روشی حافظه‌دار برای جایگزینی درایه‌های جدول جریان با هدف کاهش سربار کنترلر در شبکه‌های مبتنی بر نرم‌افزار

شیما ملکی[۱]،مهدی عباسی[۲]، محمد نصیری[۳]

[۱]دانشجوی کارشناسی ارشد مهندسی فناوری اطلاعات، دانشکده مهندسی، دانشگاه بوعلی سینا همدان،maleki.4591@yahoo.com

[۲]استادیار مهندسی کامپیوتر دانشکده مهندسی، دانشگاه بوعلی سینا همدان، abbasi@basu.ac.ir

[۳]استادیار مهندسی کامپیوتر دانشکده مهندسی، دانشگاه بوعلی سینا همدان،m.nassiri@basu.ac.ir

چکیده

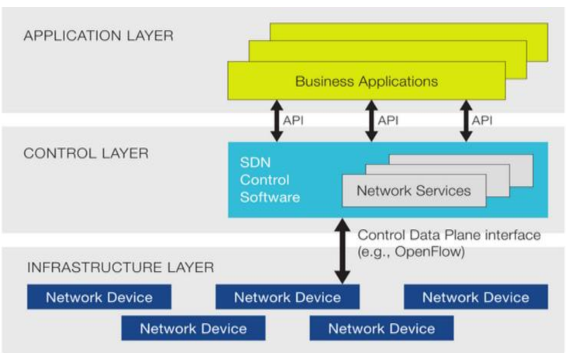
ایده شبکه‌های مبتنی بر نرم‌افزار[۱]با این هدف که کل شبکه به صورت یک موجودیت قابل برنامه‌ریزی مدیریت شود، ارائه شده و در حال توسعه است. پروتکل OpenFlow بعنوان پروتکلی مطرح در این زمینه، با بهره‌گیری از کنترلر متمرکز قابل برنامه‌ریزی، به منظور پیاده‌سازی سیاست‌های مدیریتی موردنظر، قوانین جدید هدایت بسته‌ها را در مورد جریان‌های متمایز بسته‌ها تحت عنوان درایه‌های جریان[۲]در جدول‌های جریان سوییچ‌های شبکه نصب می‌کند. جدول های جریان با وجود سرعت بالا ظرفیت محدودی دارند. بنابراین، مدت زمان نگهداری و نحوه جایگزینی درایه‌های مفیدتر، به چالشی مهم در این پروتکل تبدیل شده است. در نتیجه ناکارآمدی سیاست جایگزینی درایه‌های جدول جریان، به دلیل عدم حضور درایه‌های جریان متناظر با بسته‌های ورودی در جدول جریان سوییچ، میزان مراجعات به کنترلر جهت جایگزینی درایه‌های مذکور و در نتیجه تاخیر هدایت بسته‌ها افزایش می‌یابد. از همین‌رو، تمرکز اصلی این پژوهش، بر ارائه روشی آماری جهت جایگزینی درایه‌های جدول جریان است که بتواند سربار کنترلر را تا حد امکان کاهش دهد. ایده کلیدی در روش پیشنهادی آن است که از ویژگی‌های آماری جریان‌های ترافیکی موجود در جدول جهت انتخاب جریان انتخابی برای جایگزینی استفاده شود. پیاده‌سازی الگوریتم پیشنهادی به کمک ابزار MiniNet و مقایسه نتایج آن با الگوریتم‌های جایگزینی موجود، چون FIFO و Random نشان‌دهنده افزایش قابل توجه در نرخ برخورد در جدول جریان سوییچ Openflow بوده و برتری آن را در کاهش سربار کنترلر تایید می‌نماید.

واژه‌های کلیدی

شبکه مبتنی بر نرم‌افزار، جدول جریان، سوییچ OpenFlow، الگوریتم جایگزینی

1- مقدمه

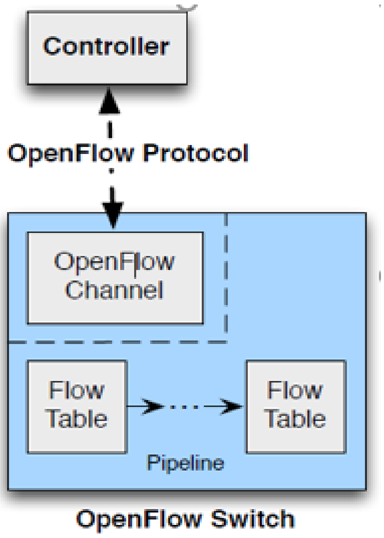
اخیراً شبکه‌های مبتنی بر نرم‌افزار در میان پژوهشگران و صنعت محبوبیت زیادی پیدا کرده است. شکل ۱، ساختار کلی از این شبکه‌ها را نشان می‌دهد. مزیت اصلی این شبکه‌ها انعطاف‌پذیری آن به دلیل جدا بودن سطح کنترل[۳] از سطح داده[۴]است. با استفاده از شبکه‌های مبتنی بر نرم‌افزار کل شبکه و عناصر آن بصورت یک شبکه مجازی دیده می‌شود. این شبکه‌ها با استفاده از نرم‌افزارها و رابط‌های برنامه‌نویسی کاربردی طراحی شده، کنترل می‌گردد. بنابراین، در شبکه‌های مبتنی بر نرم‌افزار نیازمند یک پروتکل ارتباطی بین نرم‌افزارها و سخت‌افزارها هستیم که بتواند کدهای نرم‌افزاری کنترلی را به تمامی ابزارهای گوناگون از تولیدکنندگان متفاوت اعمال کند[۱].



شکل ۱: معماری شبکه مبتنی بر نرم‌افزار[۲].

OpenFlow نام پروتکلی است که این کار را انجام می‌دهد. در معماری این پروتکل، که در شکل-۲، نمایش داده شده است، هر سوئیچ OpenFlow که عملیات هدایت بسته‌های شبکه را انجام می‌دهد، شامل یک یا چند جدول و

یک لایه انتزاعی می‌باشد. این لایه انتزاعی، برای ارتباط ایمن سوییچ با کنترلر از پروتکل OpenFlow استفاده می‌کند[۳].



شکل ۲: ساختار سوییچ OpenFlow [4]

مهمترین مولفه‌های هر سوییچ OpenFlow جدول‌های جریان آن است. جدول‌های جریان شامل درایه‌های جریان هستند، که هرکدام از این درایه‌ها تعیین می‌کند که چطور بسته‌های متعلق به یک جریان باید پردازش و ارسال شوند. درایه‌های جریان توسط کنترلر در جدول‌های جریان تعریف می‌شوند[۴]. این جدول‌ها ظرفیت محدودی دارند. بنابراین، تعداد درایه‌های جریانی که در آن‌ها می‌توان ذخیره نمود مشخص و محدود است. جدول‌های جریان از حافظه‌های تداعیگر سه وضعیتی[5] ساخته شده‌اند. حافظه‌های تداعیگر سه‌وضعیتی علاوه بر قیمت زیاد، مصرف انرژی بالایی دارد. بنابراین، افزایش اندازه جدول‌های جریان هزینه‌بر بوده و همچنین مصرف انرژی را افزایش می‌دهد[۵]. نحوه عملکرد جدول‌های جریان به این صورت است که اطلاعات هریک از بسته‌های جریان‌های ورودی با درایه‌های موجود در جدول‌های جریان مقایسه می‌شود. در صورت تطبیق بسته با یکی از درایه‌های جدول، سیاست مشخص‌شده در آن درایه جریان بر بسته مذکور اعمال می‌گردد. چنانچه بسته ورودی با هیچ‌یک از درایه‌های جریان مطابقت نداشته باشد؛ پیامی به نام Packet_in جهت پردازش آن به کنترلر ارسال می‌گردد. واضح است که در این حالت، زمان بیشتری برای پردازش بسته صرف می‌شود. در نتیجه، با افزایش تاخیر پردازش بسته‌ها، بسته‌های متعلق به جریان‌های ورودی در بافر سوییچ جمع شده و بافر سوییچ در نهایت پر می‌شود. این عملکرد منفی باعث می‌شود که بسته‌های جدید به هنگام ورود دور انداخته شوند. از آن‌جا که سوییچ‌های متعددی به کنترلر متصل است؛ بنابراین ارسال این پیام‌ها به کنترلر باعث افزایش سربار ارتباطی بین کنترلر و سوییچ می‌شود. بنابراین، یکی از معیارهای مهم برای جایگزینی درایه‌های جریان، کاهش سربار کنترلر است. این موضوع انگیزه اصلی تحقیق انجام شده در این مقاله است. با توجه به این‌که کنترلر مسئول به روزآوری و جایگزینی درایه‌های جدول جریان است، درصورت ناکارامد بودن سیاست کنترلر در جایگزینی درایه‌های جدول جریان ناپایدار شده و سربار محاسبات کنترلر افزایش می‌یابد. با استفاده از این نکته کلیدی، رویکرد پیشنهادی در این مقاله بر آن است که درایه‌های موجود در جدول جریان را بر اساس مشخصه‌های آماری جریان‌های متناظرشان به روز نماید.

در ادامه این مقاله در بخش دوم مروری بر کارهای گذشته ارائه شده و در آن مهمترین تحقیقاتی که در زمینه کاهش سربار کنترلر از طریق بهینه‌سازی الگوریتم جایگزینی درایه‌های جدول جریان انجام شده مرور شده است. در بخش بعدی پس از تعریف مشخصه‌های آماری جریان‌های شبکه، الگوریتم پیشنهادی برای کاهش سربار کنترلر بیان می‌شود. بعد محیط پیاده سازی روش پیشنهادی و روش‌های مورد مقایسه، معیارهای ارزیابی و مقایسه کارایی روش‌ها و نتایج حاصله از شبیه‌سازی در بخش چهارم مقاله به تفصیل تشریح می‌شود. بخش پنجم نیز به نتیجه‌گیری از تحقیق و ارائه پیشنهادات جهت توسعه آن اختصاص داده شده است.

2- مروری بر کارهای انجام شده

محققان راه‌حل‌هایی برای مقابله با این محدودیت جدول جریان اندیشیده‌اند که عبارتند از: تکنیک‌های خروج[6] برای حذف درایه‌ها از جدول جریان قبل از نصب درایه‌های جدید، تکنیک‌های مبتنی بر فشرده‌سازی[7] که این روش‌ها افزونگی اطلاعات بین درایه‌های جریان موجود در جدول‌های جریان را تا جای ممکن کاهش می‌دهند و تکنیک‌های شکستن و توزیع[8] است که در این روش‌ها سوییچ‌ها یک سیستم توزیع شده کلی می‌سازند که به هم وابسته هستند به جای اینکه مستقل از هم باشند. روش‌های پیشنهادی مبتنی بر خروج شامل الگوریتم‌های جایگزینی و مکانیزم‌های مبتنی بر زمان انقضا است[۶]. الگوریتم‌های جایگزینی که تاکنون در مقالات مختلف بکارگرفته شده‌اند و قابل پیاده‌سازی در شبکه مبتنی بر نرم‌افزار بوده‌اند FIFO و Random و LRU بوده‌اند.

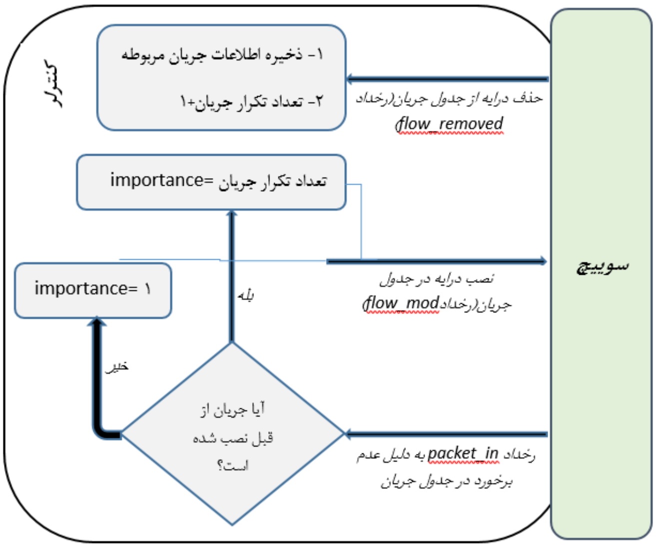
Adam zarek از دانشگاه تورنتو در سال ۲۰۱۲ در مقاله‌ای الگوریتم‌های جایگزینی (LRU,FIFO,Random) را مقایسه کرده‌اند. به این صورت که جدول جریان به صورت یک cache در نظر گرفته شده و درایه‌ها فقط زمانی حذف می‌شوند که جدول جریان پر باشد و بر اساس سیاست‌های جایگزینی نامبرده شده درایه مورد نظر انتخاب و حذف می‌شود. با بررسی این الگوریتم‌ها دریافته‌اند که FIFO و Random تفاوت نسبتا کمی با هم دارند ولی FIFO بهتر از Random است. با این وجود Random نیز نمی‌تواند خوب باشد چرا که ممکن است درایه‌هایی از جدول که ارجاع زیادی را دارند جهت جایگزینی انتخاب کند. اما LRU بهتر از دو الگوریتم دیگر است و اندازه نرخ برخورد برای آن بیشتر است. ولی LRU در آن زمان قابل پیاده سازی در شبکه‌های مبتنی بر نرم‌افزار نبوده است. در پایان عملکرد ترکیب timeoutهای مختلف با الگوریتم جایگزینیLRU نیز بررسی شده است. طبق نتایج، زمانی که طول جدول کوچکتر از تعداد درایه‌های فعال است اندازه timeout تاثیر چندانی ندارد و هر بهبود عملکردی وابسته به الگوریتم جایگزینی است[۷]. Eun-Do Kim و همکارانش نیز در سال ۲۰۱۴، در مقاله‌ای به منظور کاهش سربار کنترلر شبکه‌های مبتنی بر نرم‌افزار که بر اثر

عدم برخورد(table miss) در جدول جریان به هنگام ورود جریان‌ها ایجاد می‌شود، راهکاری را ارائه داده‌اند. در این کار، برای مدیریت جدول جریان از الگوریتمLRU در هنگام جایگزینی درایه‌های جریان، استفاده شده است و نتایج نشان داده که این الگوریتم بهتر از روش‌های FIFO و Random عمل می‌کند، از مفهوم Vacancy که در OpenFlow1.4 است استفاده کرده‌اند[۸، ۹]. اما این روش در سوییچ OpenFlow پیاده‌سازی شده است در صورتی که در شبکه مبتنی بر نرم‌افزار کارهای کنترلی باید در قسمت کنترلر پیاده شود. پیاده‌سازی الگوریتم LRU در کنترلر نیازمند این است که کنترلر از آخرین درایه‌ای که با جریان‌های ورودی به سوییچ منطبق شده است مطلع باشد و این اطلاع‌رسانی به کنترلر خود بار زیادی بر کنترلر وارد می‌کند. از آنجا که در این مقاله الگوریتم‌هایی که در کنترلر قابل پیاده‌سازی و مبتنی بر شبکه مبتنی بر نرم‌افزار هستند ارزیابی شده‌اند، لذا در این کار این الگوریتم مورد بررسی قرار نگرفته است. با توجه به مطالعات پیشین در این زمینه و همچنین نوظهور بودن شبکه‌های مبتنی بر نرم‌افزار می‌توان دید که تحقیقات در این زمینه کامل نمی‌باشد و از روش‌های آماری در مدیریت جدول جریان که می‌تواند تاثیر جدی در میزان بهینه‌سازی مکانیزم مدیریتی درایه‌های جدول جریان داشته باشد، استفاده نشده است. بنابراین، هدف ما در این پژوهش برآن بوده است که با استفاده از یک ویژگی آماری یعنی تعداد تکرار نصب جریان‌ها مکانیزم مدیریتی جدول های جریان را بهبود دهیم. در بخش بعدی راهکار پیشنهادی به طور کامل توضیح داده می‌شود. برای ارزیابی کارایی روش پیشنهادی در بخش پیاده‌سازی و ارزیابی، نتایج آن را با نتایج حاصل از الگوریتم Random و FIFO مقایسه می‌نماییم.

۳- روش پیشنهادی

در نسخه‌های قبل از OpenFlow۱,۴ هنگام نصب یک درایه جریان جدید از سوی کنترلر در جدول جریان، اگر جدول پر بود پیغامی از سوی سوییچ به کنترلر ارسال می‌شد و پر بودن جدول و عدم نصب درایه را به کنترلر جهت گرفتن تصمیم خبر می‌داد. در نسخه۱,۴ OpenFlow مفهوم جدیدی به نام Eviction اضافه شده است. اگر این ویژگی از سوی کنترلر فعال شود، هنگام نصب درایه جریان جدید در جدول جریان، اگر جدول پر باشد یکی از درایه‌ها را بر اساس ویژگی اهمیت (Importance) درایه‌ها جهت جایگزینی انتخاب می‌کند. درایه‌ای که درجه اهمیت آن از همه پایین‌تر است برای جایگزینی انتخاب می‌شود. در روش پیشنهادی نیز از همین ویژگی استفاده شده است. به این‌صورت که ویژگی Importance را بر اساس تعداد دفعاتی که جریان باید در کنترلر نصب شود اختصاص می‌دهیم. در نتیجه هنگام پر شدن جدول جریان درایه‌ای از جدول که کمتر مورد ارجاع و نصب قرار می‌گیرد از جدول حذف می‌شود. در شکل۳، روندنمای روش پیشنهادی نشان داده شده است.

در پروتکل OpenFlow رخدادی به نام flow_removed وجود دارد. زمانی که جریانی از جدول جریان حذف می‌شود، سوییچ با این رخداد، حذف شدنش را به کنترلر اطلاع می‌دهد. همان‌طور که در شکل۳ مشخص است زمانی که یک درایه جریان به هر دلیلی از جدول جریان حذف می‌شود و رخداد flow_removed ارسال می‌شود اطلاعات مربوط به این جریان در کنترلر ذخیره و به تعداد تکرار جریان مذکور یکی اضافه می‌شود. در میان جریان‌های ورودی به سوییچ ممکن است جریان‌هایی وجود داشته باشد که در فواصل زمانی زیاد تکرار می‌شوند. اهمیت درایه جریان متناظر با چنین جریان‌هایی با نرخ کمتری کاهش می‌یابد. هنگام نصب یک جریان نیز ابتدا بررسی می‌شود که آیا این درایه قبلا در جدول جریان نصب شده یا خیر. این جستجوی سریع با یک جدول درهم‌سازی پیاده‌سازی شده است. چنانچه مشخص شود که جریان قبلا در جدول نصب شده importance آن برابر تعداد تکرار جریان می‌شود و در غیر این‌صورت مقدار اولیه ۱ را می‌گیرد.



شکل ۳: روندنمای روش پیشنهادی

بنابراین، بر اساس الگوریتم شکل ۳، معمولا درایه‌هایی در جدول می‌مانند که بیشتر مورد ارجاع قرار می‌گیرند. در نتیجه انتظار می‌رود میزان عدم برخورد درجدول جریان و همچنین سربار کنترلر کاهش قابل توجهی داشته باشد. برای بررسی این موضوع به پیاده‌سازی روش پیشنهادی و مقایسه آن با روش‌های موجود می‌پردازیم.

۴- پیاده‌سازی و نتایج ارزیابی

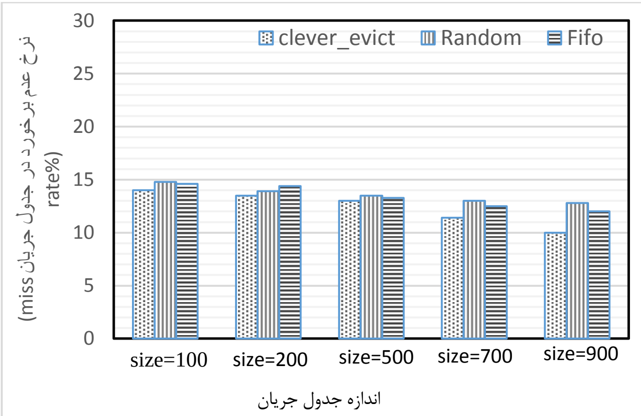
در پیاده‌سازی روش پیشنهادی از سیستم‌عامل ubuntu14.04 استفاده شده است. همچنین، از مقلد Mininet[10]، کنترلر RYU[11] و سوییچ نرم‌افزاری openvswitch[12] که پروتکل OpenFlow را دارد، استفاده شده است. برای ایجاد ترافیک از دو مجموعه داده استفاده شده است که مشخصات آن‌ها در جدول-۱ آمده است.

جدول۱: مشخصات مجموعه ترافیک های استفاده شده

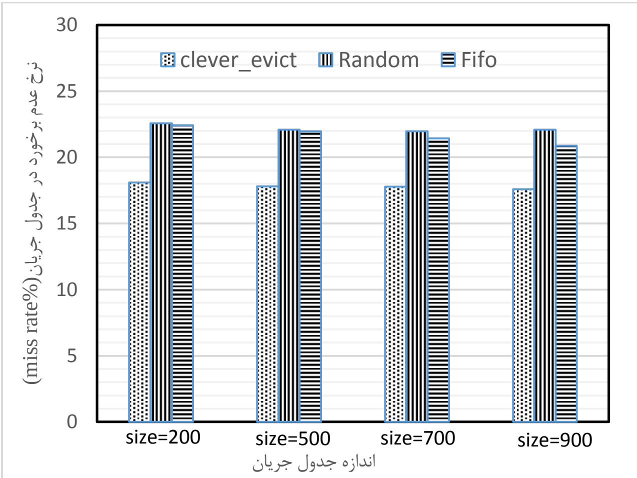
مجموعه داده	تعداد بسته ها	اندازه بسته‌ها	مدت زمان
Trace_file1	۷۹۱۶۱۵بسته	۴۴۹بایت	۳۰۰ثانیه
Trace_file2	۲۰۴۱کیلو	۶۰-۹۶بایت	۶۸۲ثانیه

Trace_file1 مجموعه ترافیکی از شبکه‌های شخصی و trace_file2 از مراکز داده دانشگاه‌ها است. اندازه hard_timeout نیز ۳۰۰ ثانیه بوده است. این پارامتر نشان دهنده حداکثر مدت زمانی است که درایه‌ها می‌توانند در جدول بمانند. اندازه idle_timeout نیز درآزمایش‌های اول تنظیم نشده و در آزمایش آخر برابر با ۱۰و ۳۰ بوده است. اندازه idletimeout حداکثر مدت زمانی است که یک درایه به صورت غیرفعال می‌تواند در جدول باشد.

در شکل ۴ و ۵ عملکرد سه روش FIFO، Random و روش پیشنهادی با عنوان clever_evict از نظر میزان نرخ عدم برخورد در جدول جریان(miss rate%) بر روی دو مجموعه ترافیک متفاوت با ویژگی‌های بالا نشان داده و مقایسه شده است. در تمام این حالات اندازه idle_timeout تنظیم نشده تا عملکرد روش‌های جایگزینی به خوبی و بدون تاثیر مقدار idle_timeout بررسی شود.

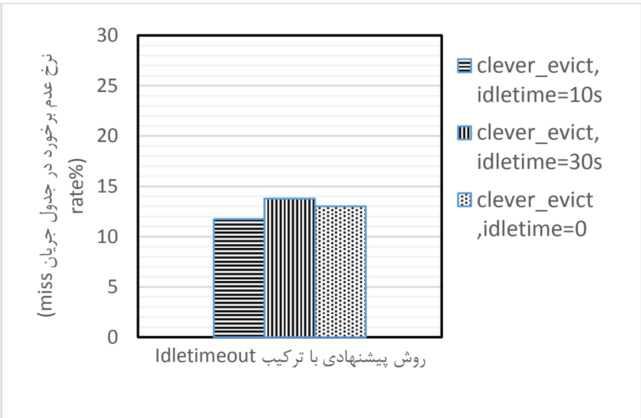


شکل ۴: تاثیر روش پیشنهادی بر عملکرد جدول جریان با trace_file1



شکل ۵: تاثیر روش پیشنهادی بر عملکرد جدول جریان با trace_file2

همان‌طور که در دو شکل ۴و۵ دیده می‌شود، دو روش FIFO و Random تقریبا نزدیک به هم ولی FIFO عملکرد بهتری داشته است، اما روش پیشنهادی در تمام آزمایشات و در حالت‌های متفاوت بهترین بوده است. میزان نرخ عدم برخورد در روش پیشنهادی به ازای هر دو مجموعه ترافیک و در سایزهای متفاوت از جدول جریان پایین‌تر از دو روش دیگر بوده است. دلیل آن این است که در رو روش FIFO و Random ممکن است درایه‌ای حذف شود که ارجاع زیادی دارند ولی در روش پیشنهادی چنین نیست و درایه‌هایی حذف می‌شوند که کمتر مورد ارجاع قرار می‌گیرند. هرچه اندازه جدول جریان بزرگتر می‌شود اندازه نرخ عدم برخورد در همه روش‌ها کاهش می‌یابد و دلیل آن بزرگ‌شدن اندازه جدول است. این کاهش نرخ عدم برخورد باعث می‌شود ارجاع به کنترلر کمتر و در نتیجه سربار آن کاهش یابد. در شکل ۶ نیز تاثیر تنظیم مقدار idle_timeout به همراه روش جایگزینی پیشنهادی نمایش داده شده است. در شکل ۶، مشاهده می‌شود که به ازای idle_timeout برابر با ۱۰ثانیه نرخ عدم برخورد پایین‌تر از حالتی است که idle_timeout برابر صفر است(وقتی میزان idle_timeout را برابر صفر قرار می‌دهیم در واقع به هنگام نصب درایه جریان مقداری برای idle_timeout تنظیم نمی‌شود). همچنین به ازای idle_timeout برابر ۳۰ثانیه این مقدار نسبت به حالت بدون idle_timeout افزایش یافته است. این مقایسه نشان می‌دهد که علاوه بر روش جایگزینی درایه‌ها در جدول جریان مقدار idle_timeout نیز در اندازه نرخ برخورد از اهمیت بالایی برخوردار است. از این رو می‌توان نتیجه گرفت که تنظیم صحیح مقدار idle_timeout خود تاثیر مهمی بر مدیریت جریان دارد که در کارهای آینده بررسی خواهد شد.



شکل ۶: روش پیشنهادی و تنظیم idle_timeout

نتیجه گیری

در شبکه‌های مبتنی بر نرم‌افزار سطح کنترلی از سطح داده جدا می‌باشد. پروتکل ارتباطی بین این دو سطح OpenFlow و سوییچی که مبتنی بر این پروتکل است سوییچ OpenFlow است. جدول‌های جریان موجود در این سوییچ‌ها با وجود سرعت بالا ظرفیت محدودی دارند. بنابراین، مدت زمان نگهداری و نحوه جایگزینی درایه‌های مفیدتر، به چالشی مهم در این پروتکل تبدیل شده است. با توجه به مطالعات پیشین در این زمینه و همچنین نوظهور بودن شبکه‌های مبتنی بر نرم‌افزار می‌توان دید که تحقیقات در این زمینه کامل نمی‌باشد و از روش‌های آماری که می‌تواند در بهینه‌سازی سیاست جایگزینی درایه‌های جدول جریان تاثیر جدی داشته باشد استفاده

نشده است. در این مقاله روشی نوین برای جایگزینی درایه‌های جدول جریان با هدف کاهش سربار کنترلر ارائه نمودیم. روش ارائه شده از ویژگی‌های آماری جریان‌ها در جهت انتخاب درایه مناسب برای جایگزینی و به روز سازی جدول جریان استفاده می‌کند. نتایج پیاده‌سازی روش پیشنهادی در کنار روش‌های FIFO وRandom نشان داد که روش پیشنهادی با افزایش نرخ برخورد، سربار کنترلر را به میزان قابل توجهی بیشتر از روش‌های موجود کاهش می‌دهد. برای کارهای آتی می‌توان از سایر ویژگی‌های آماری که از جریان‌ها بدست می‌آید به صورت جدا ونیز به صورت ترکیبی برای الگوریتم-های جایگزینی جدید و همچنین تنظیم مناسب idle_timeout و ترکیب این روش‌ها با هم استفاده کرد تا نتایج بهتری حاصل شود.

مراجع

[1] Kreutz, D., et al., Software-defined networking: A comprehensive survey. proceedings of the IEEE, 2015. 103(1): p. 14-76.
[2] Committee, O.M.E., Software-defined networking: The new norm for networks. ONF White Paper, 2012.
[3] McKeown, N., et al., OpenFlow: enabling innovation in campus networks (OpenFlow White Paper). Online: http://www. openflowswitch. org, 2008.
[4] <openflow-spec-v1.4.0.pdf>.
[5] Sezer, S., et al., Are we ready for SDN? Implementation challenges for software-defined networks. Communications Magazine, IEEE, 2013. 51(7): p. 36-43.
[6] Nguyen, X.-N., et al., Rules Placement Problem in OpenFlow Networks: a Survey. 2016.
[7] Zarek, A., Y. Ganjali, and D. Lie, Openflow timeouts demystified. Computer Engineering Research Group: University of Toronto, 2012.
[8] Kim, E.-D., et al. Flow table management scheme applying an LRU caching algorithm. in Information and Communication Technology Convergence (ICTC), 2014 International Conference on. 2014. IEEE.
[9] Kim, E.-D., et al. A flow entry management scheme for reducing controller overhead. in Advanced Communication Technology (ICACT), 2014 16th International Conference on. 2014. IEEE.
[10] Mininet: an instant virtual network on your laptop (or other PC). [Online]. Available: http://mininet.org/.
[11] https://osrg.github.io/ryu/-ryu controller.
[12] Open vSwitch: an open virtual switch. [Online]. Available: http://openvswitch.org/.

---

[1] Software defined network

[2] Flow Table

[3] Control Plane

[4] Data Plane

[5] TCAM: Ternary Content Addressable Memory

[6] Eviction

[7] Relying on Compression

[8] Split & distribution