:BeaQoS

تعادل بار و مدیریت مهلت صف در یک سوئیچ OpenFlow SDN

چکیده

خصوصیات OpenFlow فعلی قادر به تنظیم نرخ سرویس صفها درون دستگاههای OpenFlow نیست. این کمبود اجازه نمی دهد اکثر الگوریتم ها برای رضایت از الزامات کیفیت خدمات به جریانهای جدید و پایدار اعمال شود. در این مقاله ما راه حل دیگری را پیشنهاد می کنیم که از طریق تعدادی اصلاحات روی Beacon، یکی از کنترل کننده های محبوب SDN، اجرا می شود. این کار به شرح زیر است: با استفاده از آمار تقریبا در زمان واقعی از دستگاه های OpenFlow، Beacon مسیرهای مجازی را در صف های مختلف به منظور تضمین نقاط ضعف مهلت مجاز (برای مثال، جریان هنوز مفید است اگر و تنها اگر به طور کامل توسط یک زمان معین دریافت می شود) و / یا متعادل کننده صف عملکرد در یک سوئیچ OpenFlow SDN. برخلاف پیشینه، ما هیچ پایه ای یا اصلاح جدیدی از استاندارد OpenFlow را پیشنهاد نمی دهیم: مکانیزم ما، اجرا شده در کنترلر، با دستگاه های OpenFlow منظم کار می کند. تغییرات ما در کنترل کننده SDN پایه ای برای طراحی یک کلاس از الگوریتم های جدید مسیر مسیریابی است که می تواند محدودیت های مهلت و متعادل سازی صف را بدون هیچ گونه اصلاح مشخصات OpenFlow و همچنین دستگاه های Open Flow تضمین کند.

کلید واژه ها: SDN. OpenFlow . از دست دادن بسته. مهندسی ترافیک

1. مقدمه

شبکه های تعریف شده نرم افزاری (SDN) با ایجاد قابلیت برنامه ریزی، مدیریت آسان و نوآوری سریع تر، صنعت شبکه سازی را متحول می سازد [1،2]. این مزایا توسط معماری کنترل متمرکز که به شبکه اجازه می دهد تا توسط یک مرکز برنامه ریزی و کنترل شود، ممکن می‌شود.

معماری SDN هر دو از دستگاه های فعال SDN (سوئیچ ها / روتر ها) 1 و از یک کنترلر مرکزی (کنترل کننده SDN) تشکیل شده است. یک دستگاه SDN پردازش و ارائه بسته ها را با توجه به قوانین ذخیره شده در جدول جریان خود (حالت روبه جلو)، در حالی که کنترل کننده SDN وضعیت رو به جلو هر دستگاه SDN با استفاده از یک پروتکل استاندارد به نام OpenFlow (OF) [2] را تنظیم می کند. کنترل کننده SDN همچنین مسئول ساخت توپولوژی مجازی نشان دهنده توپولوژی فیزیکی است. توپولوژی مجازی توسط ماژول های نرم افزاری که در سطح بالای کنترل کننده SDN اجرا می شود برای اجرای منطق های مختلف کنترل و توابع شبکه (مانند مسیریابی، ترافیک، اقدامات فایروال) استفاده می شود.

در حال حاضر مدیریت کیفیت خدمات (QoS) در OF کاملا محدود است: در هر سوئیچ OF یک یا چند صف برای هر یک از خروجی ها تعریف می شود و برای نگاشت جریان های ورودی روی آنها استفاده می شود. ورودی های جریان داده شده به یک صف خاص با توجه به پیکربندی صف از نظر نرخ سرویس محاسبه خواهد شد، اما پیکربندی صف در خارج از پروتکل OF انجام می شود. برای مثال، نرخ خدمات صف نمی تواند توسط OF تغییر کند.

فرض کنید که جریان یک زنجیره صف از منبع به گره مقصد را می گذراند و سرعت انتقال داده ها افزایش می یابد، ممکن است نتیجه این باشد که صف ها اشغال را افزایش می دهند و ممکن است یک مشکل با پیچیدگی شبکه همراه باشد. عدم امکان تغییر نرخ تضعیف صف از طریق دستورالعمل های OF زمان واقعی می تواند منجر به کاهش شدید عملکرد جریان هایی که در این صف قرار درند بشود، زیرا بدون تخصیص نرخ مناسب، تضمین کیفیت خدمات مورد نیاز برای جریان ها بسیار مشکل است [3].

یک راه حل ممکن برای کاهش تضعیف عملکرد، شامل مسیریابی مجدد جریانهایی است که نقض محدودیتهای زمانی (مثلا جریانهایی که به طور کامل از محدودیت زمان ثابت دریافت شده است) [4] در مسیرهای کمتر یا صفهای کمتری دریافت میکنند. ایده اصلی این است که، چون ما نمی توانیم نرخ سرویس صف ها را تغییر دهیم، ما بر ترافیک ورودی عمل می کنیم، در صورت نیاز به یک زیر مجموعه از جریان ها در مسیرهای مختلف یا صف ها حرکت می کنیم. به منظور سازگاری 100٪ با سخت افزار فعلی OF، ما هیچ تغییری در مشخصات و دستورالعمل های OF اعمال نمی کنیم. در عوض ما پیشنهاد می کنیم یک کنترل کننده محبوب SDN را تغییر دهیم: Beacon [5]. راه حل پیشنهادی، BeaQoS، به یک سوئیچ تک SDN اعمال شده است، گسترش کار قبلی ما است که در [6] ارائه شده است. کنترلر جدید به روز شده ما آمار مربوط به صف ها، جریان ها و پورت های سوئیچ های OF را دریافت می کند و تخمین سرعت جریان و از دست دادن بسته صف ها را محاسبه می کند. بر اساس سیاست های قابل تنظیم، BeaQoS قادر خواهد بود زیر مجموعه ای از جریان هایی را که از صف مشکل دار عبور می کنند را انتخاب کند و آنها را دوباره به صف دیگر و با پیچیدگی کمتر هدایت کند، بنابراین عملکرد سوئیچ هابهبود می یابد. عمل جابجایی مسیر جریان ممکن است نه تنها برای مدیریت زمان پایان، بلکه همچنین برای بارگیری یکنواخت کارآمد صف، مورد بهره برداری قرار گیرد. از سوی دیگر، تعادل بار، اغلب به عنوان یک اقدام برای جلوگیری از پیچیدگی و به تبع آن، محدود کردن و به تأخیر افتادن عملکرد به حساب می آید.

باقی مانده از این مقاله به شرح زیر است. ما بخش های مربوط به این زمینه را در بخش 2 شرح دهیم. با توجه به سهم اصلی این مقاله:

ما انگیزه هایی را بیان می کنیم که منجر به بررسی رابط های چند صف با نرخ سرویس متغیر برای پشتیبانی مدیریت زمان پایان در بخش 3 می شود.

ما ایده اولیه ای را درباره مکانیزم های مسیریابی مجدد معرفی شده در این مقاله در بخش 4.1 توضیح می دهیم، جایی که ما همچنین نشان می دهیم چگونه می توان آن را در مورد معماری های چند هسته ای و مسائل متعادل کننده بار در میان صف ها مورد استفاده قرار داد؛

ما تغییرات کنترل کننده Beacon مورد نیاز برای پیاده سازی دوباره مسیر در بخش 4.2 را شرح می دهیم.

- ما پنج راهکار مسیریابی مجدد در BeaQoS پیشنهاد می دهیم: دو مورد از آنها برای بهبود مدیریت زمان پایان پیشنهاد شده است و سه مورد از آنها با هدف تعادل بار در میان صف در یک سوئیچ SDN در بخش 5 انجام شده است.

ما تجزیه و تحلیل عملکرد الگوریتم های پیشنهادی خودمان را در بخش 5 نشان می دهیم. ما در مورد نتایج به دست آمده با نتیجه گیری در بخش 7 گزارش می دهیم.

2. کارهای مربوطه

علی رغم تکنیک های ترافیکی (TE) که اغلب توسط MPLS-TE [7،8] کنترل می شود، توانایی کنترل کننده SDN برای دریافت (نرم) اطلاعات در زمان واقعی از دستگاه های SDN و تصمیم گیری بر اساس یک دید جهانی از شبکه ، همراه با توانایی تجمع جریان سفارشی در داخل دستگاه های SDN، TE را یکی از جالب ترین موارد استفاده برای شبکه های SDN می کند.

الگوریتم های متعادل کننده بار جهانی در [9] پیشنهاد شده است که حاوی متعادل کننده بار به عنوان یک جزء جدایی ناپذیر از سرویس های بزرگ cloud است و به دنبال روش هایی را برای ایجاد مقیاس پذیری، پویا و انعطاف پذیری متعادل کننده بار است. علاوه بر این [9] بیان می کند که تعادل بار بایستی اساس شبکه باشد، نه یک افزودنی، و یک نمونه اولیه متعادل کننده بار توزیع شده بر اساس این اصل ارائه می دهد.

[10]، نشان می دهد که کنترل کننده باید سوئیچ پشتیبانی از قوانین wildcard را برای یک راه حل مقیاس پذیر تر که توده های بزرگی از ترافیک مشتری را به کپی های سرور هدایت می کند، سوق دهد. [10] همچنین الگوریتم هایی را ارائه می دهد که قوانین مختصر wildcard را محاسبه می کنند که به توزیع هدف ترافیک دست می یابند و به طور خودکار سیاست های تعادل بار را بدون وقفه در اتصالات موجود تغییر می دهند. علاوه بر این، نویسندگان

این الگوریتم ها را بر روی کنترل کننده OpenFlow NOX اعمال می کنند، اثربخشی آنها را ارزیابی می کند و راه هایی را برای تحقیق بیشتر پیشنهاد می دهد.

کار ارائه شده در [11] یک سیستم را نشان می دهد که به طور مرکزی ترافیکی را که در هر سرویس به مرکز داده های اتصال متقابل ارسال می کند کنترل می کند تا صفحه اطلاعات شبکه را دوباره تنظیم کند و در نتیجه مطابق با نیازهای ترافیکی فعلی باشد. [11] تکنیک جدیدی را ایجاد می کند که مقدار کمی از ظرفیت آسیب را برای لینک ها به کار می گیرد تا به روز رسانی را در یک ازدحام آزاد قابل اثبات انجام دهد، بدون اینکه هیچ فرضی درباره ترتیب و زمان به روز رسانی در هر سوئیچ انجام شود. علاوه بر این، برای مقیاس دادن به شبکه های بزرگ در مواجهه با ظرفیت جدول انتقال محدود، [11] به طور دلخواهی مجموعه ای از مقادیری را انتخاب می کند که می تواند تقاضای فعلی را برآورده کند و این مجموعه را بدون برهم زدن ترافیک به روز کند.

جدول1 واحدهای عملکرد ترافیک برای پیکربندی های 1صف و 3صف

Performance metric	Queue configuration	
	1-queue	3-queue
BF - packet loss	25%	71.16%
DF1 - percentage of flows matching the deadline	11.43%	74.29%
DF2 - percentage of flows matching the deadline	17.39%	19.57%

مرجع [12] یک شبکه جزئی SDN (ترکیبی از دستگاه SDN و غیر SDN) را تجزیه و تحلیل می کند و نشان می دهد که چگونه از کنترل کننده متمرکز بهره مند شود تا بهبود قابل توجهی در استفاده از شبکه و همچنین کاهش تلفات بسته و تاخیر انجام شود. [12] نشان می دهد که این پیشرفت ها حتی در مواردی امکان پذیر است که تنها در بکار گیری از جزئی از یک شبکه قابلیت انعطاف پذیری SDN وجود دارد. مدیران مشکل بهینه سازی کنترل کننده SDN را برای مهندسی ترافیک با استقرار جزئی فرمول بندی می کنند و برای حل آن برنامه های سریع تقریبی چندجمله ای (FPTAS) را پیشنهاد می کنند.

این آخرین مشکل نیز در [13] مطرح شده است که یک روش مدیریت ترافیک را برای تقسیم کردن یا «بریدن» منابع شبکه به منظور مطابقت با نیازهای کاربر معرفی می کند. [13] یک جایگزین برای ترتیب بندی مکانیزم های

سطح پایین مانند شبکه های محلی مجازی، یا قرار دادن هیپنوایزم های کامپیوتری در کنترل کننده را فراهم می کند، با معرفی مقادیری که از برنامه نویسی برش های ایزوله شبکه پشتیبانی می کند. معنی برش ها تضمین می کند که پردازش بسته ها بر روی یک تکه مستقل از تمام برش های دیگر است. آنها تعریف چکیده برش خود را تعریف می کنند، الگوریتم هایی را برای کامپایل قطعات ایجاد می کنند و استفاده از آنها را با استفاده از مثال ها نشان می دهد. علاوه بر این، [13] یک برنامه نمونه برداری و یک ابزار برای بررسی خودکار ویژگی های ایزوله رسمی را توصیف می کند.

در کار قبلی ما [6]، ما یک راه حل مبتنی بر SDN پیشنهاد کردیم که استراتژی نرم افزاری را برای مقابله با جریان های غیریکنواخت ترافیکی داخل یک سیستم مبتنی بر کلاس پیشنهاد می کند. بنابراین این رویکرد مستقل از سخت افزار پایه ای است، زیرا که آن را به عنوان یک الگوریتم درون کنترل کننده SDN اجرا می کند. استراتژی پیشنهادی، جریانهای غیر سازگار را براساس مجموعه ای از داده های آماری که توسط یک نسخه اصلاح شده کنترل کننده Beacon جمع آوری شده، به منظور کاهش تضعیف کیفیت جریان های عبور از شبکه مدیریت می کنند.

برای پشتیبانی از ترافیک مهندسی در محیط SDN، پروتکل مدیریت و پیکربندی OpenFlow (OF-Config) پیشنهاد شده است. OF-Config [14] یک پروتکل است که توسط بنیاد باز شبکه ای که برای مدیریت سوئیچ های فیزیکی و مجازی در یک محیط OpenFlow توسعه داده شده است، ایجاد شده است. این ابزار به مهندسین شبکه یک نمای کلی از شبکه نشان می دهد و همچنین توانایی تنظیم سیاست ها را فراهم می کند و مدیریت ترافیک را در سراسر دستگاه فراهم می کند.

3. انگیزه

بعضی از رویکردها یک صف را برای هر ورودی خروجی در نظر می گیرند. به منظور حمایت از مکانیزم های QoS و تفاضل ترافیک، معمول است که چندین صف را تنظیم کنند [3]. اهمیت تمایز ترافیک توسط اولین گروه از شبیه سازی ها (جدول 1) در موارد زیر گزارش شده است.

ورودی های جریان داده شده به یک صف خاص به ترتیب با توجه به پیکربندی صف از نظر نرخ خدمات مطابقت خواهند داشت. اکثر رویکردهای ذکر شده در بالا، توانایی SDN / OpenFlow را برای تنظیم نرخ سرویس صفها در هر دستگاه SDN فرض می کنند. این شانس برای بهبود عملکرد سوئیچ SDN بسیار مفید خواهد بود، همانطور که از گروه دوم شبیه سازی ها (جدول 2) در زیر گزارش شده است.

جدول2 واحدهای عملکرد ترافیک برای نرخ سرویس ثابت و متغیر

Performance metric	Queue configuration	
	Fixed rate	Variable rate
BF - packet loss	0%	0%
DF1 - percentage of flows matching the deadline	100%	100%
DF2 - percentage of flows matching the deadline	25%	100%
DF - percentage of flows matching the deadline	34.78%	100%

جدول3 کلاس های ترافیک و نیازمندی های زمان پایان آن ها

Traffic class		Percentage of	Deadline
Name	Traffic descriptor	overall traffic	requirements
BF	$50 - 80\,\text{kbit/s} \times 50\,s$	30%	–
DF1	$4.5\,\text{Mbit/s} \times 1\,s$	55%	deadline: $9\,s$
DF2	$1.5\,\text{Mbit/s} \times 1\,s$	15%	deadline: $5\,s$

جدول 1 نتایج شبیه سازی هایی را که ما برای نشان دادن چگونگی سختی آن، بدون تمایز ترافیک، به منظور تامین نیازهای زمان پایان انجام دادیم نشان می دهد. در طول 120 ثانیه از شبیه سازی، یک Open vSwitch 2 s 1 ترکیبی از ترافیک تولید شده با iperf را تهیه می کند که حاوی "جریان های پس زمینه" (BF) و "جریان های زمان پایان" 3 (DF) هستند. BF جریان CBR با نرخ به طور تصادفی در مجموعه {50، 60، 70، 80} kbit / s انتخاب شده است. DF به دو کلاس تقسیم می شوند: DF1 و DF2 .DF1 دارای مهلت 5 ثانیه است، در حالی که DF2 یک محدودیت 9 ثانیه ای است. توصیفگرها و الزامات ترافیک کلی در جدول 3 تعریف شده است.

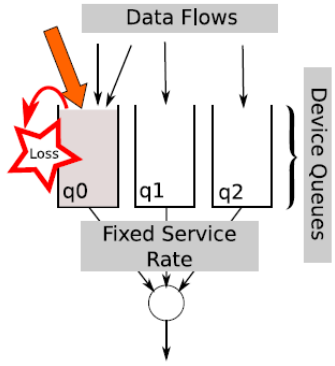
ما دو پیکربندی را با استفاده از 125 جریان تولید ایجاد کردیم. در اولین 1 s دارای یک صف در رابط خروجی (q 0) با FIFO (اولین ورودی، خروجی اول) نرخ خدمات sq 0 = 3 مگابیت در ثانیه، در حالی که در دومی 3 صف،

هر یک از آنها اختصاص داده شده به ترافیک مشخص: 0 q برای BF، 1 q برای DF1 و 2 q برای DF2. نرخ خدمات صف ها (مجموعه ای از پیش تعیین شده) s q 0 = 300 kbit / s، sq1=1.7 مگابایت بر ثانیه است. sq2 1 = مگابیت بر ثانیه. معیارهای QoS که در اینجا مورد بررسی قرار می گیرند، میزان تلفات بسته در درصد برای BF و درصد جریان های مطابق با مهلت برای DFها است که در جدول 1 نشان داده شده است.

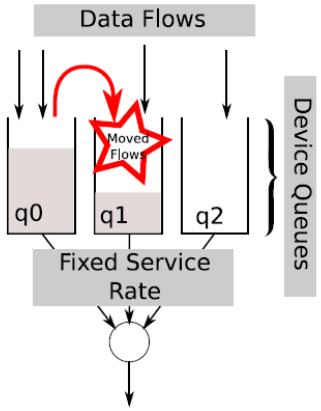
همانطور که می توان از پیکربندی 3 صفی به طور کلی درصد جریان های مطابق با زمان پایان را ثابت می کند و میزان تلفات بسته های BF را جبران می کند. تنظیم نرخ های خدمات صف های ساده متفاوت است، عملکردها تغییر خواهد کرد، اما مشخص است که تمایز ترافیک از طریق اینترفیس های چند صفی، ابزارهای اساسی برای مدیریت جریان زمان پایان و تنظیم میزان عملکرد ترافیک شبکه می دهد.

جدول 2 نتایج دومین مجموعه ای از شبیه سازی هایی را که اجرا کردیم نشان می دهد که هدف از آن ها نشان دادن چگونگی و توان تغییر نرخ خدمات صف ها است که می تواند عملکرد اجرایی مدیریت زمان پایان را بهبود بخشد. همانند شبیه سازی قبلی، 1 s ترکیبی از ترافیک از BF، DF1 و DF2 را دریافت می کند. دوباره دو پیکربندی با همان تعداد جریان تولید شده آزمایش می شوند. در تنظیمات اول s1 دارای 3 صف با نرخ سرویس پیش ثابت شده: sq 0 = 2 مگابیت بر ثانیه، sq 1 = 4 مگابیت در ثانیه و sq 2 = 4 مگابیت در ثانیه، در حالی که در دومین مورد، q1 می تواند ظرفیت اضافی را از دو طرف دیگر زمانی که نیازمند پهنای باند بیشتری است، بگیرد: sq 1 در محدوده [4 -10] Mbit / s است.

پیکربندی نرخ متغیر به طور مداوم درصد کل جریانهایی را که با زمان پایان مطابقت دارند، بدون هیچ گونه تأثیری در از دست دادن بسته BF به حداکثر 100٪ افزایش می دهد. در جدول 2، برچسب DF روی جریان های زمان پایان را بدون تمایز بین DF1 و DF2 زده می شود.



شکل 1 بارگیری در یک صف



شکل 2 مسیریابی مجدد بعضی از جریان ها

متاسفانه، همانگونه که در مقدمه ذکر شده، مشخصات فعلی OF [16] قادر به تنظیم نرخ سرویس صف نیست و این وظیفه را به یک پروتکل پیکربندی اختصاصی داده است: "پیکربندی صف در خارج از پروتکل OpenFlow انجام می شود یا از طریق یک ابزار خط فرمان یا از طریق یک پروتکل تصحیح اختصاصی. "([17]، بخش 7.3.5.10). به عنوان یک نتیجه، این مقاله، حتی اگر چندین صف برای تمایز ترافیکی اعمال شود، فرض می کند که نرخ سرویس صف در مجموعه سوئیچ SDN تنظیم شده و غیر قابل تغییر است.

4. راه حل های ممکن و اصلاح Beacon مورد نیاز

4.1. ایده کلی

اگر چه طراحی و پیاده سازی یک دستورالعمل جدید OpenFlow که قادر به تنظیم نرخ سرویس صف ها می باشد، بهترین راه حل از نظر عملکرد است، این انتخاب با یک اشکال اصلی مواجه می شود: این کاملا با سوئیچ های فعلی که نمی خواهند هر گونه مزایایی از این مقررات دریافت کنند ناسازگار است.

به همین دلیل ما یک راه حل جایگزین را پیشنهاد می کنیم که کاملا سازگار با جریان های سوئیچ ها است. ایده اساسی در شکل ها نشان داده شده است. 1 و 2

بگذارید فرض کنیم که در طول عملیات شبکه، سوئیچ OF در شکل 1، 5 جریان را دریافت می کند که از طریق 3 صف خروجی 0 q، 1 q و 2 q کنترل می شود. فرض کنیم که جریان نارنجی (یعنی بزرگترین فلش) سرعت داده خود را افزایش می دهد به طوری که 0 q بسته های بیشتری را دریافت می کند تا آن ها بتوانند از آنها استفاده کنند. 0 q نرخ ورودی بالاتر از نرخ سرویس از قبل تنظیم شده است. در این وضعیت، افزایش نرخ ورودی در نهایت منجر به از دست دادن بسته و کاهش شدید کیفیت با تجربه جریان 0 در q می شود. اگر قادر به تغییر نرخ سرویس صف نباشد، یک راه حل ممکن مسیریابی برخی جریانهای ورودی 0 q به صف دیگر (به عنوان مثال 1 q در شکل 2) به منظور کاهش 0 q نرخ ورودی است. مکانیزم های مسیریابی مجدد تلاش می کنند تا از پهنای باند اضافی استفاده نشده توسط صف های دیگر استفاده کنند.

از آنجا که ما می خواهیم هر دو سوئیچ ها و مشخصه ها را ساده نگه داریم، ما مکانیزم های مسیریابی مجدد را در کنترل کننده SDN طراحی و پیاده سازی می کنیم. حتی اگر این ایده ساده باشد، طراحی مکانیزم های مسیریابی شامل ویژگی های کنترل کننده SDN و به ویژه ویژگی ها یا الزامات زیر است:

سازگاری با نسخه های اولیه OpenFlow (که مسلما باید باشد)؛

- ایجاد یک ماژول قادر به اداره آمار؛

- اجرای رویکردهای پیشنهادی؛

- هیچ آماده سازی اولیه ای نباید با توجه به استاندارد OpenFlow فعلی اصلاح شود.

ایده مسیر مسیریابی مجدد و استراتژی های پیشنهادی در این مرحله می تواند برای اهداف مدیریتی خاص و در راستای مدیریت بهینه منابع سخت افزاری ارائه شده به مسیر یاب های نرم افزاری معمول مورد بهره برداری قرار گیرد. مسیریاب های نرم افزاری می توانند بر روی CPU های معمولی و سخت افزاری اجرا شوند، نه بر روی سخت افزار اختصاصی گران. این سخت افزار نه تنها سطح بالایی از قابلیت برنامه نویسی و انعطاف پذیری را حفظ می کند بلکه از راه حل های سخت افزاری تخصصی و اجزای شبکه ارزان تر است. به همین دلیل، مسیریاب های نرم افزاری تا حد زیادی گسترده هستند [18]. از سوی دیگر، ثابت شده است که CPU مشکل اصلی در مسیریاب نرم افزار است. پیشرفت های اخیر پیشنهاد می دهند که عملکرد پردازش بسته را از طریق پردازش موازی بر اساس پردازنده های چند هسته ای غیرفعال افزایش دهد [19]. به طور دقیقتر، مسیریاب های نرم افزاری فعلی برای چندین صف NIC ها (کنترل کننده های رابط شبکه) مورد استفاده قرار می گیرند تا بسته های ورودی را به یک صف خاص بر اساس ویژگی های بسته خاص مورد استفاده قرار دهند. با استفاده از این فیلترها، NIC ها قادر به بارگذاری کارآیی پردازش بسته های دریافتی در هسته های CPU می باشند. این همچنین تضمین می کند که هر یک از بسته های یک جریان خاص با همان هسته CPU خدمت می کند، به طوری که، برای مثال، تنظیم مجدد بسته [20] اجتناب شود.

به جای استفاده از فیلترهای سخت افزاری اختصاصی توسط NIC ها، ما یک راه حل انعطاف پذیر را بر اساس معماری OpenFlow پیشنهاد می کنیم. رویکرد ما شامل استفاده از کنترل کننده نرم افزاری OF می شود که می تواند جریان های ورودی را کنترل کند و اطلاعات را برای تصمیم گیری در مورد استراتژی صف بندی درست استفاده کند. ما مجموعه ای از الگوریتم های کنترل را قادر می سازیم که دوباره جریان را تنظیم کنند تا سبب کاهش تقارن CPU توسط توزیع جریان در میان صف های موجود شود.

4.2. پیاده‌سازی BeaQos

ما Beacon [5] را به عنوان کنترل کننده SDN انتخاب کردیم. Beacon یک کنترل کننده مبتنی بر چند رشته مبتنی بر جاوا است که به چارچوب OSGi و Spring متکی است و به Eclipse IDE ادغام شده است. در هر صورت، مستقل از انتخاب خاص کنترل کننده، تغییرات ما را می توان در هر کنترلر اجرا کرد. ساختار کنترل کننده شامل یک گروه از توابع (بسته های نامیده می شود) با ویژگی های اختصاصی است. بسته اصلی که روی آن متمرکز بودیم مسیریابی است که برای پیدا کردن راه صحیح بین منبع و مقصد برای انتقال بسته ها در نظر گرفته می شود. علاوه بر این، ما یک بسته نرم افزاری ad-hoc ایجاد کردیم، به نام Statistics، با هدف جمع آوری و پردازش آمار پیام های پاسخی که توسط سوئیچ های شبکه ارائه می شود. اصلاحات اصلی پیشنهاد شده از Beacon عبارتند از:

جدول4 آمار BeaQos در مقایسه با آمار OpenFlow 1.0

Statistics available in OpenFlow 1.0		Statistics computed by BeaQoS
Tx Bytes per Flow	→	Estimated Rate per Flow
Tx Bytes per Port	→	Estimated Rate per Port
Tx Bytes per Queue	→	Estimated Rate per Queue
FlowMatch, FlowActions, QueueID	→	Flows per Queue

کنترل کننده آمار Beacon به منظور ارسال درخواست های آماری به سوئیچ ها اصلاح شده است. ما یک عملکرد اضافه کردیم که ارسال پیام های آمار و ویژگی را با یک فاصله کنترلی (PI) قابل تنظیم از طریق یک فایل خواص خارجی آغاز می کند. ما همچنین یک کلاس اختصاص داده شده به ایجاد پیام های آماری، مانند of__flow_stats_request، ofp_port_stats_request، ofp_queue_stats_request [21] طراحی و اجرا کردیم تا اطلاعات مفیدی در مورد وضعیت جریان، پورت و صف ها بدست آوریم.

آمار این ماژول دارای دو تابع اصلی است: یکی به ایجاد ساختار داده مورد نیاز برای ایجاد یک پایگاه داده از آمار مربوط به گره های شبکه اختصاص داده شده است، یکی دیگر برای پیاده سازی مجموعه ای از داده های استخراج شده از پیام ها در مورد آمار اختصاص داده شده است. پیام های پاسخی که از سوئیچ های شبکه به دست می آیند،

شامل p_flow_stats، ofp_port_stats، ofp_queue_stats هستند. علاوه بر آمار اساسی که پروتکل OpenFlow 1.0 در دسترس قرار می دهد، ما ویژگی های خاصی را به کنترل کننده اضافه کردیم که اجازه می دهد BeaQoS از داده های جمع آوری شده بهره برداری کند تا محاسبات پارامترهای مفید برای اعمال راهبرد انتخاب شود. آمار اضافی محاسبه شده توسط BeaQoS در مقایسه با موارد موجود در OpenFlow 1.0 در جدول 4 نشان داده شده است.

ویژگی اصلی استخراج شده Estimate Rate (ER) برای پورت ها، صف ها و جریان ها است. ما ER^t برآورد شده در یک زمان معین به صورت زیر محاسبه کردیم:

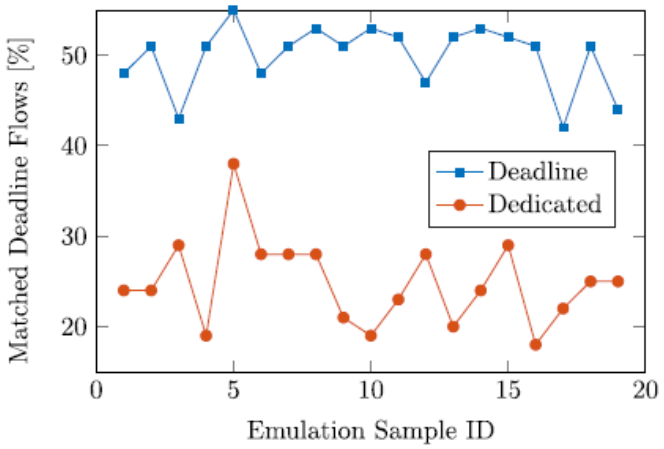
$$ER^t = \frac{TB^t - TB^{t-1}}{PI}$$

که در آن t نمونه گیری لحظه ای است، TB^t بایت های ارسال شده در لحظه فعلی است، TB^{t-1} بایت های منتقل شده در نمونه گیری لحظه قبلی است و PI نشان دهنده فاصله بازبینی در ثانیه است. بدیهی است که مقدار "بایت های منتقل شده" و در نتیجه بیان در (1) ممکن است به پورت ها، صف ها و جریان ها اعمال شود. پارامتر دیگری که ما استخراج کردیم تعداد جریانهایی که در حال حاضر متعلق به یک صف خاص هستند (جریان در هر صف) است.

مسیریابی این ماژول اصلاح شده است تا الگوریتم های پیشنهادی را اجرا کند. هنگامی که یک سوئیچ جریان جدیدی را دریافت می کند، آن را با کنترل کننده تماس می دهد تا بداند که در آن به جلو حرکت می کند. هنگامی که کنترل کننده باید هر جریان را به یک صف خاص اختصاص دهد، یک متغیر را شناسایی می کند که الگوریتم را اجرا می کند. BeaQoS یک روال را برای انتخاب صف درست بر اساس استراتژی انتخابی انجام می دهد و سپس از طریق نصب یک تغییر جریان، گره را اعلام می کند. رویکردهای پیشنهادی در بخش زیر شرح داده شده است.

5. تحلیل راهبردهای مسیریابی

در این بخش، ما دو سناریو اصلی ارائه می دهیم که در آن الگوریتم های پیشنهادی مجدد پیشنهادی مختلف را برای یافتن راه حل کارآمد مقایسه می کنیم. سناریوی اول، مسئله جریانهای اولویتی است که باید در یک زمان مشخص تحویل داده شود، همانطور که در بخش 3 آمده است. دومین مسئله مربوط به توازن بار در میان صف های مختلف در یک گره SDN واحد است.



شکل 3 درصد جریان هایی که زمان پایان را ارضا کرده و با H=100 محاسبه شده است.

جدول5 پیکر بندی صف ها

Queue ID	Service rate	Buffer size
q_0	0-3 Mbit/s	1000 packets
q_1	2 Mbit/s	1000 packets
q_2	1 Mbit/s	1000 packets

5.1. سناریوی مدیریت مأموریت

در این سناریو، ما هر دو "جریانهای پس زمینه" (BF) و "جریانهای زمان پایان" (DF) را در نظر می گیریم. همانطور که قبلا توضیح داده شده، DF جریانی است که برای آنها یک زمان مشخص مرتبط وجود دارد: جریان مفید است اگر و تنها اگر آن را در زمان مشخص شده [4] کامل کند. DF ها در برنامه های کاربردی مرکز داده (مثلا جستجو در وب، شبکه های اجتماعی) مورد توجه قرار می گیرند، در حالی که درخواست های کاربر باید در یک

هدف مشخص داده شود و وقتی که زمان به پایان می رسد، پاسخ ها، صرف نظر از تکمیل آنها، ارسال می شوند. علاوه بر این، سرویس های آنلاین دارای یک جریان عملیاتی مجموع بخش ها هستند که تقاضای کاربر در میان لایه های چندگانه سرورها (کارکننده ها) تقسیم می شوند و سپس نتایج آنها برای ایجاد پاسخ جمع می شود. ترکیبی از اهداف اهداف و جریان عملیاتی پارتیشن بندی، پیامدهای ترافیکی داخل مرکز داده را دارد. به طور خاص، برای هر جریان شبکه ای که توسط این کارکنان آغاز شده است، یک زمان مشخص مرتبط با آن وجود دارد.

ما دو طرح را برای ارائه پشتیبانی اولیه برای مدیریت زمان پایان در داخل شبکه SDN با مخلوط ترافیک BF، DF1 (هر جریان با مهلت 1) و DF2 (هر جریان با مهلت 2) پیشنهاد داده و پیاده سازی می کنیم. برای روشن شدن توصیف این رویکردها فرض می کنیم که تمام واسط های هر سوئیچ با سه صف پیکربندی شده اند: q0، q1، q2 به BF اختصاص داده شده است، در حالی که بقیه برای DF1 و DF2 استفاده می شود. طرح ها عبارتند از:

اختصاص داده شده این طرح هر کلاس ترافیک را به یک صف خاص از پورت سوئیچ در نظر گرفته شده اختصاص می دهد. پس از ورود جریان جدید داخل سوئیچ، موتور مسیریابی کنترل کننده Bea- Con تصمیم می گیرد کدام صف را براساس توصیفگر ترافیک جریان انتخاب کند. BF در q0، DF1 به q1 اختصاص داده می شود و DF2 به q2 اختصاص داده می شود.

زمان پایان این طرح زمانی شروع می شود که کنترل کننده یک درخواست از یک سوئیچ در مورد نحوه مدیریت جریان پیشنهادی دریافت می کند. ماژول مسیریابی نوع زمینه سرویس 5 را بررسی می کند: BF در q0 ثبت می شود، در حالی که برای DF1 یا DF2، کنترل کننده صف کمتر استفاده شده *qi را انتخاب می کند. استفاده از صف ها بر اساس تابع زیر U (qi) محاسبه می شود:

$$i^* = \arg\min_{i=1,2} U(q_i); \quad U(q_i) = s_{q_i} - \sum_{\nu} target_k \cdot n_{k,q_i} \quad i = 1, 2$$

که sqi نرخ سرویس qi است، یک اولویت شناخته شده قابل پیکربندی از یک فایل ویژگی های خارجی؛ k شاخصی که در میان کلاسهای خدمات قرار دارد (در اینجا DF1 و DF2)؛ هدف k نرخی است که ما نیاز داریم تا جریان کلاس k را تضمین کنیم. $N_{k,qi}$ تعداد جریانهایی که متعلق به کلاس k هستند و به qi اختصاص دارند.

هدف به حداکثر رساندن تعداد DF که دارای مهلت همسان است، حتی در صورت لزوم با هزینه جریان پس زمینه،است.

ما تجزیه و تحلیل عملکرد در PC درحال اجرای Mininet را انجام می دهیم(نسخه 2.1.0) [25]. سناریو از دو میزبان متصل به سوئیچ SDN تشکیل شده است. اجرای انتخابی سوئیچ Open vSwitch 2.0.2 [15] است که توسط یک نمونه از BeaQoS اجرا می شود که در همان دستگاه اجرا می شود. هر پورت سوئیچ با 3 صف، q0، q1، q2 پیکربندی شده است. نرخ تعیین شده برای هر بافر در جدول 5 نشان داده شده است. نرخ خدمات عمومی 3 مگابیت بر ثانیه است. صف به BF اختصاص داده شده است نرخ سرویس متغیر در محدوده 0 تا 3 مگابیت بر ثانیه: این بدان معنی است که q0 می تواند تنها در صورتی که صف های اولویت استفاده ای از پهنای باند کل را نداشته باشند ارائه شود. نرخ سرویس صف ها از طریق کنترل ترافیک (tc) در هسته لینوکس پیکربندی شده است.

ترافیک استفاده شده برای این شبیه سازی ها، از طریق ابزار iperf تولید شده است، همانطور که در بالا گفته شد، از 3 نوع جریان، BF، DF1 و DF2 از درصد و ویژگی های زیر تشکیل شده است: 30٪ از کل ترافیک BF، که مشخص است با نرخ تصادفی انتخاب شده در مجموعه {50، 60، 70، 80} kbit / s و مدت زمان 50 ثانیه؛ 55٪ DF1 است، داده ها را با سرعت 4.5 مگابیت بر ثانیه برای 1 ثانیه تولید می کند و با مهلت 9 ثانیه ای مواجه می شود؛ و 15٪ DF2 با نرخ داده 1.5 Mbit / s برای 1 ثانیه و با مهلت 5 ثانیه است. خلاصه ای از توصیفگرها و الزامات ترافیک در جدول 3 گزارش شده است، که قبلا برای نتایج در بخش 3 استفاده شده است.

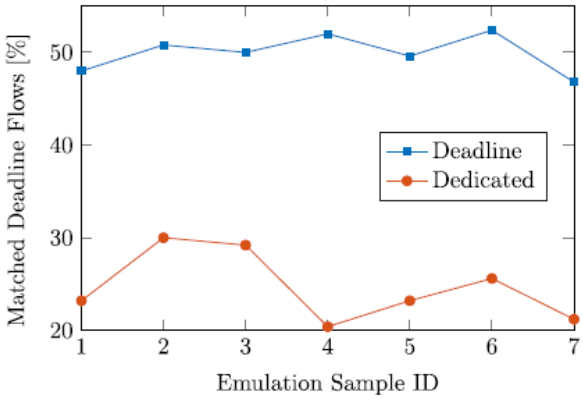
در این سناریو، عملکرد ارائه شده دو راه حل پیشنهاد شده را مقایسه می کنیم: اختصاص داده شده و مهلت مقدماتی.

ما یک شبیه سازی 3 ساعته از3000 جریانی که به شکل BF و DF جریان یافته است، همانطور که در بالا توضیح داده شد، اجرا شد. ما مقادیر به دست آمده را به طور متوسط برروی محور افقی (H) جریانهای تعادلی ارائه می کنیم. هر مقدار میانگین به نام ID شبیه سازی نمونه است. معیارهای استفاده شده برای مقایسه رویکردهای پیشنهادی، درصد جریانهای متقابل زمان مشخص شده (مثلا درصد جریانهای رضایت بخش زمان مشخص شده) و از

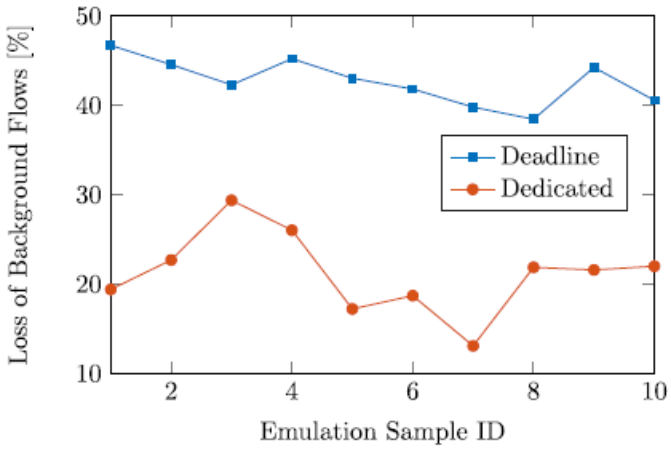
دست دادن جریان های پس زمینه (یعنی درصد بسته های از دست رفته جریانهای BF) است. برای نگرانی هایی شکل های 3 و 4، نشان می دهد که جریان های مطابق با زمان مشخص شده، به ترتیب روی H = 100 و H = 250، با توجه به جریان های DF تنها مورد استفاده قرار می گیرد. شکل های 5 و 6 نشان می دهد که افت جریان پس زمینه، دوباره به ترتیب روی H = 100 و H = 250، اما شامل تنها جریان BF است.

به طور مستقیم، مقادیر H بزرگ وضعیت حالت پایدار سیستم را ضبط می کنند و مقادیر H کوچکتر نویز اندازه گیری بیشتری را نشان می دهند. به جای انتخاب یک خاص H یا تلاش برای گرفتن یک رفتار مساوی حالت ماندگار، ما تصمیم گرفتیم اجراهای بیش از حد محور افقی زمان را به منظور دستیابی به رویکرد واقع گرایانه تر، همانطور که در [26] و [27] مورد بحث قرار گرفت، ردیابی کنیم. نتایج این آزمایش ها نشان می دهد که طرح زمان پایان اجازه می دهد تا محدودیت زمان تعداد بسیار بیشتری از DF را از طرح اختصاصی رعایت کند. در عمل، طرح زمان پایان می تواند به طور متوسط عملکرد این روش را دو برابر کند، با اشاره به جریان های مطابق زمان پایان (شکل 3 و 4). بهبود تعداد جریان DF که مطابق با آخرین زمان پایان است، با هزینه ترافیک BF بدست می آید که از تلفات بسته های بسیار بالاتر نسبت به طرح اختصاصی رنج می برد، همانطور که در شکل های 5 و 6 نشان داده شده است.

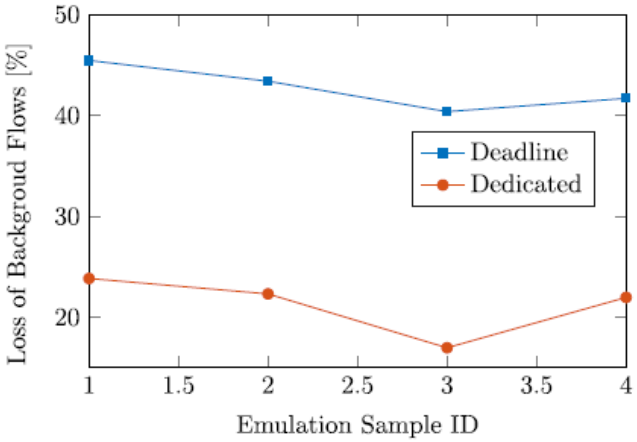
به طور خلاصه به طور مستقل از مقدار H، با توجه به درصد زمان پایان های ارضا شده برای جریان DF، با هزینه افزایش ضرر به دست آمده در جریان BF ، روش زمان پایان بهتر از یک اختصاص داده شده است.



شکل 4 درصد جریان هایی که شرط زمان پایان را برآورده کرده اند با h=250



شکل 5 درصد بسته های گم شده برای جریان های پس زمینه با H=100



شکل 6درصد بسته های گم شده برای جریان های پس زمینه با H=250

5.2. سناروی متعادل سازی صف

در مورد استراتژی های متعادل سازی بار، ما سه طرح برای مقابله بار ترافیکی در هر صف ارائه می دهیم. برای بهتر نشان دادن اصول عملی راه حل های ما، سناریو شبکه ای را در نظر می گیریم که در آن رابط هر سوئیچ دارای چهار صف موجود، q0، q1، q2 و q3 است. نرخ خدمات رابط خروجی به طور مساوی بین صف های مختلف تقسیم می شود. طرح های پیشنهادی عبارتند از:

بار حداقل این طرح شامل تخصیص جریان آینده به صفی است که کمترین بارگذاری را دارد. این وظیفه توسط ماژول مسیریابی کنترل BeaQoS انجام می شود. هنگامی که یک جریان جدید به سوئیچ SDN می رسد، کنترل

کننده میزان برآورد شده (محاسبه شده به عنوان (1)) صف ها متعلق به پورت خروجی مورد نظر را بررسی می کند و یکی از آن مقدار حداقل را انتخاب می کند.

اگر ما به میزان جریانها به عنوان اعداد فکر کنیم، ممکن است مشکل متعادل سازی بار را در میان صف های موجود به عنوان یک مشکل از پراکندگی مجموعه ای از اعداد به مجموعه ای از زیر مجموعه ها، مدل کنیم، به طوری که مقدار اعداد در هر زیرمجموعه (یعنی صف از سوئیچ ها) تا حد ممکن نزدیک است [28]. این مشکل در حال حاضر در پیشینه به عنوان Multi-Way Sharing شناخته شده است و NP-complete است. به خاطر ساده بودن ما می خواهیم یک الگوریتم را اجرا کنیم، که ما آن را Multiway می نامیم، و بر اساس ابتکاری حریصانه است که در زیر توضیح داده شده است.

Multiway در این طرح، تمام جریانها در ابتدا در صف q0 قرار می گیرند. سپس کنترل کننده به صورت دوره ای یک طرح را اجرا می کند که جریان ها را به ترتیب کاهشی براساس نرخ برآورد شده (ER) در (1) مرتب می کند و هر جریان را اختصاص می دهد. که با توجه به ترتیب کاهشی ER، اختصاص داده شده به صف با بهره وری کمتر، به منظور متعادل سازی بار در میان صف ها، تجزیه و تحلیل شده است.

N-migrations وقتی که تعداد جریانها عظیم است، رویکرد چند گانه به شدت محاسباتی به نظر می رسد، زیرا باید تجزیه و تحلیل و احتمالا تمام جریان های عبور از رابط را انجام دهد. به همین دلیل استراتژی N Migration را معرفی کردیم، جایی که تعداد مهاجرت جریان به N محدود می شود. الگوریتم در زمانهای برنامه ریزی شده اجرا می شود و N بار یک روتین اجرا می شود که جریان را از صف با بیشترین بار انتخاب می کند و دوباره آن را در صف با کمترین بار مسیریابی می کند. جریان انتخاب شده توسط استراتژی این است که اطمینان از بهترین متعادل سازی بار در میان صف ها حاصل شود. این انتخاب با ارزیابی تمام پیامدهای ممکن از طریق یک شبیه سازی ساده مسیر یابی مجدد انجام می شود.

اگر چه این استراتژی ها ممکن است مشابه باشند، نتایج عملکرد متفاوت هستند. تست های مربوط به تعادل صف استفاده از یک توپولوژی بسیار ساده مشابه Mininet همانطور که در سناریوی زمان پایان توضیح داده شده است،

می باشد. نرخ کلی در دسترس 4 مگابیت بر ثانیه است. تفاوت اصلی در پیکربندی صف ها درون سوئیچ OpenFlow است: هر درگاه سوئیچ دارای چهار صف، q0، q1، q2، q3 است و سرعت رابط خروجی به طور مساوی بین صف های مختلفی تقسیم می شود به طوری که هر یک دارای 1 مگابیت / ثانیه است.

ترافیک استفاده شده در این شبیه سازی ها با استفاده از ابزار iperf ساخته شده و شامل جریان هایی با نرخ تصادفی انتخاب شده در مجموعه {50، 60، 70، 80، 90، 100} kbit / s است. مدت زمان جریان 50 ثانیه است. شبکه با افزایش حجم کار آزمایش شده: 100، 125 و 150 جریان در حال اجرا با دانه های مختلف.

برای تجزیه و تحلیل بهتر نتایج، ما یک شاخص عملکرد ارائه می دهیم که اندازه گیری دقت الگوریتم ما را با توجه به راه حل بهینه ای ارائه می دهد که به طور ایده آل مقدار ترافیک دقیق موجود در هر صف را برای متعادل کردن بار مجاز کند. ما این پارامتر را index می نامیم و آن را در هر زمان لحظه ای t به صورت زیر محاسبه می کنیم:

$$index^t = \frac{\sum_i \left(r_{q_i}^t - \bar{r}^t\right)^2}{4}, \qquad t = 0, 1, \ldots$$

که r^t_{qi} نرخ خروجی اندازه گیری شده صف qi است و r^t برابر نرخ بهینه صف است که هر دو در لحظه t به دست می آیند. به عبارت دیگر، شاخص t یک اندازه از فاصله بین راه حل ما و یک ایده آل است.

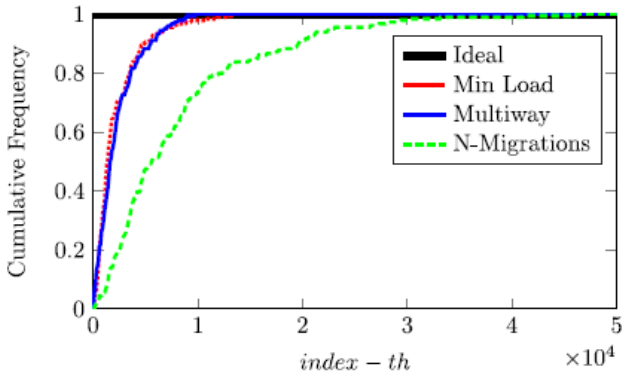
نمودارهای زیر نشان می دهد که توالی فرکانس (CF) $index^t$ است. CF به عنوان تعداد رخدادها در کل نمونه ها تعریف می شود که $index^t$ زیر آستانه معینی (index -th) است. شکل 7-9 نشان می دهد CF بر حسب index -th، برای Min Load، Multiway و N-Migrations در مورد 100، 125 و 150 جریان است. برای آنچه که در مورد رویکرد N-Migrations است، N پارامتر برای هر شبیه سازی روی 1 تنظیم شده است.

نتایج نشان می دهد که در تمام موارد مورد بررسی، طرح های Min Load و Multiway یک رفتار بسیار رضایت بخش را نشان می دهند و عملکرد مناسبتری را نسبت به رویکرد N-Migrations نشان می دهند. مثلا زمانی که 100 جریان درون شبکه را در نظر بگیریم، همانطور که در شکل 7 نشان داده شده است، می توانیم بگوییم که در 90٪ موارد فاصله بین راه حل ما و ایده آل برای استراتژی Min Load و Multiway بیشتر از 5000 نمی شود. برعکس، منحنی دقت N- مهاجرت روند کمتر تری نسبت به راه حل های جایگزین دارد: ارزش index برای این

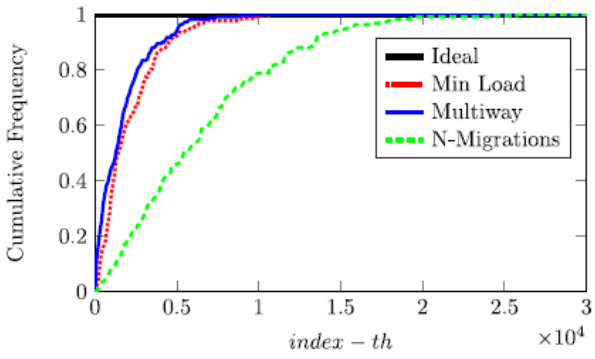
رویکرد در 50٪ موارد کمتر از 5000 است. به طور خاص، مهم است که توجه داشته باشیم که رفتارهای Min Load و Multiway خیلی نزدیک به ایده آل هستند (CF برای هر مقدار index -th، از جمله 0 است) و برای یک Index -th همپوشانی نسبتا کوچکی دارد.

همچنین شبیه سازی هایی که شامل 125 و 150 جریان هستند، همان رفتار را تأیید می کنند، همانطور که در شکل 8 و 9 نشان داده شده است.

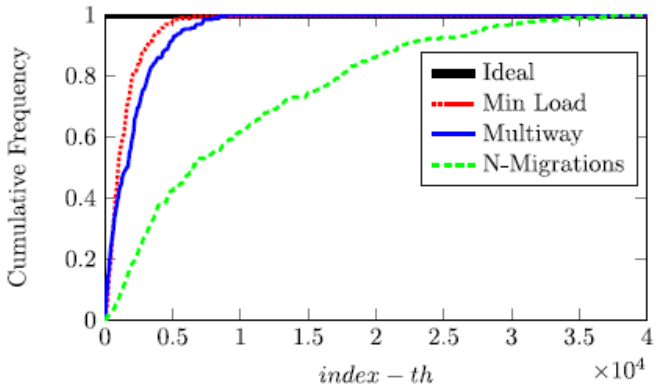
N-Migrations: نتایج نشان می دهد که رویکرد N-Migrations نمیتواند عملکرد مشابه Min Load و Multiway را به دست آورد. این به خاطر انتخاب پارامتر N است که کلید الگوریتم است. این پارامتر می تواند به منظور تنظیم عملکرد این روش تنظیم شود: همان طور که پارامتر N رشد می کند، رفتار الگوریتم به طرح Multi-way نزدیک می شود. انتخاب پارامتر N منجر به مبادله ای بین عملکرد و پیچیدگی محاسباتی می شود.



شکل 7عملکرد متعادل کردن صف با 100 جریان



شکل 8 عملکرد متعادل کردن صف با 250 جریان



شکل 9 عملکرد متعادل صف با 150 جریان

6. ملاحظات

6.1. افزایش کارایی

در مورد آمار (نگاه کنید به جدول 4) به دست آمده: انواع پیام های فرستاده شده توسط کنترل کننده، درخواست های جریان، صف و پورت است که برای جمع آوری اطلاعات در مورد نرخ های پورت، نرخ صف و آمار جریان هر کدام استفاده می شود. کنترل کننده سه پاسخ آماری را دریافت می کند، یکی برای پورت ها، یکی برای صف ها و یکی برای تمام جریان های عبور سوئیچ OpenFlow در یک لحظه داده شده است.

از آنجا که حداکثر اطلاعات ارسال شده از طریق قاب اترنت 1500 بایت است، هر آمار ارسال جریان آمار می تواند تنها اطلاعات مربوط به 10 جریان را گزارش کند. به همین دلیل تعداد بسته های آمار جریان در مورد جریان های F، f / 10 . با توجه به N تعداد سوئیچ های تشکیل دهنده شبکه و با توجه به دو بسته دیگر برای آمار پورت و صف، تعداد بسته های p که کنترل کننده باید در هر دوره پردازش کند:

$$p = \left(\left\lceil \frac{f}{10} \right\rceil + 2 \right) \cdot N$$

با توجه به تعداد قابل توجهی از جریانهای f و سوئیچهای N، تعداد بسته های دریافتی توسط کنترلر میتواند بزرگ باشد. این قیمت یک کنترل دقیق از یک شبکه SDN در سطح جریان (IntServ) می باشد. تعداد p می تواند با

استفاده از آمار جریان برای تعداد کمی از جریان های "جمع" کاهش می یابد. این می تواند کنترل ریز دانه را کاهش دهد اما کنترل کننده را از مدیریت تعداد زیادی از بسته ها رها می کند.

تجزیه و تحلیل عملکرد با یک سناریوی مقیاس بزرگ، گام بعدی کار ما در این موضوع خواهد بود.

6.2. هماهنگی سوئیچ

حتی اگر در این مقاله نتایج را با استفاده از یک سوئیچ OpenFlow در شبکه نشان می دهیم، ممکن است مفهوم را در دستگاه های مختلف SDN گسترش دهیم. ماژول مسیریابی که در کنترل BeaQoS اجرا می شود می تواند بیش از یک سوئیچ واحد را اداره کند. برای هر سوئیچ، کنترل کننده تمام پارامترهای مورد نیاز را محاسبه می کند تا بهترین الگوریتم انتخاب شده را ارائه دهد. به منظور گسترش این مفهوم به کل شبکه، با توجه به مسیر خاصی به مقصد، ممکن است بتوانید صف بهینه *qi را برای هر سوئیچ متعلق به مسیر خاص محاسبه کنید.

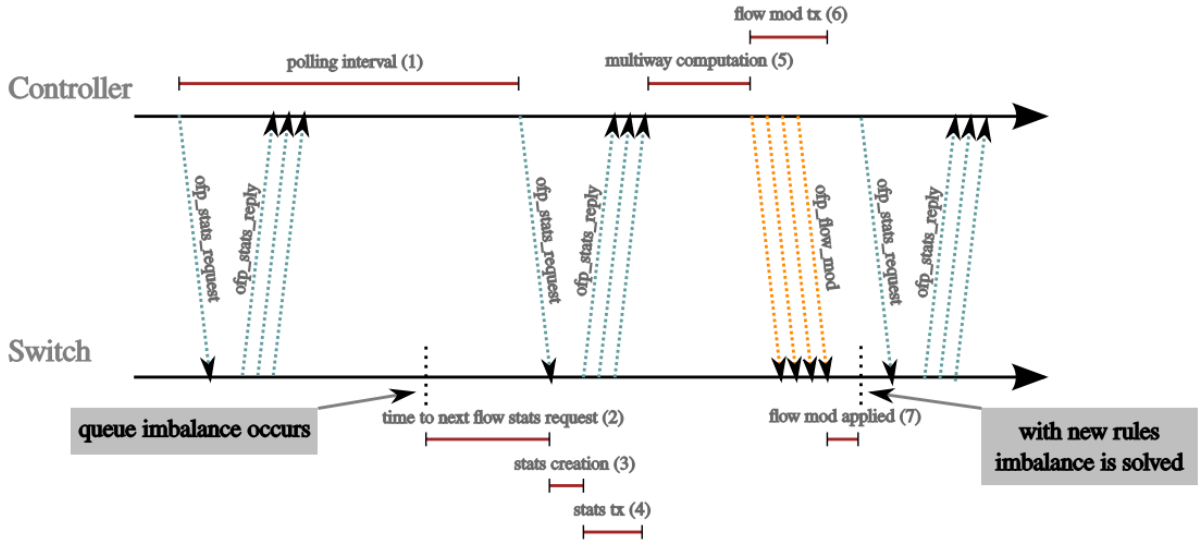
به همین ترتیب، از آنجا که BeaQoS کنترل کننده دارای کل شبکه است، راه حل دیگری نیز این است که تمام مسیرهای موجود بین منبع و مقصد را برای جریان در نظر گرفته شده مورد بررسی قرار دهیم. سپس کنترل کننده می تواند بهترین مسیر برای جریان خاص را محاسبه و سرانجام تصمیم صف بهینه *qi را برای تمام سوئیچ ها متعلق به مسیر انتخاب شده را انتخاب کند.

6.3. عملکرد زمان بندی و سربار در سناریوی متعادل کردن صف

در زمانبندی سناریوهای صف بندی، عملکرد زمانبندی ضروری است تا تضمین توازن بار لحظه ای تقریبا بین صفهای هر سوئیچ برقرار شود. مشکل اصلی این روش به علت طبیعت از راه دور اعمال کنترل کننده SDN است که مانند عملکردهای داخلی سوئیچ عمل می کند. زمان سپری شده از رویداد عدم تعادل بار در کلید و تعادل صف جدید (تاخیر تعادل صف) می تواند به عنوان مجموع اجزای مختلف بیان شود، همانطور که در شکل 10 نشان داده شده است.

تمام آزمون های ما با تعداد کمی از جریان ها انجام می شود. تا کنترل کننده برای مدیریت عملکرد هر جریان مجاز شود. با توجه به الگوریتم Multiway، کنترل کننده می تواند مقدار کل جریان عبوری یک سوئیچ SDN در یک زمان در محدوده میلی ثانیه مرتب دهد. علاوه بر این، با توجه به این که کنترل کننده با سوئیچ ها با استفاده از اتصال خارج از باند متصل است، زمان لازم برای ارائه تغییرات جریان ناچیز است.

در یک سناریوی بزرگ در مقیاس بزرگ با تعداد زیادی جریان، جمع کردن جریان، کاهش تعداد آمار جریان ارسال شده و زمان محاسبه الگوریتم Multiway امکان پذیر است.



شکل 10 عملکرد زمانی در سناریو متعادل کردن صف

7. نتیجه گیری

عدم امکان تنظیم نرخ سرویس صفها در یک سوئیچ OpenFlow از طریق یک دستورالعمل، محدودیتی است که میتواند قابلیتهای مدیریت کیفیت را در یک شبکه SDN کاهش دهد، اما این یک واقعیت برای حال حاضر است.

در این مقاله، با بهره گیری از مکانیزم مسیریابی مجدد، ما یک روش ارائه می دهیم که قادر به ارائه یک پشتیبانی مدیریت پایه زمان پایان و یک توازن خط صف کارآمد بدون هیچ گونه اصلاح مشخصات و سوئیچ های OpenFlow است. ما BeaQoS را ارائه می کنیم، نسخه به روز شده از کنترل کننده Beacon قادر به دریافت آمار از سوئیچ

OpenFlow، محاسبه آمار پیچیده تر و تصمیم گیری بهترین راه حل مسیریابی صف. ما نتایج حاصل از تست های عملکرد را نشان می دهیم که در آن ما رویکردهای مدیریت جایگزینی مؤثر و راه حل های متعادل سازی صف را مقایسه می کنیم. موارد مورد مطالعه ما نشان می دهد که راه حل های پیشنهادی می توانند نتایج رضایت بخش را در هنگام استفاده از محیط فعلی OpenFlow به دست آورند.

تحولات آینده به تست های مقیاس پذیر راه حل های ما اختصاص یافته و به مطالعه طرح های پیچیده تر مدیریت صف می تواند منجر به پیشرفت بیشتر در عملکرد شود. ما همچنین قصد توسعه رویکرد مسیریاب داخلی خود را برای محاسبه مسیرهای جایگزین بین منبع و مقصد به منظور کاهش تراکم شبکه داریم.

References

[1] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker, Ethane: Taking control of the enterprise, in: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, in: SIGCOMM '07, 2007, pp. 11–12.

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

[3] M. Marchese, QoS Over Heterogeneous Networks, Wiley Publishing, 2007.

[4] C. Wilson, H. Ballani, T. Karagiannis, A. Rowtron, Better never than late: meeting deadlines in datacenter networks, in: Proceedings of the ACM SIGCOMM 2011 Conference, in: SIGCOMM '11, 2011, pp. 50–61.

[5] D. Erickson, The Beacon Openflow Controller, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, in: HotSDN '13, 2013, pp. 13–18.

[6] L. Boero, M. Cello, C. Garibotto, M. Marchese, M. Mongelli, Management of non-conformant traffic in openflow environments, in: Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2015 International Symposium on, IEEE, 2015, pp. 1–6.

[7] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, S. G, Rsvp-te: extensions to rsvp for lsp tunnels, 2001, (RFC 3209).

[8] D. Applegate, M. Thorup, Load optimal mpls routing with n + m labels, in: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, 1, 2003, pp. 555–565.

[9] N. Handigol, S. Seetharaman, M. Flajslik, R. Johari, N. McKeown, Aster∗x: load- -balancing as a network primitive, in: Plenary Demo, 9th GENI Engineering Conference, in: 9th GENI, 2010.

[10] R. Wang, D. Butnariu, J. Rexford, Openflow-based server load balancing gone wild, in: Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, in: Hot-ICE'11, 2011. 12–12

[11] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, R. Wattenhofer, Achieving high utilization with software-driven wan, in: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, in: SIGCOMM '13, 2013, pp. 15–26.

[12] S. Agarwal, M. Kodialam, T. Lakshman, Traffic engineering in software defined networks, in: INFOCOM, 2013 Proceedings IEEE, 2013, pp. 2211–2219.

[13] S. Gutz, A. Story, C. Schlesinger, N. Foster, Splendid isolation: a slice abstraction for software-defined networks, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, in: HotSDN '12, 2012, pp. 79–84.

[14] OpenFlow Management and Configuration Protocol, Open Networking Foundation, 2014 https://www.opennetworking.org/images/stories/downloads/ sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf.
[15] Open vSwitch, 2014, (http://openvswitch.org/).
[16] OpenFlow Switch Specification - version 1.5.0, 2015, (https:// www.opennetworking.org/images/stories/downloads/sdn-resources/ onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf).
[17] Openflow switch specification - version 1.4.0, 2013, (https:// www.opennetworking.org/images/stories/downloads/sdn-resources/ onf-specifications/openflow/openflow-spec-v1.4.0.pdf).
[18] M. Dobrescu, N. Egi, K. Argyraki, B. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, S. Ratnasamy, Routebricks: exploiting parallelism to scale software routers, ACM Symposium on Operating Systems Principles (SOSP), 2009.
[19] T. Meyer, D. Raumer, F. Wohlfart, B. Wolfinger, G. Carle, Validated model-based performance prediction of multi-core software routers, Praxis der Informationsverarbeitung und Kommunikation (PIK), pp. 1–12.
[20] T. Meyer, D. Raumer, F. Wohlfart, B. Wolfinger, G. Carle, Low latency packet processing in software routers, in: Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2014), International Symposium on, 2014b, pp. 556–563.
[21] OpenFlow Switch Specification - version 1.0.0, 2009, (https:// www.opennetworking.org/images/stories/downloads/sdn-resources/ onf-specifications/openflow/openflow-spec-v1.0.0.pdf).
[22] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, Dynamo: Amazon's highly available key-value store, SIGOPS Oper. Syst. Rev. 41 (6) (2007).
[23] T. Hoff, 10 ebay secrets for planet wide scaling, 2009. (http://highscalability. com/blog/2009/11/17/10-ebay-secrets-for-planet-wide-scaling.html)
[24] W. Vogels, Performance and scalability, 2009 http://www.allthingsdistributed. com/2006/04/performance_and_scalability.html.
[25] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, N. McKeown, Reproducible network experiments using container-based emulation, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, in: CoNEXT '12, ACM, New York, NY, USA, 2012, pp. 253–264.
[26] C.G. Cassandras, Y. Wardi, B. Melamed, G. Sun, C.G. Panayiotou, Perturbation analysis for online control and optimization of stochastic fluid models, IEEE Trans. Autom. Control 47 (8) (2002) 1234–1248.
[27] M. Cello, M. Marchese, M. Mongelli, On the qos estimation in an openflow network: The packet loss case, IEEE Commun. Lett. 20 (3) (2016) 554–557.
[28] R.E. Korf, Multi-way number partitioning, in: International Joint Conferences on Artificial Intelligence, Citeseer, 2009, pp. 538–543.