دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

گزارش درس روش تحقیق

گرایش نرم افزار

عنوان

افزایش کارآیی در شبکه‌های نرم افزار بنیان با استفاده از مجازی‌سازی توابع شبکه

نگارش

پرهام الوانی

استاد راهنما

دکتر سعید شیری

خرداد ۱۳۹۵

چکیده

در روش تخصیص منابع براساس مجازی سازی توابع شبکه برای بهبود تخصیص منابع در شبکه‌های نرم‌افزار بنیان از مجازی سازی توابع شبکه استفاده می‌شود، در این روش تمامی قسمت‌های شبکه به صورت نرم‌افزاری پیاده‌سازی میشوند و برای بهبود زمان لازم برای پردازش بسته‌ها از کتابخانه‌هایی مانند DPDK استفاده میشود.

واژه‌های کلیدی:

شبکه‌های نرم افزار بنیان، مجازی‌سازی توابع شبکه، کتابخانه‌های پردازش بسته‌ها

فهرست عناوین

صفحه

فهرست اشکال

صفحه

صفحه

فهرست جداول

No table of figures entries found.

۱

فصل اول

مقدمه

مقدمه

شبکه‌های کامپیوتری به طور معمول از تعداد زیادی از دستگاه‌های شبکه‌ای مانند روترها، سوئیچ‌ها و انواع متعددی از میان افزارها ساخته شده‌اند که وظیفه‌ی بازرسی ترافیک و یا انتقال آن، به دست این میان افزارها می‌باشد.

اپراتورهای شبکه باید با استفاده شرایط شبکه با پیکربندی‌ها و سیاست‌هایی که در نظر می‌گیرند اتفاقات شبکه را کنترل و برنامه ریزی کنند. در واقع اپراتور شبکه مسئول پیاده‌سازی این سیاست‌ها و پیکربندی‌هاست و باید به صورت دستی سیاست‌های سازمان را به دستورات پیکربندی تبدیل کند.

امروزه با گسترده شدن شبکه‌ها و افزوده شدن انواع مختلفی از میان افزارها برای کنترل و بررسی ترافیک شبکه و مسیریابی آن کار مدیریت شبکه امری بسیار سخت و پیچیده شده است و هزینه‌ی پیاده سازی سخت افزاری ابزارآلات شبکه‌ای بسیار افزایش یافته است.

شبکه‌های نرم افزار بنیان به ما این وعده را می‌دهند که با نرم افزاری کردن این فرآیند این مشکلات را حذف کنند ولی مشکل اصلی نرم افزاری سازی مدیریت شبکه سربار زیاد آن می‌باشد. در این مقاله قصد داریم این سربار را به وسیله‌ی مجازی سازی توابع شبکه‌ای کاهش دهیم.

۲

فصل دوم

ادبیات موضوع

ادبیات موضوع

۲.۱ شبکه‌های نرم‌افزار بنیان

سیر تکاملی دستگاه ها و تجهیزات جانبی سیار ،مجازی‌سازی سرورها و ظهور سرویسهای کلاود، منجر به بازبینی دوباره معماری رایج شبکه‌ها شده است .معماری بسیاری از شبکه‌های سنتی، سلسله مراتبی است که با استفاده از گره‌هایی از سوئیچ های اترنت در یک ساختار درختی شکل می‌گیرد. این معماری زمانی که بحث ارتباطات کلاینت/سرور مطرح شود، ملموس تر خواهد بود اما چنین معماری ایستایی، برای ارتباطات پویا و نیازهای شرکت ها در زمینه مراکز داده و رسانه‌های سرویس دهنده، کافی نیست. مواجهه با نیاز های کنونی بازار با استفاده از معماری‌های متداول شبکه تقریباً غیرممکن است .شرکتهای فناوری اطلاعات، برای رویارویای با مسائلی نظیر رکود یا کاهش بودجه از ابزارهای مدیریتی در سطح ماشین و پردازش‌های دستی بهره می گیرند .شرکتهای ارائه دهنده سرویسهای مخابراتی نیز با چالشهای تشابهی روبرو هستند، چرا که تقاضا برای دسترسی به پهنای باند شبکه‌های پویا رو به افزایش چشمگیری است؛ در عین حال، با افزایش هزینه‌های مربوط به تجهیزات مرکزی و کاهش درآمد، سود این شرکتها به خطر می افتد. معماری شبکه‌های موجود، به گونه ای طراحی نشده اند که نیازهای کنونی شرکت ها، سرویس دهنده های مخابراتی و کاربران را بر طرف کنند، به عبارت دیگر، طراحان شبکه با محدودیت های مانند: پیچیدگی ،سیاست های متناقض، فقدان مقیاس‌پذیری و وابستگی به فروشنده وجود نداشتن هماهنگی بین نیازهای بازار و قابلیت های شبکه، صنعت ITرا به‌سوی نقطه انحرافی می کشاند. برای جلوگیری از چنین رخدادی، صنعت معماری نرم‌افزار بنیان یا SDN را مطرح کرد و استانداردهای مرتبط با آن را توسعه داد. [2]

۲.۲ پروتکل OpenFlow

OpenFlow از اصل قاعده SDN، که جداسازی کنترل از حمل و نقل داده‌ها است، پیروی می‌کند. OpenFlow نحوه تبادل اطلاعات بین صفحه کنترل[۱] و صفحه داده[۲] را استانداردسازی کرده است. در معماری OpenFlow دستگاه حمل و نقل یا سوئیچ OpenFlow ، شامل یک یا چند جدول جریان[۳] و یک لایه انتزاعی است. برای ارتباط ایمن این دستگاه با کنترلر از پروتکل OpenFlow استفاده شده است. جداول جریان شامل مقادیر جریان هستند، که هرکدام از این مقادیر تعیین می‌کنند که چطور بسته‌های متعلق به یک جریان[۴] باید پردازش و ارسال شوند.

ارتباط بین کنترلر و سوئیچ بوسیله پروتکل OpenFlow شکل می‌گیرد. در این پروتکل مجموعه‌ای از پیام‌ها تعریف می‌شود که می توانند بین سوئیچ و کنترلر است روی یک کانال امن جابجا شوند. با استفاده از پروتکل OpenFlow می‌توان شبکه را کنترل کرد و مقادیر داخل جداول سوئیچ ها را حذف و اضافه و آپدیت کرد.. که این امر می تواند به صورت دو صورت واکنشی (یعنی در پاسخ به ورود یک بسته عملی انجام شود) و فعالانه[۵] انجام شود.

[1] Control plane

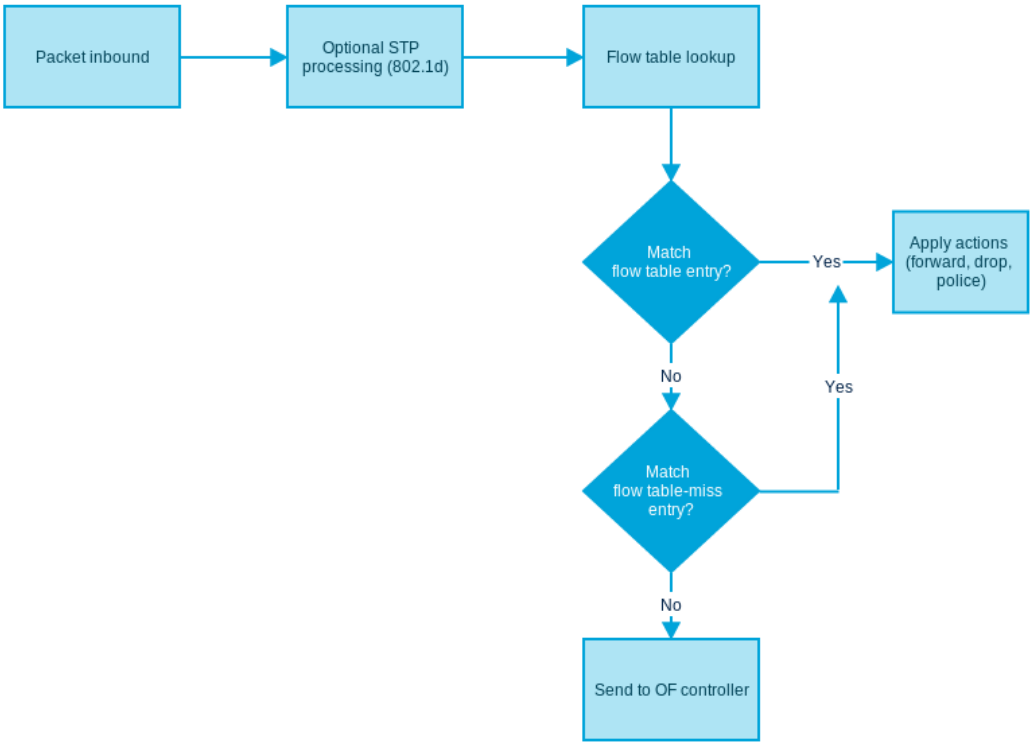
[2] Forwarding Plane

[3] Flow Table

[4] Flow

[5] passively

[6] proactively



شکل ۲-۱ flow-chart مربوط به تطابق بسته با سیاست‌ها

عناصر هر جریان معمولا شامل فیلدهای تطابق، شمارنده‌ها و مجموعه‌ای از دستورالعمل‌ها و اقدامات است.

۲.۲.۱ فیلدهای تطابق[۷]

برای تطابق بسته‌های ورودی استفاده می‌شود. فیلدهای تطابق ممکن است شامل اطلاعات پیدا شده در هدر بسته، پورت ورودی و یا metadata ها باشد. از metadata ها برای انتقال داده بین جداول جریان‌ها استفاده می‌شود.

[7] Match fields

۲.۲.۲ شمارنده‌ها

جهت جمع آوری آمار برای جریان خاص از این فیلد استفاده می‌شود. این آمار ها اطلاعاتی از قبیل تعداد بسته‌های دریافتی، تعداد بایت و مدت زمان وجود یک جریان در جدول جریان است.

۲.۲.۳ مجموعه‌ای از دستورالعمل‌ها و اقدامات[8]

به محض ورود بسته‌ها به سوئیچ OpenFlow، فیلدهای سرآیند[9]بسته‌ها دریافت می‌شوند و با فیلدهای موجود در جدول جریان‌های ورودی مطابقت داده می‌شوند. اگر یک ورودی تطابق پیدا کرد، سوئیچ یکسری دستورها و اقدام‌ها را روی آن اعمال می‌کند.

اگر ورودی‌ها با جدول جریان تطابق نداشتند، خود سوئیچ در برابر این ورودی عکس العمل نشان می‌دهد. این عکس العمل وابسته به دستورهایی است که از قبل برای جدول Table-miss تعریف شده است. هر جدول جریان باید شامل یک جدول برای عدم تطابق باشد. در این جدول برای ورودی‌های خاص که تطبیق داده نشده‌اند یکسری قوانین و عکس العمل ها تعریف شده است. از جمله این عملکردها: حذف کردن بسته ورودی، ادامه عمل تطابق روی جدول جریان بعدی و ارسال بسته به کنترلر با استفاده از پروتکل OpenFlow است.

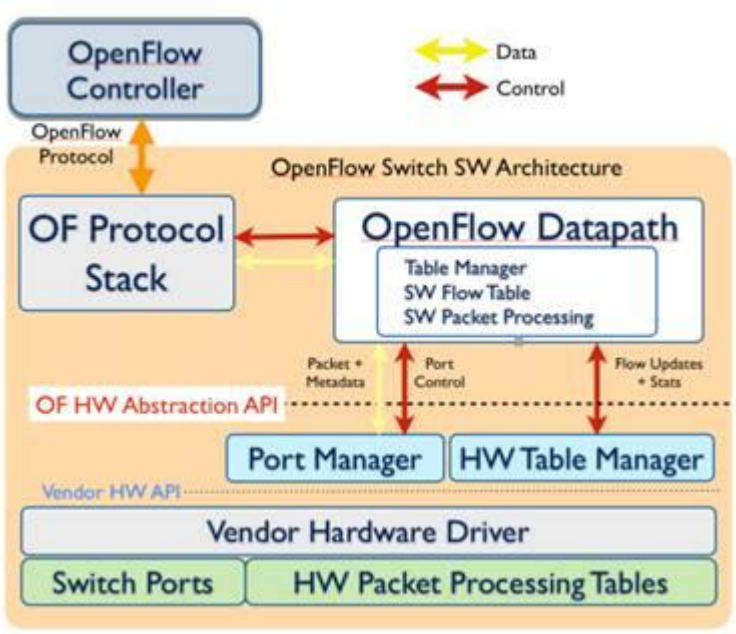
از نسخه ۱,۱ به بعد پروتکل OpenFlow، این پروتکل سیستم چند جدولی و پردازش خط لوله‌ای را پشتیبانی می کند. منظور از پردازش خط لوله‌ای تکنیکی برای تجزیه‌ی یک فرآیند ترتیبی به تعدادی زیرعمل است که هر زیر عمل در یک قطعه اختصاصی ویژه اجرا می‌شود.

[8] Actions

[9] Header

۲.۳ تجهیزات حمل و نقل

لایه زیرساخت شامل تجهیزات فیزیکی همچون روتر،سوئیچ، سوئیچ های مجازی، تجهیزات بیسیم و ... هستند که در حوزه SDN به همه آنها سوییچ OpenFlow گفته می‌شود.



شکل ۲-۲ معماری OpenFlow

تصویر فوق معماری سوییچ OpenFlow را نشان می‌دهد که لایه کنترل خارج از سویچ قرار دارد و کنترلر از طریق پروتکل OpenFlow با سویچ در ارتباط است. برای ارتباط بین کنترلر و سوئیچ پروتکل‌های دیگری هم پیشنهاد شده‌اند که در این بین پروتکل OpenFlow استاندارد شده است.

۲.۴ کنترلر SDN

ایده SDN، مزایای فراوانی نظیر انعطاف پذیری، قابلیت برنامه پذیری و تحقق یک شبکه متمرکز را به همراه دارد. علیرغم کلیه مزیت‌های فوق، همیشه نگرانی‌هایی درباره کارایی و مقیاس پذیری وجود داشته است.

مشکلات مقیاس پذیری:

یک کنترلر نمی‌تواند به همراه رشد شبکه (افزایش تعداد سوئیچ‌ها، مسیرها، پهنای باند و ...) مقیاس پذیری خود را حفظ کند. برای مثال اگر یک کنترلر فقط ۱۱۱ سوئیچ را قبول کند و ۱۱۲ سوئیچ وجود داشته باشد، باید به فکر حل این مشکل باشیم. اگر سرویس‌های شبکه رشد افزاینده‌ای داشته باشند، قاعدتا درخواست‌های زیادی به سمت کنترلر خواهند رفت و این ازدیاد می‌تواند کنترلر را دچار مشکل کند و اگر تمهیداتی اندیشیده نشود ممکن است درخواست‌های ورودی با شکست مواجه شوند. برای حل این مشکلات پیشنهاداتی داده شده است، مثلا: می توان بجای استفاده از یک کنترلر از چندین کنترلر، جهت کاهش تاخیر زمانی یا افزایش تحمل خطا استفاده کرد. سطح داده‌ایی و کنترلی تجزیه شده در SDN می‌توانند دارای توپولوژی‌ها و تکنولوژی‌های مختلفی باشند و به صورت مستقل از هم به کار گرفته شوند. همین نوع توپولوژی‌ها و چیدمان عناصر SDN می‌توانند در عملکرد و مقیاس پذیری شبکه نقش مهم و چشم گیری داشته باشند.

۳

فصل سوم

شرح مساله و مرور روش‌های طراحی

شرح مساله و مرور روش‌های طراحی

مساله‌ی اصلی سربار زیاد شبکه‌های نرم افزار بنیان است که عملا این گونه شبکه‌ها را در هسته‌ی اصلی شبکه بی‌ا ستفاده میکند. کارهای زیادی برای افزایش کارآیی این شبکه‌ها انجام شده ا ست که از میان آن‌ها ۲ مورد را که از روش مجازی سازی توابع شبکه ا ستفاده کرده‌اند به همراه اینجا مرور میکنیم. نکته مهم در اینجا این است که مجازی‌سازی توابع شبکه‌ای خود در کنار افزایش بهره‌بری شبکه‌های نرم افزار بنیان مشکل ایجاد سربار در سرورها را دارد. این بسترهای پیشنهادی در واقع مشکل سربار سرورها را حل کرده‌اند.

۳.۱ Devoflow

Devoflow یک افزونه‌ای ا ست که به OpenFlow ا ضافه شده و در شبکه‌های بزرگ با حجم ترافیک بالا استفاده می‌شود.

در این روش برای کاهش ســـربار flow-base switching از دو مکانیزم به نام‌های Rule cloning و Local actions استفاده می‌شود.

در مدل ا ستاندارد پروتکل OpenFlow اگر از قوانینی تحت عنوان wildcard ا ستفاده کنیم تا تقا ضاها به کنترلر را کاهش دهیم آنگاه تمام بسته‌هایی که با این قوانین هماهنگ هستند تحت عنوان یک Flow طبقه بندی می‌شـــوند. در مکانیزم اول devoflow بیان می‌شـــود که می‌توان در صـــورت نیاز قوانین wildcard را با پرچم CLONE مشـــخص کرده و به این ترتیب آنها را وادار کرد بســـته‌های هماهنگ با این قوانین را به Flowهای مختلف تقسیم کنند. برای این امر سوئیچ یک قانون جدید با مشخصات بسته ورودی هماهنگ با قانون wildcard ایجاد میکند و به این ترتیب ســـایر بســـته‌های این Flow با قانون جدید مطابقت پیدا میکنند.

مکانیزم دوم به زبان ساده بیان میکند که برای کاهش تقا ضاها به کنترلر می‌توان تعدادی از تصمیم‌ها را در سمت سوئیچ گرفته و سربار کنترلر را کاهش داد.

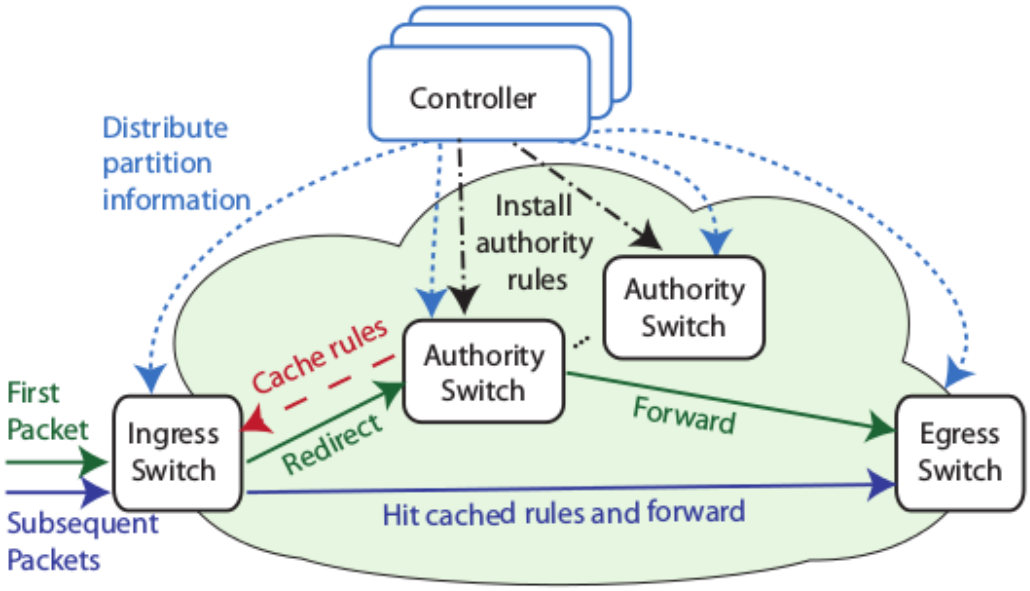
این روش به اینصورت عمل می‌کند که اگر Flow ورودی کوچک باشد از Devoflow استفاده می‌شود و اگر Flow بزرگ باشد از کنترلر استفاده می‌کند که این امر باعث میشود جریان‌های کوچک حافظه‌ای را اشغال نکنند و فضا را در اختیار جریان‌های بزرگتر قرار دهد.

۳.۲ DIFANE

در روش DIFANE، سوئیچ‌ها به سه دسته تقسیم می‌شوند:

[۱] Authority Switch, [2] Ingress Switch, [3] Egress Switch

ابتدا بسته‌ها به سوئیچ‌های گروه [۲] می‌رسند، اگر این گروه از سوئیچ‌ها قوانین لازم برای این بسته‌ها را داشتند، عملیات‌های لازم را انجام داده و بسته‌ها را به سمت سوئیچ‌های گروه [۳] می‌فرستند، در غیر این صورت بسته‌ها به سمت سوئیچ‌های گروه [۱] فرستاده می شوند تا عملیات‌های لازم توسط آن‌ها انجام شود، لازم به ذکر است این عملیات در سوئیچ‌های گروه [۲]، cache می‌شوند.



شکل ۳-۱ DIFANE

One Big Switch Abstraction

این پیشنهاد شامل دو روش است که مستقل از یکدیگر ارائه شده‌اند و هدف روش دوم بهبود روش اول است.

Palette پالت

One Big Switch یک سوئیچ بزرگ

Palette

مشکل نحوه توزیع قوانین بین سوئیچ‌ها است، هدف اصلی Palette این است که تا جایی که امکان دارد تعداد قوانینی که نیاز است در سوئیچ نصب شود را کم و قوانین را تا حد امکان بین همه سوئیچ‌ها توزیع کند و برای

بهینه کردن فرآیند قانون گذاری، سیاست‌ها را به دو دسته سیاست‌های endpoint و مسیریابی تقسیم می‌کند.

مفهوم Load Balancing مختصرا تقسیم کردن بار شبکه بین بخش‌های مختلف است.

سیاست‌های Endpoint

سیاست‌های Endpoint مانند: کنترل دسترسی و Load Balancing، شبکه را به صورت یک سوئیچ بزرگ میبینند که جزئیات معماری داخلی شبکه را پنهان میکند. یک سیاست بسته‌هایی که باید forward یا drop شوند در کنار تغییراتی که باید روی سرآیند بسته‌ها انجام شود را مشخص می‌کند.

سیاست‌های مسیریابی

از سوی دیگر، با استفاده از سیاستهای مسیریابی، مشخص می‌شود Flowها در شبکه باید از چه مسیری بروند. این سیاست‌ها عموما از الگوریتم‌های Traffic Engineering مشتق میشوند.

ایده اصلی Palette تقسیم بندی جدول سیاست‌های endpoint به جدول‌های کوچکتر و سپس توزیع آنها روی سوئیچ‌ها است.

که الگوریتم آنها شامل دو مرحله است:

۱. تعیین k، منظور از k، تعداد جدول است که مشخص می کنیم چه تعداد جدول مورد نیاز است.

۲. تقسیم قوانین روی k جدول

One Big Switch

در این روش م شکل تق سیم بهینه قوانین برای هر م سیر را حل می شود، به این ترتیب که هنگام توزیع قوانین روی سوئیچ‌ها با توجه به این مو ضوع که هر سوئیچ در چه م سیری دخالت دارد تنها قوانین لازم برای آن مسیرها را به آن سوئیچ می‌دهیم.

E2 ۳.۳

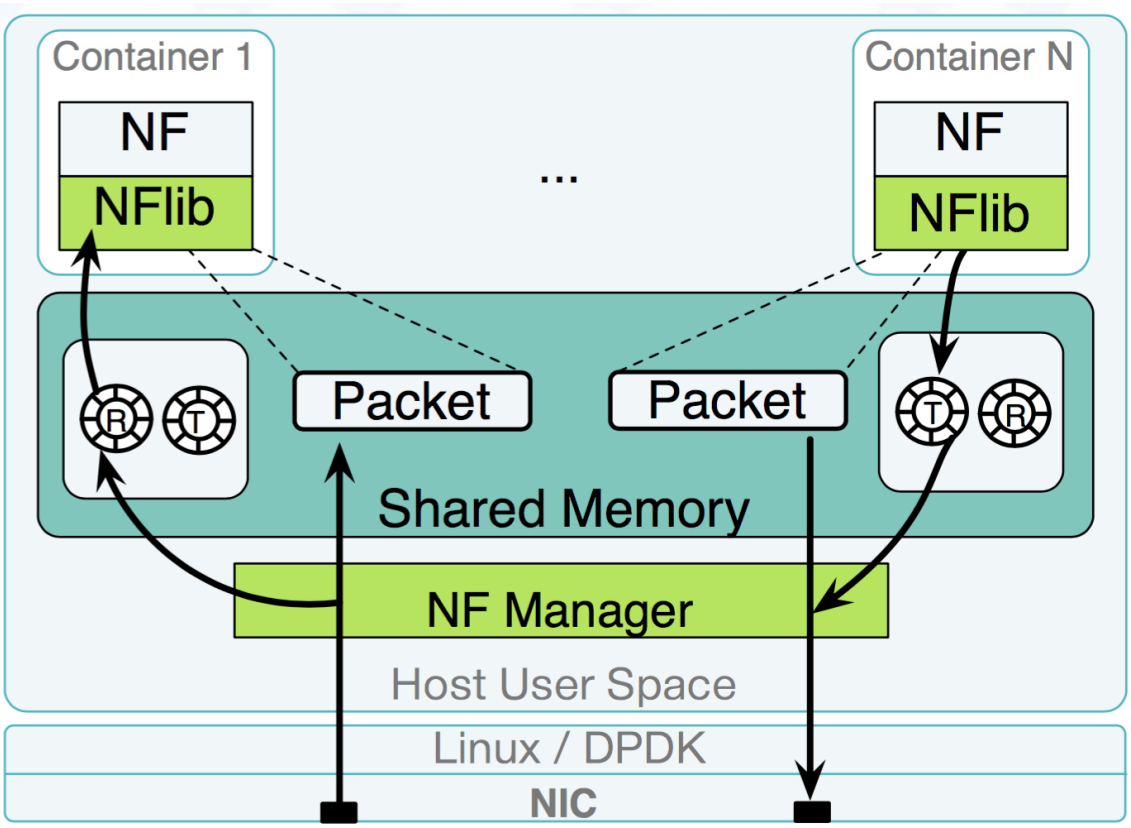
این پلتفرم که توسط دانشگاه برکلی ارائه شده است و در این بستر شما توابع شبکه‌ای خود را به صورت مجازی تعریف کرده و این بستر این توابع را با یک روش ریاضی از پیش اثبات شده به صورت بهینه بین سرورهای فیزیکی توزیع میکند، در این توزیع سعی می شود بار سرورها به حداقل بر سد. این امر باعث کاهش سربار سرورها و در نهایت افزایش کارآیی شبکه میگردد.

ClickOS ۳.۴

این پلتفرم که جد اصلی روش حل ما تلقی میشود توابع شبکه را به صورت مجازی روی بستر KVM و روی سیستم عامل لینوکس اجرا میکند. این بستر مدیریت بر روی توابع شبکه‌ای را راحتتر میکند ولی در مورد بهبود پردازش بسته‌ها صحبتی نمیکند.

NetVM ۳.۵

این بستر که راه کار پیشنهادی ما می‌باشد، در ادامه‌ی کار ClickOS می‌باشد، در اینجا از Docker به جای KVM استفاده کرده‌ایم که بهبود چشمگیری در مجازی سازی توابع شبکه حاصل میکند و مدیریت‌هایی که بستر ClickOS فراهم آورده بود را نیز همچنان پشتیبانی میکند.



شکل ۳-۲ معماری بستر NetVM

۴

فصل چهارم

شرح روند پیاده‌سازی

شرح روند پیاده‌سازی

برای پیاده سازی این بستر از DPDK و بستر مجازی‌سازی Docker استفاده شده است. توابع شبکه‌ای روی Docker مجازی سازی شده و بسته‌ها به وسیله‌ی DPDK پردازش شده و به دست توابع شبکه‌ای می‌رسند. از آنجایی که پیاده سازی این طرح در مراح اولیه می‌باشد هنوز توابع شبکه‌ای پیچیده‌ای روی این بستر تست نشده‌اند، تنها تا به الان یک سوئیچ لایه ۲ و یک Bridge روی این بستر پیاده سازی شده‌اند.

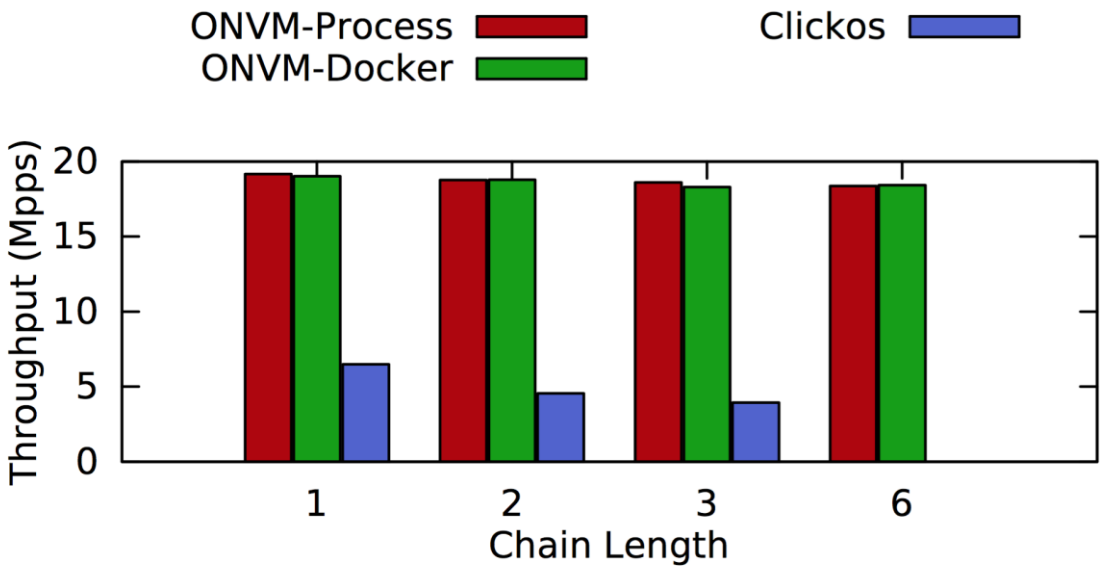
۵

فصل پنجم

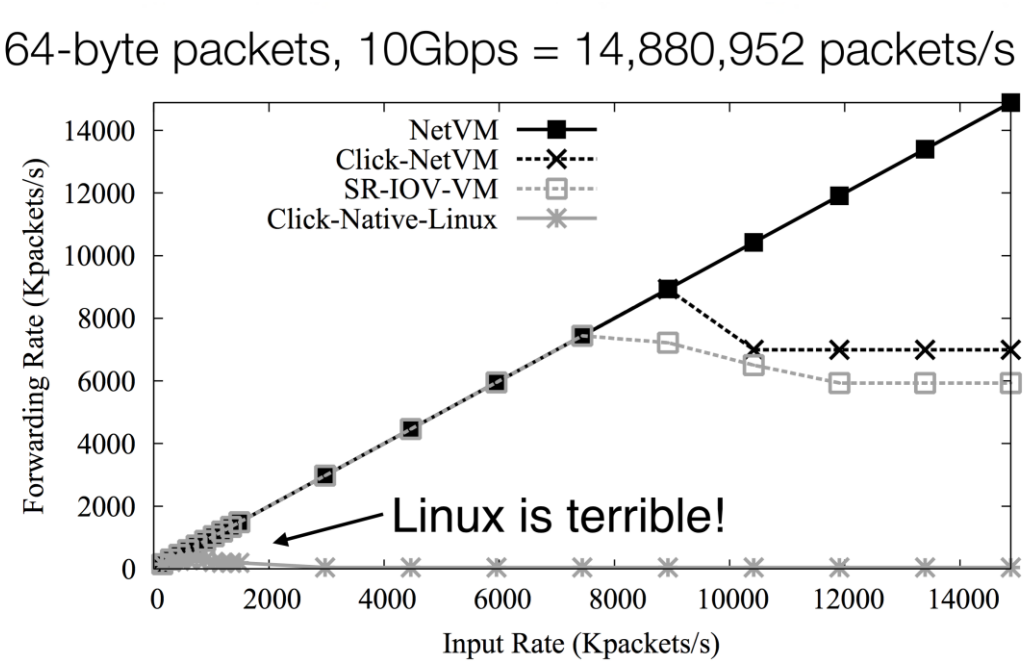
آزمایش و بررسی نتایج

آزمایش و بررسی نتایج

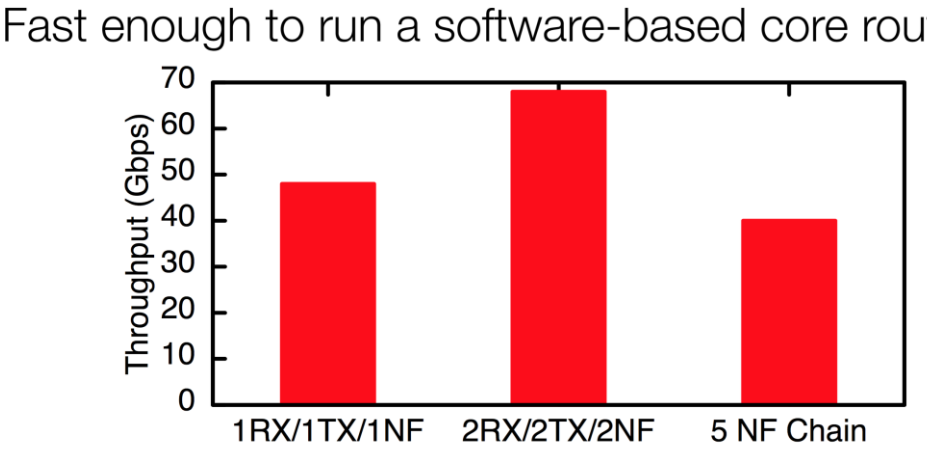
در این بستر با توجه به استفاده از DPDK برای پردازش بسته‌ها بدون نیاز به دخالت هسته سیستم عامل لینوکس و استفاده از بستر مجازی‌سازی Docker برای مجازی سازی توابع شبکه‌ای افزایش کارآیی خوبی نسبت به مدل‌های پیشین به ویژه بستر ClickOS حاصل شده است. در ادامه این نمودارهایی برای مقایسه گذردهی این بستر NetVM با ClickOS و سیستم عامل لینوکس (بدون هیچ نرم افزار اضافه‌ای برای فراهم آوردن بستر مجازی‌سازی و ...) آورده شده است.



شکل ۵-۱ مقایسه گذردهی بین بستر ClickOS و NetVM



شکل ۵-۲ مقایسه قدرت پردازش بسته‌ها بین NetVM و ClickOS و Linux



شکل ۵-۳ گذردهی NetVM در حالت‌های مختلف از اجرای NFها

۶

فصل ششم

نتیجه‌گیری و کارهای آینده

نتیجه‌گیری و کارهای آینده

در این بستر از مجازی سازی توابع شبکه‌ای برای بهبود کارآیی شبکه‌های نرم افزار بنیان استفاده شده است، این در حالی است که مجازی سازی توابع شبکه تنها راه کار نیست و تحقیقات گسترده‌ای برای افزایش کارآیی شبکه‌های نرم افزار بنیان از راه کارهای مختلف در جریان است، حتی با استفاده از مجازی سازی توابع شبکه هم هنوز این راه حل بهترین راه حل ممکن نبوده و تحقیقات بسیاری در این زمینه در جریان است.

منابع و مراجع

[۱] J. Hwang, K. Ramakrishnan, and T. Wood. NetVM: High Performance and Flexible Networking using Virtualization on Commodity Platforms. In Symposium on Networked System Design and Implementation, NSDI 14, Apr. 2014.

[۲] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. ClickOS and the Art of Network Function Virtualization. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), pages 459–473, Seattle, WA, Apr. 2014. USENIX Association.

[۳] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker. E2: A Framework for NFV Applications. In Proceedings of the 25th Symposium on Operating Systems Principles, SOSP '15, pages 121–136, New York, NY, USA, 2015. ACM.