر گونه خطا را در لینک به سرعت مشخص می‌کند.

دو نوع خطا در شبکه قابل تصور است: 1- در لینک های درون pod 2- در لینک های بین هسته و pod. در متن مقاله این دو جدا بررسی شده و لذا ما نیز آن‌ها را جداگانه بررسی خواهیم کرد.

* *خطا در لینک های درون pod*: این خطا سه دسته از ترافیک‌ها را دچار مشکل خواهد ساخت. دسته اول ترافیک هایی هستند از سوییچ لایه پایینی خارج می‌شوند. در این حالت به لینک خراب وزن بی نهایت داده شده تا ترافیک از لینک جایگزین عبور نماید.

[1] Scheduling

دسته بعدی ترافیک درون pod است که از یکی از سوییچ های بالایی به عنوان گره میانی استفاده می‌کند. برای خطایابی در این زمینه سوییچ بالایی هشداری را به همه سوییچ های لایه زیرین خود فرستاده تا آن‌ها مسیر جایگزین را انتخاب نمایند.

دسته آخر نیز ترافیک بین pod هستند که به سوییچ بالایی وارد می‌شوند. در این صورت سوییچ لایه بالایی pod به هسته خبر می‌دهد که نمی تواند ترافیک را به مقصد برساند. سوییچ هسته نیز به سایر سوییچ های Pod اعلام می‌کند که از سایر سوییچ های هسته استفاده نمایند.

* *خطا در لینک های متصل به هسته:* در این حالت نیز دو دسته ترافیک دچار مشکل می‌شود. دسته اول ترافیک بین pod در حال خارج شدن است. در این حالت جدول مسیریابی محلی در سوییچ لایه بالایی درون Pod لینک خراب را کنار گذاشته و از یک سوییچ هسته دیگر استفاده می‌نماید.

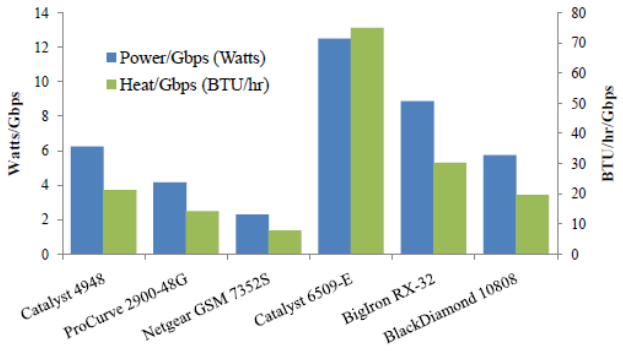
دسته دوم نیز ترافیک بین pod وارد شونده به سوییچ هسته است. در این حالت سوییچ هسته وظیفه دارد ناتوانی اش در ارسال بسته به مقصد را به همه سوییچ های لایه بالایی pod گزارش دهد تا آن‌ها از یک سوییچ هسته دیگر استفاده نمایند.

پس از رفع خرابی نیز مراحل بالا برگشت یافته تا به وضعیت اولیه برگردیم.

- مسائل مربوط به توان و گرما

در کنار کارایی و هزینه، مسئله مهم دیگری در ارتباط با مراکز داده وجود دارد و آن مصرف توان است. سوییچ های لایه بالایی در مرکزداده معمولاً هزارات وات توان مصرف می‌کنند، و لذا در مجموع ممکن است توان مصرفی یک مرکزداده به صدها کیلووات برسد. در کنار این مسئله انتشار گرما هم مهم است. سوییچ های موجود گرمای زیادی تولید کرده و لذا وجود یک سیستم خنک‌کننده مناسب حیاتی است.

در متن مقاله توان مصرفی و گرمای تولید شده در طرح پیشنهادی با سایر روش‌ها مقایسه شده است. برای مقایسه از دیتاشیت های انتشاریافته توسط سازندگان سوییچ استفاده شده است. همچنین میزان توان مصرفی و گرمای تولید شده بر پهنای‌باند کل سوییچ تقسیم شده تا یک مقدار نرمالیزه به دست آید و وابسته به اندازه سوییچ نباشد. شکل 7 این مقادیر را برای چند سوییچ موجود نشان می‌دهد.



تصویر 7: مقایسه توان مصرفی و حرارت تولیدی چند سوییچ

- **پیاده‌سازی و ارزیابی**

در یک شبکه fat tree با چهار پورت، 16 میزبان و 4 pod وجود دارد. هر کدام از این podها چهار تا سوییچ دارند. در نتیجه کلاً 20 سوییچ وجود دارد. (12 تا سوییچ pod به اضافه 4 تا سوییچ هسته) در پیاده‌سازی انجام شده این 36 المان (20 تا سوییچ + 16 تا میزبان) درون 10 تا سرور فیزیکی جای گرفته اند. این ده ماشین توسط یک سوییچ 48 پورته ProCurve 2900 با لینک های 1 Gbps به هم متصل شده اند. هر ماشین یک CPU دو هسته ای Intel Xeon دارد که هر کدام 4096 کیلوبایت کش داشته و 4 GB نیز حافظه رم دارد. ضمناً سیستم عامل تمامی آن‌ها لینوکس 2.6 است. سوییچ های هر pod درون یک ماشین قرار گرفته اند.

برای ایجاد الگوهای ارتباطی میان میزبان‌ها از سناریوهای زیر بهره گرفته شده است. البته این قید نیز لحاظ شده که هر میزبان از تنها یک فرستنده بسته دریافت نماید (نگاشت یک به یک باشد).

* *تصادفی*: هر میزبان به هر میزبان دیگر با احتمال uniform در شبکه بسته بفرستد.

* *stride(i)*: هر میزبان با اندیس x به یک میزبان با اندیس (x+i) mod 16 بسته بفرستد.

* *staggered prob (SubnetP, PodP)*: هر میزبان به یک میزبان درون سابنت خودش با احتمال SubnetP و به یک میزبان درون Pod خودش با احتمال PodPبسته می‌فرستد. مثلاً اگر تابع برای یک میزبان به صورت staggered prob (0.5, 0.3) مقداردهی شده باشد، آن میزبان در دراز مدت نیمی از بسته‌ها را به درون سابنت خودش و 0.3 آن‌ها را به pod خودش می‌فرستد. 0.2 از بسته‌ها نیز به خارج از pod خودش ارسال می‌شود. ملاحظه می‌کنیم که این سناریو نسبت به سناریوهای قبلی به شرایط واقعی نزدیک تر است.

* *ارتباط به داخل یک pod*: در این سناریو تمامی بسته‌ها به درون یک pod مشخص و با استفاده از یک سوییچ هسته مشترک فرستاده می‌شود. این یک حالت worst case بوده و در آن پارامتر oversubscription به فرم k-1:1 در می‌آید. چرا که تمامی k-1 تا pod به یک pod بسته می‌فرستند.

* *خروج به یک ID مشترک*: در این حالت نیز شرایط بدی برقرار بوده، چرا که هر میزبان تمامی بسته هایش را به میزبان هایی با ID مشترک می‌فرستد. در این حالت نیاز به FlowClassifier بوده تا ترافیک را بین سوییچ های هسته مختلف پخش نماید.

جدول زیر اطلاعات سناریوهای فوق را نشان می‌دهد.

Test	Tree	Two-Level Table	Flow Classification	Flow Scheduling
Random	53.4%	75.0%	76.3%	93.5%
Stride (1)	100.0%	100.0%	100.0%	100.0%
Stride (2)	78.1%	100.0%	100.0%	99.5%
Stride (4)	27.9%	100.0%	100.0%	100.0%
Stride (8)	28.0%	100.0%	100.0%	99.9%
Staggered Prob (1.0, 0.0)	100.0%	100.0%	100.0%	100.0%
Staggered Prob (0.5, 0.3)	83.6%	82.0%	86.2%	93.4%
Staggered Prob (0.2, 0.3)	64.9%	75.6%	80.2%	88.5%
Worst cases:				
Inter-pod Incoming	28.0%	50.6%	75.1%	99.9%
Same-ID Outgoing	27.8%	38.5%	75.4%	87.4%

در جدول بالا هر عدد پهنای‌باند مجموع در شبکه را نسبت به حالت ایده آل نشان می‌دهد. برای مثال در ترافیک تصادفی تنها 53 درصد از پهنای‌باند در حالت درخت استفاده می‌شود. با استفاده از جداول دومرحله ای که در این فصل به توصیف آن پرداختیم، این عدد به 75 درصد افزایش یافته و با استفاده از زمان بندی ترافیک به بیش از 93 درصد نیز می‌رسد.

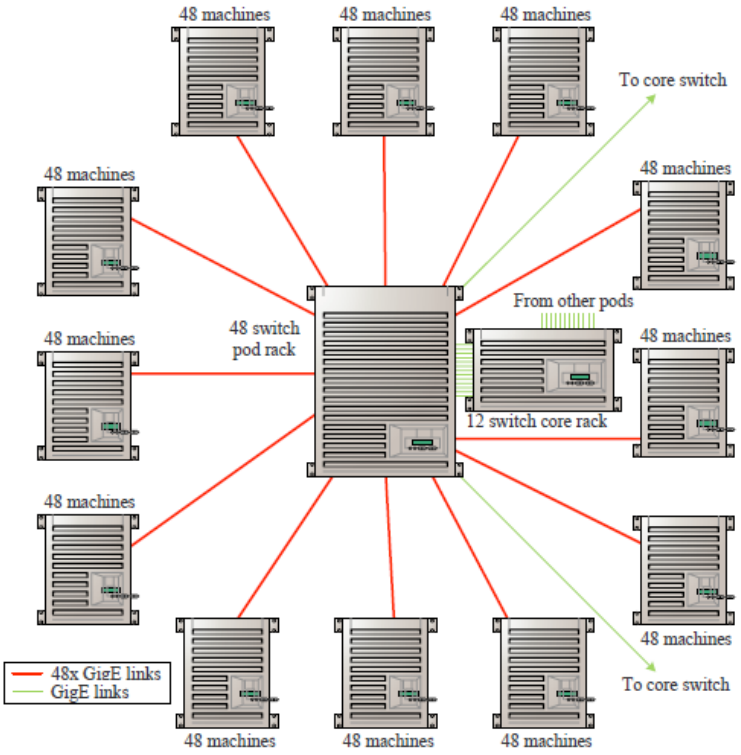
لازم به ذکر است برای ساختن جدول مذکور هر سناریو 5 بار و هر بار به مدت یک دقیقه اجرا شده است. همچنین این مسئله نیز باید ذکر شود که هر چند استفاده از کلاس بندی و زمان بندی ترافیک راندمان را افزایش می‌دهد (برای مثال در سطر آخر بیش از 50 درصد نیز افرایش داشته است) اما استفاده از آن‌ها نیازمند به صرف زمان و حافظه است. جدول زیر اطلاعات مربوط به این مسئله را نشان می‌دهد. برای مثال اگر 16 سرور داشته باشیم، 50 میکروثانیه برای هر درخواست زمان نیاز است، که این به معنای ایجاد تاخیر و کاهش کیفیت سرویس می‌باشد.

k	Hosts	Avg Time/ Req (μs)	Link-state Memory	Flow-state Memory
4	16	50.9	64 B	4 KB
16	1,024	55.3	4 KB	205 KB
24	3,456	116.8	14 KB	691 KB
32	8,192	237.6	33 KB	1.64 MB
48	27,648	754.43	111 KB	5.53 MB

- بسته‌بندی[1]

در این بخش از مقاله بحث شده است که توپولوژی fat tree این مشکل را دارد که اتصال میان اجزای آن نیاز به سیم‌کشی فراوانی دارد. البته این یک مسئله ذاتی است و چندان قابل حل نیست. ممکن است بتوان با لینک های ده گیگی مسئله را تا حدی تعدیل کرد، اما به سبب افزایش خیلی زیاد هزینه‌ها از این روش در مقاله استفاده نشده است. به هر حال نویسندگان مقاله روشی را برای کاستن از پیچیدگی موجود ارایه داده اند. ادعا شده است در این روش بسیاری از اتصالات خارجی دیگر مورد نیاز نبوده و طول سیم‌ها نیز کاهش می‌یابد. همچنین این روش امکان پیاده‌سازی مرحله به مرحله شبکه را فراهم می‌سازد.

چنان چه در تصویر 8 دیده می‌شود، هر pod شامل 576 تا ماشین و 48 تا سوییچ یک گیگی 48پورته است. برای سادگی فرض شده است هر میزبان درون یک رک جای می‌گیرد و هر رک نیز 48 میزبان دارد. پس هر pod در مجموع 12 رک (576 تقسیم بر 48) خواهد داشت. 48 سوییچی که دو لایه اول در معماری fat tree را تشکیل می‌دهند درون رک مرکزی قرار دارند. این‌ها در مجموع 1152 پورت خواهند داشت. با این راهکار بسیاری از سیم‌کشی‌ها درون رک‌ها انجام شده و لذا اهداف گفته شده محقق می‌گردد.



تصویر 8: نحوه packaging سوییچ ها و سرورها

[1] Packaging

فصل سوم

معماری PortLand

در ابتدای مقاله [3] ذکر شده است که گرایش فراوانی در صنعت به سوی انتقال محاسبات و ذخیره سازی به مراکز داده و اجرای برنامه‌ها بر روی سرورهای آن‌ها وجود دارد. در نتیجه این گرایش، Mega Datacenterها به وجود آمده اند که برنامه‌ها را بر روی صدها هزار سرور اجرا می‌سازند. برای مثال یک جستجوی اینترنتی ممکن است بیش از هزار سرور را به کار بگیرد. این روند لزوم طراحی یک شبکه قوی در مراکز داده را روشن می‌سازد. به گفته نویسندگان مقاله در شبکه های فعلی از فناوری های موجود برای شبکه های مرکزداده استفاده می‌شود که چندان مناسب نیستند. برای مثال در یک شبکه با 100000 سرور که هر کدام 32 ماشین مجازی داشته باشند، سه میلیون آدرس IP خواهیم داشت. اگر هر 25 سرور یک سوییچ لازم داشته باشند، 8000 سوییچ برای راه اندازی این شبکه لازم است. این اعداد و ارقام به خوبی نشان می‌دهند که چنین شبکه ای ابعاد بزرگ تری نسبت به شبکه های موجود دارد.

پنج سناریو در متن مقاله بررسی شده که شامل انتظارات متداول در یک شبکه مرکزداده هستند. در واقع هدف از طراحی پورت‌لند برآورده ساختن این پنج حالت است. در ادامه این سناریوها را مرور خواهیم کرد.

1- هر ماشین مجازی ممکن است از یک سرور به یک سرور دیگر انتقال یابد، ضمن این که آدرس IP آن نباید تغییر یابد. در صورت تغییر آدرس IP، ارتباطات TCP موجود از بین رفته و دچار مشکل می‌شویم.

2- مدیر شبکه نیازی به پیکربندی هر سوییچ در زمان اضافه شدن آن به شبکه نخواهد داشت. در واقع سوییچ‌ها باید plug and play باشند.

3- هر میزبان در شبکه باید بتواند با هر میزبان دیگری ارتباط برقرار سازد (از طریق هر کدام از مسیرهای فیزیکی موجود در شبکه)

4- هیچ حلقه تکراری نباید وجود داشته باشد.

5- در مقیاس بالا خرابی طبیعی است، لذا مکانیزم شناسایی خرابی باید سریع و موثر باشد. ضمن این که ارتباطات قبلی تا جایی که ماهیت خطا اجازه می‌دهد، نباید دچار مشکل بشوند.

سناریوهای 1 و 2 می‌تواند با یک فابریک لایه دو سرتاسری محقق گردد. در لایه سه سوییچ‌ها نیاز به یک پیکربندی اولیه داشته، ضمن این که انتقال ماشین های مجازی راحت نیست. (توجه شود که آدرس IP سلسله مراتبی است) از طرفی در لایه دو مشکل مقیاس پذیری به طور جدی وجود خواهد داشت. همچنین برای برآورده ساختن حالت 3 نیاز هست که هر سوییچ میلیون‌ها مدخل در جدول جریان خود داشته باشد.

سناریوی چهارم در هر دو حالت لایه 2 و 3 محقق نمی شود، چرا که در پروتکل های مسیریابی رخ دادن حلقه کاملاً طبیعی است. پروتکل درخت پوشا (STP) نیز اصلاً بهینه نیست.

در سناریوی پنجم نیاز است که پروتکل های مسیریابی خطا را سریعاً به نقطه مطلوب گزارش دهند، اما پروتکل های موجود (از جمله OSPF) بر مبنای همه‌پخشی هستند. البته می‌دانیم که ارسال همه پخشی سریع ترین راه برای رساندن پیغام است، و در متن مقاله دقیقاً مشخص نشده است که چرا ماهیت همه پخشی این پروتکل‌ها برای هشدار دادن خطا دردسرساز است.

با توجه به همه موارد ذکر شده در زمینه کاستی های پروتکل های لایه دو و سه، معماری پورت‌لند طراحی شده که اصول طراحی آن در بخش بعدی به تفصیل بررسی شده است.

- **معماری PortLand**

هدف از طراحی پورت‌لند این بوده است که آدرس دهی، مسیریابی و هدایت بسته‌ها در یک شبکه مرکزداده به گونه ای مقیاس پذیر ارایه گردد. این طراحی بر اساس این حقیقت انجام شده است که در مراکز داده توپولوژی شبکه معمولاً ثابت و شناخته شده است، چرا که تغییرات در آن نیازمند برنامه ریزی دقیق و انجام عملیات فراوان است. اگر نیاز باشد شبکه گسترش یابد، معمولاً تعدادی "برگ" به توپولوژی درخت چندریشه ای اضافه می‌شود و کلیات ساختار آن تغییر نمی یابد.

در ادامه اجزای پورت‌لند را طبق توضیحات مقاله [3] معرفی خواهیم نمود.

- مدیر فابریک[1]

پورت‌لند از یک مدیر فابریک متمرکز از لحاظ منطقی بهره می‌برد. این قسمت یک حالت نرم[2] از پیکربندی شبکه در خود نگه می‌دارد. در ادبیات کامپیوتری حالت نرم به مجموعه اطلاعاتی گفته می‌شود که برای بهینگی سیستم مفید هستند، اما وجود آن‌ها ضرورتی ندارد [ویکی پدیا]. مدیر فابریک یک پروسه است که بر روی یک ماشین مجزا اجرا

[1] Fabric Manager
[2] Soft State

شده و به منظور ARP Resolution، شناسایی خطا و مدیریت چندپخشی اجرا می‌شود. تمامی این موارد بعداً توضیح داده خواهند شد. مدیر فابریک ممکن است یک میزبان درون همین شبکه بوده یا بر روی یک شبکه کنترلی مجزا قرار داشته باشد.

همواره یک مصالحه ذاتی بین سادگی پروتکل و مقاومت[1] آن وجود داشته است، در شرایطی که یک کارکرد را بخواهیم به صورت توزیع شده پیاده‌سازی نماییم. در پورت‌لند تصمیم گرفته شده است که دانش مرکزی به صورت حالت نرم (و نه حالت سخت) گردآوری شود. با این کار، نیاز به یک مسوول شبکه که مدیر فابریک را پیکربندی نماید از بین می‌رود (پیکربندی شامل تعیین تعداد سوییچ ها، مکان آن‌ها و شناسه شان). در جریان پیاده‌سازی، احتمالاً یک یا چند نسخه پشتیبان از مدیر فابریک وجود خواهد داشت. نیازی به سنکرون سازی دقیق نمونه های کپی نیست، چرا که آن‌ها حالت سخت را در خود نگهداری نمی کنند.

- آدرس لایه فیزیکی صوری[2] (PMAC)

این آدرس‌ها که پایه ای برای هدایت و مسیریابی بهینه و نیز جابجایی VMها هستند، PMAC نامیده می‌شوند. در پورت‌لند به هر میزبان انتهایی یک PMAC نسبت داده خواهد شد و این آدرس مکان دارنده آن را در توپولوژی شبکه مشخص خواهد نمود. (سلسله مراتبی است) برای مثال تمامی نقاط درون یک pod متعلق به یک پیشوند هستند. لازم به ذکر است میزبان های انتهایی خود از وجود PMAC خبر نداشته و تنها آدرس مک واقعی (AMAC) خود را می‌دانند.

میزبان هایی که درخواست ARP می‌فرستند، PMAC مربوط به مقصد را دریافت نموده و تمامی مراحل ارسال بسته بر اساس همین PMAC انجام می‌شود. البته سوییچ نهایی دوباره PMAC را به AMAC تبدیل نموده تا میزبان مقصد از این آدرس خبری نداشته باشد. لازم به ذکر است چون سیستم آدرس دهی ذکرشده سلسله مراتبی است، جداول مسیریابی در سوییچ‌ها کوچک خواهند ماند.

سوییچ های لبه در معماری پورت‌لند، یک شماره pod یکتا و یک شماره موقعیت مشخص درون هر pod دریافت خواهند داشت. در این طرح از Location Discovery Protocol برای تخصیص این اعداد استفاده شده است. همچنین به تمامی ماشین های مجازی یک PMAC یکتا بر اساس ساختار pod.position.port.vmid نسبت داده می‌شود. این ساختار به فرم زیر است:

---

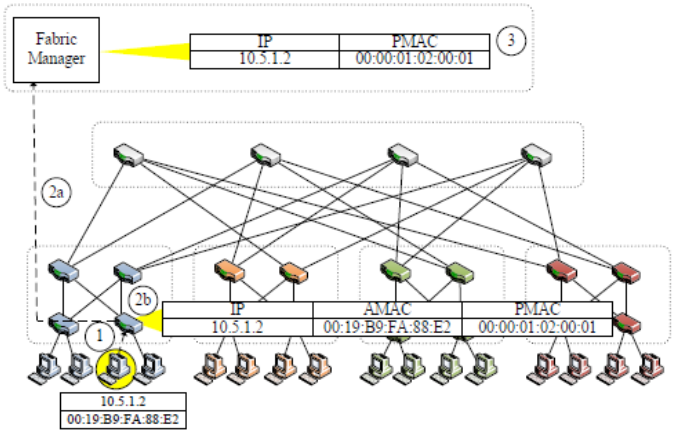
[1] Robustness

[2] Pseudo MAC

نام	pod	Position	Port	Vmid
تعداد بیت	16	8	8	16
توضیحات	نام pod	موقعیت درون pod مورد نظر	پورت سوییچ که میزبان به آن وصل شده	شناسه ماشین مجازی بر روی میزبان

لازم به ذکر است که با اضافه شدن یک VM جدید، عدد بعدیِ استفاده نشده (از مجموعه اعداد vmid) به آن نسبت داده می‌شود. اگر یک VM برای مدت مشخصی ترافیک ورودی یا خروجی نداشته باشد، عدد آن برای استفاده مجدد پس گرفته می‌شود. روال کار این طرح، به نظر شبیه به نحوه تخصیص خودکار IP توسط پروتکل DHCP می‌آید.

هنگامی که یک سوییچ آدرس مبدا MAC ناشناخته ای را مشاهده نماید، سرآیند بسته را به پشته نرم‌افزاری خود می‌فرستد. نرم‌افزار یک مدخل را در جدول PMAC محلی خواهد ساخت. در این مدخل آدرس مک واقعی (AMAC) و آدرس IP به آدرس PMAC تناظر داده می‌شوند. سوییچ آدرس PMAC را بر اساس آن چه که در بند قبلی توضیح داده شده است، خواهد ساخت. همچنین سوییچ ایجاد این مدخل را به مدیر فابریک گزارش خواهد داد. در تصویر 9 موضوع ذکر شده به تصویر کشیده شده است. در این تصویر، مرحله 1 ارسال بسته از میزبان به سوییچ را نشان می‌دهد. مراحل 2a و 2b نیز ساخت مدخل جریان در سوییچ و ارسال آن به مدیر فابریک را نشان می‌دهد.



تصویر 9: معماری پورت‌لند

پس از این که مدیر فابریک از یک PMAC و آدرس IP متناظر با آن اطلاع یافت، برای پاسخ دادن به درخواست ARP از آن استفاده می‌کند. افزون بر این، سوییچ های لبه هنگامی که یک بسته به مقصد یکی از میزبان های متصل به آن‌ها می‌رسد، آدرس PMAC موجود در سرآیند را با آدرس AMAC عوض می‌کنند.

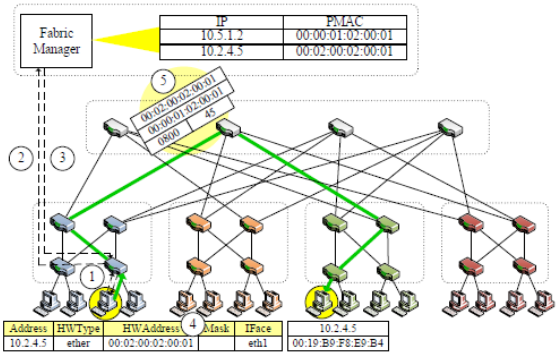
در متن مقاله مزایای چنین روش آدرس دهی شرح داده شده است که در ادامه آن را مرور می‌کنیم.

در این روش شناسه میزبان از مکان آن مجزا است، لذا با سخت افزار سوییچ های تجاری سازگار است. علاوه بر این، در این روش یک پروتکل جدید معرفی نشده و تبدیل PMAC به AMAC و برعکس به سادگی توسط پشته نرم‌افزاری سوییچ امکان پذیر است.

- ARP مبتنی بر پروکسی

در اترنت پیغام های ARP به طور عادی به تمام ناحیه لایه دو پخش می‌شوند. در این مقاله از مدیر فابریک برای تغییر وضعیت فعلی بهره برده شده است. چنان چه در تصویر 10 دیده می‌شود، در مرحله یک سوییچ درخواست ARP را تفسیر نموده و سپس آن را به مدیر فابریک می‌فرستند (مرحله دو). مدیر فابریک در جدول PMAC خود به دنبال یک مدخل با IP مربوطه می‌گردد. اگر چنین مدخلی یافته شد، PMAC مورد نظر را به سوییچ بر می‌گرداند (مرحله 3). در مرحله 4 سوییچ ARP Reply را به میزبان برمیگرداند.

ممکن است مدیر فابریک تناظر بین IP و PMAC را نداشته باشد. (برای مثال بعد از یک خرابی در شبکه). در این حالت مدیر فابریک بسته را بین تمام سوییچ‌ها پخش خواهد کرد. شیوه ارسال همه‌پخشی بهینه در حالت بدون خطا سرراست است: بسته درخواست ARP به سوییچ های هسته فرستاده شده، آن‌ها بسته را به هر کدام از podها فرستاده و در نهایت بسته مذکور به سوییچ های لبه می‌رسد. میزبان مقصد AMAC را در پاسخ بر می‌گرداند، که سوییچ ورودی آن را به PMAC مناسب تبدیل کرده و در نهایت پاسخ را به مدیر فابریک و میزبان درخواست‌کننده می‌فرستد.



تصویر 10: نحوه هدایت بسته در پورت‌لند

توجه شود که فرستنده همواره PMAC را دریافت کرده (چه مدیر فابریک آن را در خود داشته باشد و چه نداشته باشد) و تمامی مسائل مربوط به هدایت و مسیریابی بر اساس این آدرس های سلسله مراتبی انجام می‌شود. این روش بسیار مقیاس پذیرتر از اترنت معمولی است. در اترنت برای هر کدام از آدرس های MAC (که تعداد آن‌ها ممکن است به صدها هزار برسد) یک مدخل جدول در هر سوییچ مورد نیاز است، اما در این شیوه تنها O(K) آرایه لازم است. k تعداد podهای موجود در شبکه است. در واقع سوییچ‌ها تنها لازم است بدانند که یک بسته به کدام pod قرار است هدایت شود. البته در شرایطی که بخواهد توزیع بار بین چندین مسیر انجام شود، اطلاعات بیشتری مورد نیاز است.

در این جا لازم است به یک مسئله در ارتباط با جابجایی VMها اشاره شود. پس از این که یک ماشین مجازی از یک سرور به سرور دیگری انتقال یافت، یک پیغام ARP به سوییچ فرستاده و در آن IP جدید خودش را قرار می‌دهد. این پیغام به مدیر فابریک فرستاده شده تا اصلاحات لازم اعمال شود. (احتمالاً آدرس مک واقعی میزبان مبنای کار قرار خواهد گرفت، چرا که در این حالت IP و PMAC هر دو تغییر پیدا کرده و مسلماً نمی توانند کلید پایگاه داده باشند. البته در متن مقاله به این موضوع اشاره ای نشده است.) در این میان ممکن است برخی میزبان‌ها آدرس PMAC قبلی را که دیگر نامعتبر است، در حافظه cache خود ذخیره داشته باشند. برای حل این مشکل، مدیر فابریک یک پیغام بی‌اعتبارسازی[1] را به سوییچی که VM جابجاشده قبلاً به آن متصل بوده است می‌فرستد. اگر هر بسته ای به مقصد PMAC قبلی برسد، سوییچ پیغام بی‌اعتبارسازی را به فرستنده بر می‌گرداند. البته ممکن است سوییچ بسته اول را خودش تصحیح نموده و به مقصد هدایت کند، تا packet loss نداشته باشیم.

- شناسایی مکان توزیع شده

در پورت‌لند از موقعیت سوییچ‌ها برای هدایت بهتر و موثرتر ترافیک بهره برده می‌شود. موقعیت سوییچ‌ها می‌تواند به طور دستی تعیین شود، که البته با هدف طراحی پورت‌لند در تناقض است. برای این که روال کار خودکار باشد، یک پروتکل ویژه طراحی شده است. این پروتکل Location Discovery Protocol یا به اختصار LDP نام داشته که نیاز به پیکربندی اولیه را مرتفع می‌سازد. سوییچ‌ها در پورت‌لند تا زمانی که موقعیت خود را شناسایی نکرده باشند، از هدایت بسته خودداری می‌کنند.

سوییچ‌ها به طور متناوب بسته های LDM (پیغام شناسایی مکان) به تمامی پورت های خود ارسال می‌نمایند. LDMها شامل اطلاعات زیر است:

[1] invalidation

- شناسه سوییچ (switch_id): یک شناسه که برای هر سوییچ یکتا است، برای مثال کمترین مقدار آدرس MAC بین تمام پورت ها. این پارامتر را سوییچ از همان ابتدا می‌داند.
- عدد مربوط به pod: عددی که برای تمامی سوییچ های درون یک pod مشترک است. سوییچ های هسته چنین عددی نخواهند داشت.
- موقعیت (pos): عددی که به هر سوییچ لبه تخصیص داده شده و درون هر pod یکتاست.
- سطح درخت (level): عددی که نشان می‌دهد سوییچ در لبه، هسته یا لایه توزیع قرار دارد و مقدار آن 0، 1 و یا 2 است.
- Up/down (dir): یک عدد باینری که مشخص می‌کند یک پورت رو به بالا (ریشه درخت) یا رو به پایین است.

منطق اساسی در نحوه کار الگوریتم LDP (که در تصویر 11 دیده می‌شود)، این است که سوییچ های لبه پیغام LDM را تنها از سوییچ های لایه توزیع دریافت خواهند داشت، چرا که میزبان های انتهایی در پروتکل شرکت ندارند. بنابراین اگر سوییچی از تعداد مشخصی از پورت هایش چنین پیغامی را دریافت ننماید، مشخص می‌شود که در لایه لبه قرار دارد. سپس این موضوع به سوییچ های لایه توزیع اطلاع داده شده و آن‌ها نیز از موقعیت خود باخبر می‌شوند. به همین ترتیب سوییچ های هسته زمانی از موقعیت خود خبردار می‌شوند که بفهمند تمامی پورت هایشان به سوییچ لایه توزیع متصل است.

Algorithm 1 *LDP_listener_thread*()

```
1: While (true)
2:     For each tp in tentative_pos
3:         If (curr_time − tp.time) > timeout
4:             tentative_pos ← tentative_pos − {tp};
5:     ▷ Case 1: On receipt of LDM P
6:     Neighbors ← Neighbors ⋃ {switch that sent P}
7:     If (curr_time − start_time > T and |Neighbors| ≤ k/2)
8:         my_level ← 0; incoming_port ← up;
9:         Acquire_position_thread();
10:    If (P.level = 0 and P.dir = up)
11:        my_level ← 1; incoming_port ← down;
12:    Else If (P.dir = down)
13:        incoming_port ← up;
14:    If (my_level = −1 and |Neighbors| = k)
15:        is_core ← true;
16:        For each switch in Neighbors
17:            If (switch.level ≠ 1 or switch.dir ≠ −1)
18:                is_core ← false; break;
19:        If (is_core = true)
20:            my_level ← 2; Set dir of all ports to down;
21:    If (P.pos ≠ −1 and P.pos ⊈ Pos_used)
22:        Pos_used ← Pos_used ⋃ {P.pos};
23:    If (P.pod ≠ −1 and my_level ≠ 2)
24:        my_pod ← P.pod;
25:
26:    ▷ Case 2: On receipt of position proposal P
27:    If (P.proposal ⊈ (Pos_used ⋃ tentative_pos))
28:        reply ← {"Yes"};
29:        tentative_pos ← tentative_pos ⋃ {P.proposal};
30:    Else
31:        reply ← {"No", Pos_used, tentative_pos};
```

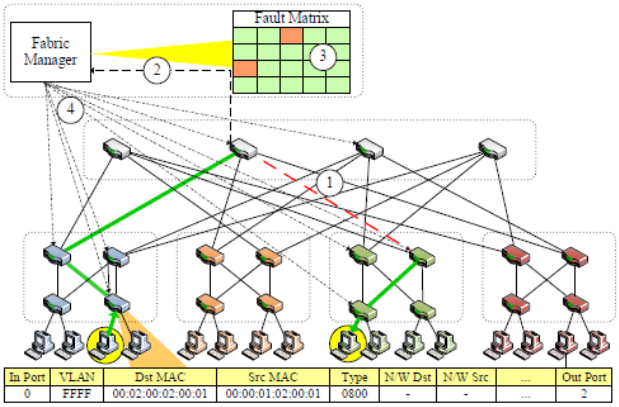
تصویر 11: الگوریتم LDP

حال می‌خواهیم به بیان اجمالی عملکرد الگوریتم بپردازیم. خط 2 تا 4 به تعیین موقعیت اختصاص داشته و بعداً توضیح داده می‌شود. در خط 6 لیست همسایه هایی که پیغام ارسال کرده اند بروز می‌شود. در خط 7 و 8 اگر سوییچ برای مدتی طولانی (معیار عدد T است) از بیش تر از k/2 همسایه‌ها پیغام دریافت ننماید، نتیجه می‌گیرد که سوییچ لبه است (بر اساس همان منطقی که کمی قبل گفته شد). البته سوییچ می‌تواند با ارسال پینگ مطمئن شود خطایی در کار نیست و عدم ارسال پیغام LDM به علت آن است که تعدادی از همسایه‌ها سرور هستند. این مسئله در الگوریتم فوق دیده نشده است.

در خط 10 اگر یک سوییچ پیغام LDM را از سوییچی که می‌داند در لبه قرار دارد، دریافت نماید (بر روی پورت با جهت بالا) نتیجه می‌گیرد که در لایه توزیع قرار دارد. خطوط 14 تا 20 الگوریتم به تعیین سوییچ های هسته می‌پردازند که جزییات آن از حوصله بحث خارج است.

- **مسیریابی مقاوم در برابر خطا**

از آن جایی که در مسیریابی پورت‌لند توپولوژی ثابت فرض شده است، شناسایی خطا اهمیت بالایی دارد. ارتباطات LDP علاوه بر این که به شناسایی توپولوژی کمک می‌نمایند، برای شناسایی زنده بودن[1] همسایه‌ها نیز به کار می‌روند. اگر یک سوییچ برای مدتی هیچ پیغام LDP از همسایه اش دریافت نکرد (که در این بحث این پیغام‌ها می‌توانند keepalive نیز نامیده شوند) فرض می‌کند که لینک کار نمی کند (مرحله یک در تصویر 12) پس از آن در مرحله 2 این موضوع را به مدیر فابریک اطلاع می‌دهد. مدیر فابریک نیز یک ماتریس خطا دارد که تمامی ارتباطات موجود در شبکه در آن لحاظ شده است. در مرحله 3 مدیر فابریک این ماتریس خطا را به روز می‌سازد. در متن مقاله جزییات این ماتریس توضیح داده نشده است. در نهایت، مدیر فابریک به تمام سوییچ هایی که از این خرابی تاثیر می‌پذیرند اطلاع داده می‌شود (مرحله 4). در این صورت سوییچ‌ها جدول جریان خود را مجدداً محاسبه می‌نمایند.



تصویر 12: خطایابی در پورت‌لند

در روش های مسیریابی سنتی تعداد زیادی پیام لازم است. در واقع $O(n^2)$ پیام لازم است، چرا که هر کدام از n سوییچ باید به n-1 سوییچ دیگر خرابی را اطلاع دهد. اما در پورت‌لند $O(n)$ پیام نیاز است، از آن جایی که سوییچ خرابی را به مدیر فابریک گزارش داده و آن نیز به n سوییچ موجود اطلاع می‌دهد.

[1] liveness

- **مقایسه با سایر معماری های پیشنهادی**

در جدول زیر معماری پورت‌لند با دو طرح دیگر به نام TRILL و SEATTLE مقایسه شده است.

System	Topology	Forwarding		Routing	ARP	Loops	Multicast
		Switch State	Addressing				
TRILL	General	*O(number of global hosts)*	Flat; MAC-in-MAC encapsulation	Switch broadcast	All switches map MAC address to remote switch	TRILL header with TTL	ISIS extensions based on MOSPF
SEATTLE	General	*O(number of global hosts)*	Flat	Switch broadcast	One-hop DHT	Unicast loops possible	New construct: groups
PortLand	Multi-rooted tree	*O(number of local ports)*	Hierarchical	Location Discovery Protocol; Fabric manager for faults	Fabric manager	Provably loop free; no additional header	Broadcast-free routing; multi-rooted spanning trees

چنان چه دیده می‌شود، TRILL از کپسوله سازی بسته های MAC درون بسته های MAC استفاده می‌کند. مسئله ای که در هر دو معماری مذکور وجود دارد این است که از آدرس دهی تخت استفاده کرده، لذا حجم جدول جریان سوییچ‌ها در این‌ها بزرگ است. در پورت‌لند استفاده از PMAC موجب می‌شود که حالت ذخیره شده در سوییچ‌ها محدود باشد. این ویژگی در مراکز داده جدید که در آن‌ها تمامی سرورها با یکدیگر ممکن است برای بازه های کوتاهی ارتباط داشته باشند مهم است. برای مثال می‌توان به یک جستجوی اینترنتی اشاره کرد که در آن سرورها همکاری گسترده ای در جهت یافتن پاسخ دارند.

هر دو معماری TRILL و SEATTLE از الگوریتم های همه پخشی برای تشخیص توپولوژی شبکه بهره می‌برند، اما در پورت‌لند دیدیم که از یک دانش اولیه در زمینه توپولوژی شبکه استفاده شده تا سوییچ‌ها بر اساس ارتباطات محلی (همان پروتکل LDP) موقعیت خودشان را تشخیص دهند.

در متن مقاله این طرح‌ها در زمینه های دیگری نیز مقایسه شده اند که برای شرح آن لازم است معماری TRILL و SEATTLE کامل شناخته شود.

- **پیاده‌سازی پورت‌لند**

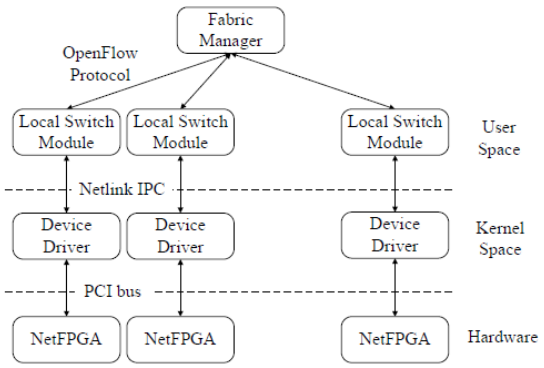
سخت افزار موجود شامل 20 عدد سوییچ NetFPGA بیست پورته می‌باشد. این بردها بر روی ماشین اینتل دوهسته ای 3.2 GHz سوار شده اند. این ماشین 3 GB حافظه درونی نیز داشته است. همچنین 16 میزبان انتهایی پیش‌بینی شده است که همگی آن‌ها سیستم‌عامل لینوکس داشته اند.

سوییچ‌ها از پروتکل OpenFlow 0.8 استفاده کرده اند. ظاهراً در زمان نگارش مقاله این پروتکل در وضعیت تست بوده است و هنوز یک استاندارد به حساب نمی آمده است. همچنین سوییچ‌ها مجهز به حافظه 32 سطری TCAM

بوده و البته یک SRAM با حافظه 32K نیز داشته اند. ده فیلد هر بسته ورودی (آدرس های لایه های دو تا چهار) مبنای هدایت بوده است.

در ارتباط با معماری سیستم نیز این گونه بحث شده است تمامی بسته های درخواست ARP و عضویت IGMP به پشته نرم‌افزار سوییچ هدایت می‌شود. این پشته بر روی همان کامپیوتری است که سوییچ‌ها بر روی آن قرار دارند. لازم به ذکر است که چند بسته اول هر جریان هدایت سخت افزاری را از دست داده و توسط نرم‌افزار هدایت می‌شوند.

طرح کلی پیاده‌سازی پورت‌لند در شکل 13 دیده می‌شود.



تصویر 13: طرح کلی پیاده‌سازی پورت‌لند

همان طور که دیده می‌شود مدیر فابریک از طریق پروتکل اپن فلو با ماژول نرم‌افزاری سوییچ‌ها در ارتباط است. یکی از کارهایی که مدیر فابریک انجام می‌دهد تست کردن متصل بودن سوییچ‌ها است. این تست دائمی مدیر فابریک را قادر می‌سازد تا ماتریس خطا را کامل نماید.

مسئله دیگر در ارتباط با مدیر فابریک این است که در پیاده‌سازی طرح از حالت کنترلی out of band برای آن استفاده شده است، حال آن که طبق ادعای خودشان می‌توانست به سادگی یک المان همانند سایر سرورها در نظر گرفته شده و از ارتباطات in band برای آن استفاده شود.

در پایان این بخش جدول زیر را آورده ایم که در آن پیچیدگی حالت های ذخیره شده در هر سوییچ و نیز مدیر فابریک آورده شده است. برای مثال سطرهایی که برای نگاشت IP به PMAC لازم است از مرتبه O(k/2) بوده و این یعنی مرتبه تعداد سطرها هیچ گاه از نصف تعداد پورت های سوییچ بیش تر نخواهد شد، حتی اگر شبکه صدها گره داشته باشد. نحوه محاسبه این مقادیر در متن مقاله بدون توضیح مانده است.

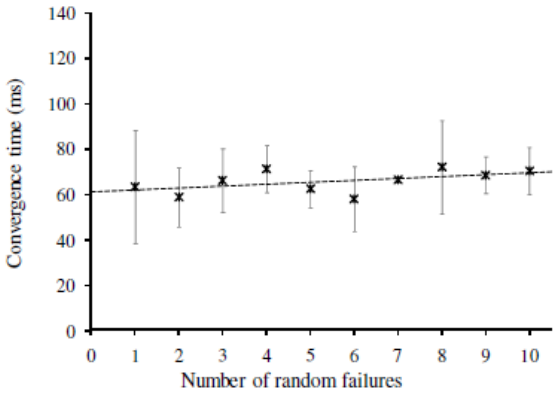
$k = Number\ of\ ports\ on\ the\ switches,$
$m = Number\ of\ local\ multicast\ groups,$
$p = Number\ of\ multicast\ groups\ active\ in\ the\ system.$

State	Switch	Fabric Manager
Connectivity Matrix	$O(k^3/2)$	$O(k^3/2)$
Multicast Flows	$O(m)$	$O(p)$
$IP \rightarrow PMAC$ mappings	$O(k/2)$	$O(k^3/4)$

- **ارزیابی**

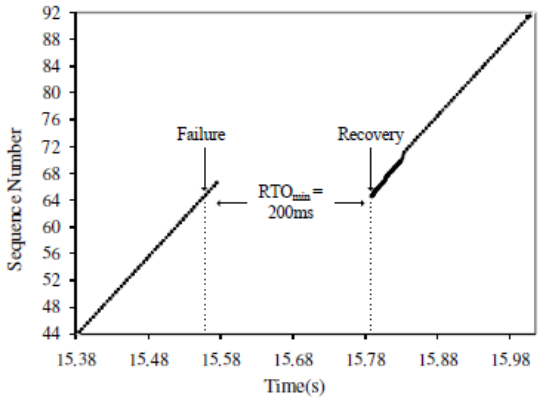
در این بخش کارایی و مقیاس پذیری طرح پورت‌لند، طبق نمودارهای رسم شده در متن مقاله بررسی می‌شود.

- *زمان همگرایی با خطاهای افزایش داده شده*: در این آزمایش زمان همگرایی برای جریان های UDP و در حضور خطاهای تصادفی برای لینک‌ها سنجیده شده است. یک فرستنده بسته هایی را با نرخ 250 Mbps ارسال نموده و گیرنده نیز در یک pod جداگانه قرار دارد. در حالتی که حداقل یکی از خطاها در مسیر معمول بین فرستنده و گیرنده رخ بدهد، زمان بازیابی را بررسی کرده ایم. در تصویر 14 زمان همگرایی برای 20 بار اجرای آزمایش نشان داده شده است.



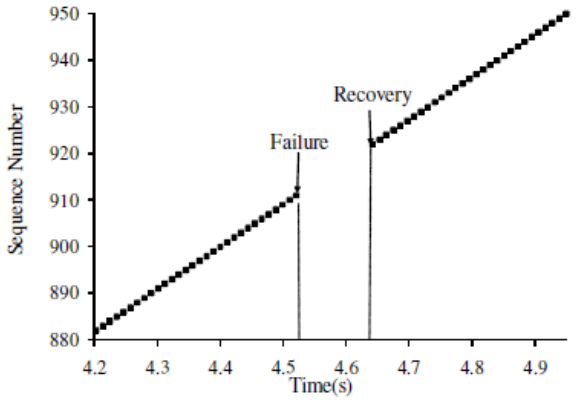
تصویر 14: زمان همگرایی برای ترافیک UDP

- *همگرایی TCP*: همان آزمایش قبلی برای ترافیک TCP نیز تکرار شده است. برای مشاهده ترافیک شبکه از نرم‌افزار tcpdump استفاده شده است. چنان چه در تصویر 15 دیده می‌شود، زمان همگرایی برای TCP همواره بیشتر بوده است. البته تمامی شرایط مربوط به شبکه یکسان در نظر گرفته شده، و این تفاوت به خاطر مکانیزم مبتنی بر پنجره TCP است. هنگامی که خطایی رخ می‌دهد، پنجره جاری از ابتدا فرستاده می‌شود. در آزمایش انجام شده مقدار زمان تایم اوت 200 میلی ثانیه در نظر گرفته شده است.



تصویر 15: زمان همگرایی برای پروتکل TCP

- *همگرایی چندپخشی*: زمان لازم برای ایجاد یک هسته جدید، زمانی که یکی از عضو[1]های گروه ارتباطش را با هسته از دست می‌دهد، آزمایش گردید. در این آزمایش، فرستنده جریان ترافیکی را به گروهی که سه عضو دارد فرستاد. در زمان 4.5 میلی ثانیه خطایی که منجر به قطع ارتباط یکی از عضوها شد، ایجاد کردیم. نتایج در تصویر 16 دیده می‌شود.

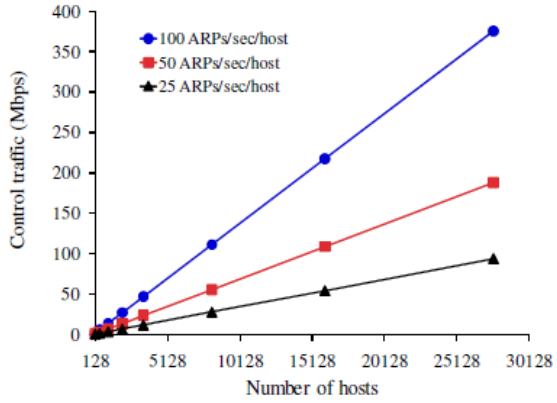


تصویر 16: همگرایی چندپخشی

- *مقیاس پذیری*: یکی از نگرانی‌ها در طراحی پورت‌لند مقیاس پذیر بودن مدیر فابریک آن در توپولوژی‌ها بزرگ تر است. از آن جایی که چنین نمونه بزرگی در دسترس نبوده است، سعی شده است از همان

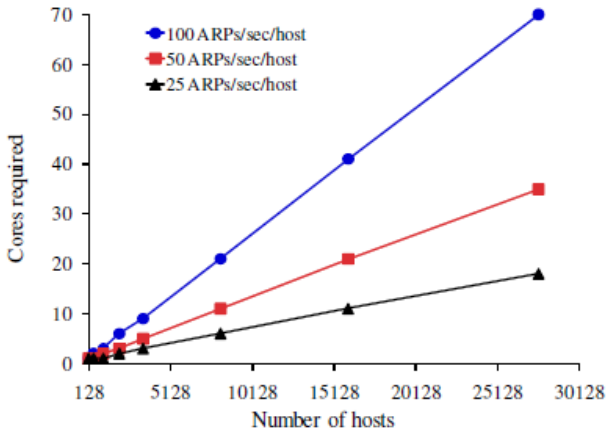
[1] subscriber

سیستم پیاده‌سازی شده اطلاعات لازم استخراج گردد. در تصویر 17 تعداد پیغام های کنترلی ARP که مدیر فابریک در اندازه های مختلف سایز کلاستر انتظار دارد ببیند، مشاهده می‌شود.



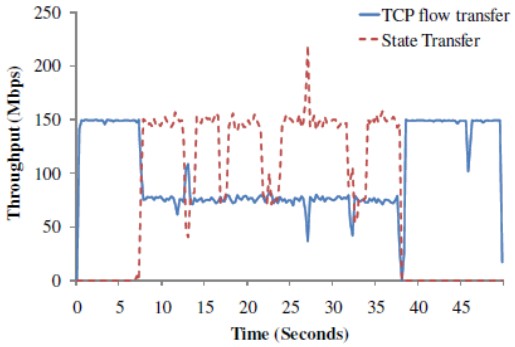
تصویر 17: مقیاس پذیری (ترافیک کنترلی بر حسب تعداد میزبان‌ها)

آز آن جایی که نویسندگان مقاله علاقه داشته اند طرح را در شرایط بحرانی ببینند، شرایطی در نظر گرفته شده که در آن هر کاربر 25، 50 و 100 پیغام درخواست ARP در هر ثانیه بفرستد. به گفته خود آن‌ها حتی عدد 25 نیز برای مراکز داده آن زمان و با توجه به حافظه کش با تایم اوت 60 ثانیه در کامپیوترها، عدد بزرگی بوده است. در یک مرکزداده که هر کدام از 27468 میزبان موجود 100 پیغام ARP در ثانیه بفرستند، مدیر فابریک باید 376 Mbps ترافیک کنترلی را مدیریت نماید. چالش بزرگ تر میزان زمانی است که CPU باید به این‌ها اختصاص دهد. خوشبختانه این پروسه کاملاً قابلیت موازی سازی داشته و می‌تواند در هسته های مختلف اجرا گردد. تصویر 18 شرایط سخت افزاری لازم برای مدیر فابریک بر حسب تابعی از تعداد میزبان‌ها را نشان می‌دهد.



تصویر 18: هسته‌های پردازشی لازم به ازای تعداد میزبان‌ها

- *جابجایی ماشین های مجازی*: آخرین مسئله ای که مورد ارزیابی قرار گرفته است، میزان پشتیبانی پورت‌لند از جابجایی ماشین های مجازی است. در این آزمایش فرستنده یک ترافیک 150 Mbps را به یک ماشین مجازی درون یکی از podها ارسال می‌کند. سپس این ماشین مجازی به یک pod دیگر منتقل گشته است. پس از جابجایی، ماشین مجازی آدرس MAC جدیدش را اعلام نموده و سوییچ این موضوع را به تمامی میزبان هایی که با وی در ارتباط بوده اند (از جمله فرستنده در همین آزمایش)، خبر می‌دهد. تصویر 19 وضعیت ارتباط TCP ذکرشده را در تمامی حالت‌ها نشان می‌دهد.



تصویر 19: گذردهی در صورت جابجایی ماشین مجازی

در زمان جابجایی ماشین مجازی، گذردهی ارتباط TCP به علت این که اطلاعات آن ماشین در حال انتقال به سرور جدید است، کاهش می‌یابد. (در واقع بخشی از پهنای‌باند به انتقال ماشین مجازی اختصاص یافته است.) پس از پایان انتقال، ترافیک TCP در یک لحظه به صفر می‌رسد، چرا که فرستنده در حال شناسایی آدرس جدید مقصد است. پس از دریافت آدرس جدید، ترافیک به وضعیت اولیه باز می‌گردد (ثانیه 40 به بعد).

جمع‌بندی و نتیجه‌گیری

در این پروژه ابتدا معماری و پیاده‌سازی شبکه‌های مرکز داده را مورد بررسی قرار دادیم. شبکه‌های مرکز داده ویژگی‌های جالب توجهی دارند که توجه به آن‌ها حیاتی می‌باشد. برخی از این ویژگی‌ها عبارت‌اند از:

- *ثابت بودن توپولوژی در بازه‌های طولانی*: طراحی و نصب شبکه‌های مرکز داده معمولاً هزینه‌بر بوده، لذا اعمال تغییرات در آن‌ها دیر به دیر انجام می‌شود. این یک مزیت برای طراحی پروتکل‌های شبکه در مرکز داده است. در این پروژه معماری پورت‌لند را بررسی نموده و دیدیم که بر اساس فرض ثابت بودن توپولوژی، سوییچ‌ها به صورت خودکار جایگاهشان را در گراف شبکه مشخص می‌سازند.
- *تعداد زیاد میزبان‌های شبکه*: در یک مرکز داده تعداد زیادی سرور وجود داشته که به لطف مجازی‌سازی، هر کدام از آن‌ها نیز تعدادی ماشین مجازی خواهند داشت. این مسئله موجب می‌شود که میلیون‌ها نقطه انتهایی با یکدیگر و نیز خارج از مرکز داده ارتباط برقرار سازند. پروتکل‌های معمول (نظیر اترنت) برای این هدف مناسب نیستند. در این پروژه دو طرح پیشنهادی بررسی شد که برای مسائل شبکه‌ای (همچون مسیریابی و آدرس‌دهی) در شبکه مرکز داده راهکار ارایه داده‌اند.
- *جابجایی نقاط انتهایی*: تکنولوژی مجازی‌سازی نه تنها تعداد نقاط انتهایی را افزایش می‌دهد، بلکه جابجایی ماشین‌های مجازی نیز یک اتفاق معمول در این مدل است. اگر از آدرس‌دهی سلسله مراتبی (نظیر IP) استفاده شود، جابجایی ماشین‌های مجازی ارتباطات TCP موجود را (به علت تغییر آدرس) قطع کرده و این وضعیت چندان مطلوب نیست. در معماری پورت‌لند تدابیری اتخاذ شده است تا در صورت استفاده از آدرس‌های سلسله مراتبی این مشکلات پیش نیاید.

با توجه به ویژگی‌های بالا، پژوهشگران طرح‌هایی برای معماری شبکه مرکز داده ارایه کرده اند. دو نمونه از این طرح‌ها را در این پروژه شرح داده ایم. البته طرح‌های بررسی شده پیچیدگی‌های فراوانی داشته و به نظر می‌رسد مسئله را خیلی اصولی حل نکرده اند. شاید بهره‌گیری از معماری SDN راهکار نهایی برای شبکه مرکز داده باشد. لایه کنترلی متمرکز در این معماری می‌تواند آدرس‌دهی، مسیریابی و بسیاری از مسائل را حل نماید؛ با این حال باید چاره‌ای اندیشید تا کنترلر به یک گلوگاه تبدیل نشود.

منابع:

[1] Al-Fares, Mohammad, Alexander Loukissas, and Amin Vahdat. "A scalable, commodity data center network architecture." ACM SIGCOMM Computer Communication Review 38.4 (2008): 63-74.

[2] Farrington, Nathan, Erik Rubow, and Amin Vahdat. "Data center switch architecture in the age of merchant silicon." High Performance Interconnects, 2009. HOTI 2009. 17th IEEE Symposium on. IEEE, 2009.

[3] Mysore, Radhika Niranjan, et al. "PortLand: a scalable fault-tolerant layer 2 data center network fabric." SIGCOMM. Vol. 9. 2009.

[4] Schwabe, Arne, and Holger Karl. "Using MAC addresses as efficient routing labels in data centers." Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014.