مقاله پژوهشی

استفاده از معماری SAHAR جهت مقاوم‌کردن سطح کنترلی شبکه نرم‌افزار محور در برابر حملات منع سرویس

مهران شتابی و احمد اکبری

چکیده: شبکه نرم‌افزار محور (SDN) نسل بعدی معماری شبکه است که با جداکردن سطح داده و سطح کنترلی، کنترل متمرکزی را با هدف بهبود قابلیت مدیریت و سازگاری شبکه امکان‌پذیر می‌سازد. با این حال به دلیل سیاست کنترل متمرکز، این نوع شبکه مستعد از دسترس خارج شدن سطح کنترلی در مقابل حمله منع سرویس است. در حالت واکنشی، افزایش قابل توجه رخدادهای ناشی از ورود جریان‌های جدید به شبکه فشار زیادی به سطح کنترلی اعمال می‌کند. همچنین، وجود رخدادهای مکرر مانند جمع‌آوری اطلاعات آماری از سراسر شبکه که باعث تداخل شدید با عملکرد پایه سطح کنترلی می‌شود، می‌تواند به شدت بر کارایی سطح کنترلی اثر بگذارد. برای مقاومت در برابر حمله و جلوگیری از فلج‌شدن شبکه، در این مقاله معماری جدیدی به نام SAHAR معرفی شده که از یک جعبه کنترلی متشکل از یک کنترل‌کننده هماهنگ‌کننده، یک کنترل‌کننده اصلی نصاب قوانین جریان و یک یا چند کنترل‌کننده فرعی نصاب قوانین جریان (بر حسب نیاز) استفاده می‌کند. اختصاص وظایف نظارتی و مدیریتی به کنترل‌کننده هماهنگ‌کننده باعث کاهش بار کنترل‌کننده‌های نصاب قوانین جریان می‌شود. علاوه بر آن، تقسیم ترافیک ورودی بین کنترل‌کننده‌های نصاب قوانین جریان توسط کنترل‌کننده هماهنگ‌کننده بار را در سطح کنترلی توزیع می‌کند. بدین ترتیب، با تخصیص بار ترافیکی ناشی از حمله منع سرویس به یک یا چند کنترل‌کننده فرعی نصاب قوانین جریان، معماری SAHAR می‌تواند از مختل‌شدن عملکرد کنترل‌کننده اصلی نصاب قوانین جریان جلوگیری کرده و در برابر حملات منع سرویس مقاومت کند. آزمایش‌های انجام‌شده نشان می‌دهند که SAHAR در مقایسه با راهکارهای موجود، کارایی بهتری در مواجهه با حمله منع سرویس از خود نشان می‌دهد.

***کلیدواژه:* پروتکل OpenFlow، حمله منع سرویس، شبکه نرم‌افزار محور.**

۱- مقدمه

شبکه نرم‌افزار محور (SDN) نسل بعدی معماری شبکه است که در آن با مجزاشدن توابع کنترلی و ارسال شبکه از یکدیگر، سطح کنترلی برنامه‌پذیری را برای شبکه فراهم کرده و مدیریت شبکه را تسهیل می‌کند. از میان موارد موجود جهت محقق‌کردن شبکه نرم‌افزار محور، پروتکل OpenFlow [۱] بیشتر از همه مورد استفاده قرار گرفته است. در شبکه OpenFlow، ترافیک شبکه توسط قوانین جریانی که توسط کنترل‌کننده در جداول جریان سوئیچ‌ها نصب شده‌اند هدایت می‌شود. به طور کلی، دو روش برای تولید قوانین جریان توسط کنترل‌کننده وجود دارد: روش پیش‌فعال و روش واکنشی [۲]. در روش پیش‌فعال، کنترل‌کننده بر اساس استراتژی از پیش تعیین شده، قوانین جریان را قبل از شروع به کار شبکه نرم‌افزار محور در سوئیچ‌ها نصب می‌کند. در روش واکنشی، از قبل هیچ قانون جریان مشخصی در سوئیچ‌ها نصب نشده است. برای هر جریان جدیدی که با هیچ یک از قوانین جریان نصب‌شده مطابقت نداشته باشد یک رخداد فقدان جدول اتفاق می‌افتد که بدین ترتیب، یک درخواست OpenFlow به کنترل‌کننده ارسال شده و یک قانون جریان جدید بر اساس این درخواست تعیین می‌شود. در شبکه‌های نرم‌افزار محور، معمولاً از روش واکنشی که انعطاف‌پذیر است، استفاده می‌شود.

اگرچه OpenFlow برای تسهیل مدیریت شبکه و افزایش کاربرد آن از مزایای بسیار خوبی برخوردار است، اما رویکرد واکنشی آن در قبال نصب قوانین جریان به راحتی باعث می‌شود راهی برای حملات منع سرویس (DoS) به کنترل‌کننده ایجاد شود، چرا که کنترل‌کننده مجبور است به تمامی بسته‌های ناشی از فقدان جدول رسیدگی کند [۳] تا [۵]. میزبانی که سازش کرده است می‌تواند تعداد زیادی جریان جعلی کوتاه مدت را ایجاد کرده و به شبکه OpenFlow ارسال کند. هنگامی که سوئیچ OpenFlow این جریانات مخرب را دریافت کرد، رخدادهای مربوط به فقدان جدول حادث می‌شود و به این ترتیب تعداد زیادی درخواست جریان (بسته‌های packet_in) به کنترل‌کننده ارسال می‌شود. در نتیجه منابع کنترل‌کننده کاملاً توسط این درخواست‌های جریان که با فرکانس بالا ایجاد می‌شوند مصرف شده و عملکرد عادی کنترل‌کننده مختل می‌شود [۶]. پیامدهای کلی ناشی از حمله منع سرویس به شبکه OpenFlow شامل ۱) پربارشدن سوئیچ‌ها، ۲) بروز ازدحام در کانال ارتباطی سطح داده با سطح کنترلی، ۳) پربارشدن کنترل‌کننده و ۴) سرریزشدن جداول جریان سوئیچ است [۷] و [۸]. از بین این پیامدها، پربارشدن سوئیچ‌ها و سرریزشدن جداول جریان سوئیچ تنها سوئیچ قربانی را تهدید می‌کند، در حالی که ناهنجاری مربوط به سرویس‌دهی کنترل‌کننده می‌تواند باعث مختل‌شدن و از کار افتادن کل شبکه شود. بنابراین، بروز ازدحام در کانال ارتباطی سطح داده با سطح کنترلی و پربارشدن کنترل‌کننده بیشترین تهدیدات ناشی از حمله منع سرویس در شبکه OpenFlow را موجب می‌شود.

در این مقاله به منظور مقاوم‌کردن شبکه نرم‌افزار محور در برابر حملات منع سرویس و افزایش دسترس‌پذیری آن، از معماری SAHAR[1]

این مقاله در تاریخ ۲۰ آبان ماه ۱۳۹۸ دریافت و در تاریخ ۱۱ مرداد ماه ۱۳۹۹ بازنگری شد.

مهران شتابی، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران، (email: mshetabi@gmail.com).

احمد اکبری (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران، (email: akbari@iust.ac.ir).

1. Scalable Control Plane Architecture for Achieving High Availability through Task Division and Redundancy

برای سطح کنترلی شبکه نرم‌افزار محور استفاده شده است. معماری SAHAR که متشکل از یک جعبه کنترلی می‌باشد با هدف افزایش دسترس‌پذیری شبکه نرم‌افزار محور در برابر تغییرات چشم‌گیر در رخدادهای حاصل از ورود جریان‌های جدید به شبکه (ناشی از خواص زمانی و مکانی ترافیک شبکه) و وجود رخدادهای مکرر حاصل از جمع‌آوری اطلاعات آماری از کل شبکه که باعث پربارشدن کنترل‌کننده می‌شود، طراحی شده است. این جعبه کنترلی شامل یک کنترل‌کننده هماهنگ‌کننده، یک کنترل‌کننده اصلی نصاب جریان و یک یا چند (بر حسب نیاز) کنترل‌کننده فرعی نصاب جریان است. با واگذارکردن کارهای نظارتی و مدیریتی شبکه نرم‌افزار محور به کنترل‌کننده هماهنگ‌کننده، عملاً بار کاری کنترل‌کننده‌های نصاب جریان کاهش می‌یابد. وظیفه کنترل‌کننده‌های نصاب جریان پیکربندی سطح داده شبکه نرم‌افزار محور از طریق نصب قوانین جریان متناسب با برنامه‌های کاربردی شبکه‌ای در حال اجرا بر روی آنها است. علاوه بر این، با افزایش بار ترافیک ورودی به شبکه که می‌تواند ناشی از حمله منع سرویس باشد، کنترل‌کننده هماهنگ‌کننده ترافیک ورودی را به روش آماری دسته‌بندی کرده و پس از فعال‌کردن یک یا چند (بر حسب نیاز) کنترل‌کننده فرعی نصاب جریان، بخشی از ترافیک ورودی را در قالب دسته‌های تشخیص داده شده به هر یک از آنها تخصیص می‌دهد. این امر باعث توزیع بار ترافیک ورودی بین کنترل‌کننده‌های نصاب جریان شده که در نتیجه، بار کاری هر یک از کنترل‌کننده‌های نصاب جریان کاهش می‌یابد. بدین ترتیب، در معماری SAHAR با در نظر گرفتن قابلیت چند کنترل‌کننده‌ای ارائه‌شده در OpenFlow Specification ۱.۲ [۹]، از افزونگی کنترل‌کننده‌ها و تقسیم کار بین آنها استفاده می‌شود تا با تخصیص ترافیک ناشی از حمله منع سرویس به یک یا چند کنترل‌کننده فرعی نصاب جریان اثر حمله به کنترل‌کننده شبکه نرم‌افزار محور خنثی شود (یا حداقل کاهش یابد) و دسترس‌پذیری شبکه نرم‌افزار محور افزایش یافته و تغییرات شدید و ناگهانی ترافیک شبکه را تحمل کند.

در ادامه، ساختار مقاله بیان شده است. در بخش ۲ به کارهایی که به مقاوم‌سازی سطح کنترلی شبکه نرم‌افزار محور در برابر حملات منع سرویس پرداخته‌اند اشاره شده است. در بخش ۳، میزان بار ورودی به کنترل‌کننده شبکه نرم‌افزار محور مورد بررسی قرار گرفته است. معماری پیشنهادی- SAHAR- جهت مقاوم‌کردن سطح کنترلی شبکه نرم‌افزار محور در بخش ۴ بیان شده است. بخش‌های ۵ و ۶، به ترتیب به بررسی اثر حمله اشباع‌کردن سطح کنترلی توسط سطح داده در معماری SAHAR و بررسی تأثیر حملات منع سرویس بر روی معماری SAHAR پرداخته‌اند. در بخش ۷، کارایی معماری SAHAR در برابر حمله منع سرویس با چند راهکار موجود مورد مقایسه قرار گرفته و نهایتاً در بخش ۸، نتیجه‌گیری مقاله آورده شده است.

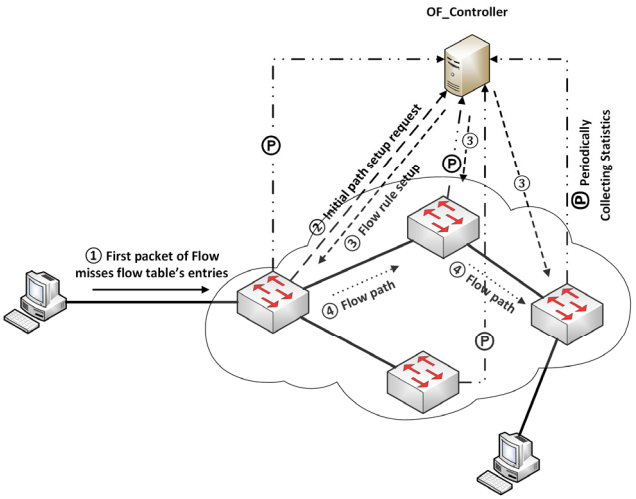
۲- کارهای مرتبط

مطالعات مربوط به کاهش تأثیر حملات منع سرویس به شبکه نرم‌افزار محور را می‌توان به دو دسته تقسیم کرد: محافظت از سطح داده و محافظت از سطح کنترلی.

محافظت از سطح داده بر گسترش عملکرد سطح داده یا اضافه‌کردن امکانات اضافی در سطح داده برای دفاع از حمله تمرکز دارد. OF-Guard [۱۰] از یک حافظه نهان در سطح داده استفاده می‌کند تا از غرق‌شدن کنترل‌کننده با درخواست‌ها جلوگیری کند. با این حال، استقرار چنین حافظه نهانی در سطح داده نامعین است که این باعث می‌شود تا معماری فاقد انعطاف‌پذیری شود. در AVANT-GUARD [۱۱] واحدی برای پیاده‌سازی SYN Proxy پیشنهاد شده است که قبل از انتقال جریان TCP ورودی به شبکه، عمل TCP Handshake را در سوئیچ‌ها انجام می‌دهد. با این حال، استفاده از این واحد باعث افزایش فشار بیشتر بر بافر سوئیچ می‌شود و علاوه بر این، در تعداد درگاه‌های پروکسی نیز محدودیت وجود دارد. LineSwitch [۱۲] برای بهبود AVANT-GUARD پیشنهاد شده است، چرا که می‌تواند به طور تصادفی جریان‌های آمده از منبع IP یکسان را که قبلاً یک بار TCP Handshake را در مورد آنها انجام داده است، پروکسی کند. SDN-Shield [۱۳] راهکار دیگری بر پایه AVANT-GUARD است که در آن از چندین واحد کاهش‌دهنده حمله مبتنی بر مجازی‌سازی تابع شبکه (NFV) جهت مقابله در برابر حملات توزیع‌شده منع سرویس در سطح داده شبکه نرم‌افزار محور استفاده شده است. با این حال، تمامی این روش‌ها که بر اساس SYN Proxy می‌باشند برای سایر پروتکل‌های شبکه نامعتبر هستند. Flood-Guard [۱۴] یک تحلیلگر قانون جریان پیش‌فعال را ارائه می‌دهد که می‌تواند کد منبع برنامه کاربردی مبتنی بر OpenFlow را تجزیه و تحلیل کرده و با نظارت بر وضعیت سراسری هر برنامه در زمان اجرا، چندین قانون پیش‌فعال را برای حفظ اجرای سیاست شبکه ایجاد کند. علاوه بر این، Flood-Guard از یک حافظه نهان در سطح داده استفاده می‌کند تا بسته‌های باقیمانده حاصل از فقدان جدول را بر اساس پروتکل نگهداری کند. با این حال، تحلیلگر کد منبع بسیار پیچیده است و فاقد قابلیت حمل و نقل در استقرار شبکه می‌باشد. همچنین، حافظه نهان سطح داده نمی‌تواند رفتاری عادلانه در برابر جریان‌های جدید خوش‌خیم را که از همان پروتکل جریان حمله استفاده کرده‌اند تضمین کند.

محافظت از سطح کنترلی بر بهبود امنیت سیاست کنترل‌کننده یا اجرای روش‌های تشخیص و فیلتر برای کاهش حمله منع سرویس تمرکز دارد. در FlowRanger [۱۵] بر روی کنترل‌کننده از الگوریتم رتبه‌بندی برای شناسایی کاربران عادی و پیش‌پردازش بسته‌های دارای اولویت بالاتر استفاده می‌شود. SGuard [۱۶] یک مکانیزم کنترل دسترسی برای شناسایی منشأ بسته‌ها معرفی می‌کند و از شش عنصر به عنوان بردار ویژگی در واحد دسته‌بندی برای شناسایی ترافیک مخرب استفاده می‌کند. Lim و همکارانش [۱۷] با تقسیم جریان‌ها به چندین صف، سیاست زمان‌بندی برای کنترل‌کننده پیشنهاد می‌کنند. هر صف با سوئیچی که در مسیر حمله منع سرویس واقع شده است مطابقت داشته و کنترل‌کننده بر اساس زمان‌بندی دوره‌ای (round-robin) به آن رسیدگی می‌کند. MLFQ[1] [۱۸] با استفاده از چندین صف درخواست قابل انبساط و جمع‌پذیر، اشتراک عادلانه‌ای از منابع کنترل‌کننده بین سوئیچ‌ها و میزبان‌های موجود در شبکه ایجاد می‌کند. در SDN-Shield [۱۳] ترافیک احتمالاً مخرب به سیستم تشخیص نفوذ هدایت شده و با تجمیع قوانین جریان به نحوی مناسب، تأثیر حمله منع سرویس کاهش می‌یابد. FloodDefender [۱۹] روشی برای توزیع بسته‌های مربوط به فقدان جدول از سوئیچ قربانی بر روی سوئیچ‌های همسایه آن پیشنهاد می‌کند تا ازدحام کانال بین سوئیچ قربانی و کنترل‌کننده کم شود. علاوه بر این، FloodDefender بسته‌های packet_in را به طور موقت نگهداری کرده و از فیلتر دومرحله‌ای برای شناسایی متناوب حمله استفاده می‌کند. با این حال، بافر بسته‌های packet_in باعث ایجاد تأخیر طولانی برای رسیدگی به جریان‌های جدید خوش‌خیم می‌شود. همچنین، توزیع بسته‌های مربوط

1. Multi-Layer Fair Queueing



شکل ۱: روند عملکرد کنترل‌کننده در شبکه نرم‌افزار محور جهت برقراری مسیر برای یک جریان جدید.

به فقدان جدول می‌تواند به سوئیچ‌های همسایه که تحت حمله سیلابی قرار دارند آسیب برساند. FMD[1] [۲۰] از روشی مبتنی بر مهاجرت جریان در سطح کنترلی شبکه نرم‌افزار محور جهت مقابله با حمله منع سرویس استفاده می‌کند. در این روش، از یک کنترل‌کننده برده به عنوان جانشینی برای کنترل‌کننده ارباب استفاده می‌شود که درخواست‌های مخربی که حجم زیادی نیز دارند به آن ارسال می‌شود. کنترل‌کننده ارباب، به تمامی درخواست‌های عادی OpenFlow رسیدگی می‌کند. پس از شناسایی حمله منع سرویس، FMD درخواست‌های تهدیدآمیز OpenFlow را به کنترل‌کننده برده منتقل می‌کند تا از کانال ارتباطی بین سوئیچ قربانی و کنترل‌کننده ارباب محافظت کند. برای محافظت از کنترل‌کننده ارباب در برابر پرباری، درخواست‌های مهاجرت‌شده به طور موقت در کنترل‌کننده برده ذخیره می‌شوند و برای پردازش بیشتر با سرعت محدود به کنترل‌کننده ارباب تحویل داده می‌شوند. SGS[2] [۲۱] برای محافظت از سطح کنترلی در برابر حملات منع سرویس ارائه شده و مشخصه اصلی آن استقرار چند کنترل‌کننده در سطح کنترلی به صورت خوشه‌بندی شده است. رویه‌های SGS در دو واحد سازماندهی می‌شود: تشخیص ترافیک ناهنجار و دفاع پویای کنترل‌کننده. تشخیص ترافیک ناهنجار بر روی سوئیچ‌های موجود در سطح داده متمرکز است تا جریانات جعلی را از موارد قانونی با استفاده از بردار ویژگی چهارتایی تشخیص دهد. دفاع پویای کنترل‌کننده با نگاشت مجدد کنترل‌کننده و ارسال پیام کنترل دسترسی به سوئیچ، اثرات ناشی از حمله منع سرویس را در سطح کنترلی کاهش می‌دهد.

همان طور که بیان شد، در راهکارهایی که در دسته محافظت از سطح داده قرار می‌گیرند، اکثر روش‌ها نیاز به تغییراتی در سوئیچ‌های OpenFlow یا اضافه‌کردن امکاناتی خاص به سطح داده دارند. از طرفی دیگر، تمام روش‌های دسته دوم (محافظت از سطح کنترلی) سعی دارند تا میزان مصرف منابع را که ناشی از حمله منع سرویس است به حداقل برسانند. در عمل، نرخ ورود بسته‌های OpenFlow نمی‌تواند توسط کنترل‌کننده محدود شود. علاوه بر این، ازدحام کانال ارتباطی سطح داده با سطح کنترلی که از نوع TCP یا TLS می‌باشد، دلیل اصلی بروز حمله منع سرویس در سطح کنترلی است. به منظور رفع محدودیت‌های اشاره‌شده در بالا، در این مقاله معماری SAHAR جهت مقابله با حملات منع سرویس معرفی می‌شود.

SAHAR با داشتن کنترل‌کننده‌های فرعی نصاب قوانین جریان که تکثیری از کنترل‌کننده اصلی نصاب قوانین جریان است و تخصیص ترافیک ورودی حاصل از حمله منع سرویس به آنها بدون در نظر گرفتن نوع پروتکل به کار رفته، نه تنها از کانال ارتباطی سطح داده با سطح کنترلی در برابر بروز ازدحام محافظت می‌کند بلکه از کنترل‌کننده اصلی نصاب قوانین جریان نیز محافظت می‌کند تا حملات منع سرویس کمترین تأثیر را بر روی جریان‌های جدید خوش‌خیم داشته باشد که این به معنی جامع‌بودن آن است. همچنین، جهت استفاده از SAHAR نیازی به هیچ گونه تغییری در برنامه‌های کاربردی شبکه و شبکه زیرساخت وجود ندارد که این به معنی شفافیت آن است.

۳- بررسی میزان بار ورودی به کنترل‌کننده شبکه نرم‌افزار محور

همان طور که در شکل ۱ آمده است، در یک شبکه نرم‌افزار محور که از سوئیچ‌های OpenFlow تشکیل شده است، کنترل‌کننده OpenFlow با ارسال متناوب پیام‌های چندبخشی [۲۲] به سوئیچ‌های تحت کنترل خود اطلاعات آماری را از سوئیچ‌ها جمع‌آوری کرده و با استفاده از این اطلاعات، پایگاه اطلاعات شبکه (NIB) خود را ایجاد می‌کند. بر اساس داده‌های همین پایگاه اطلاعات شبکه است که کنترل‌کننده می‌تواند دید سراسری از کل شبکه تحت کنترل خود داشته باشد.

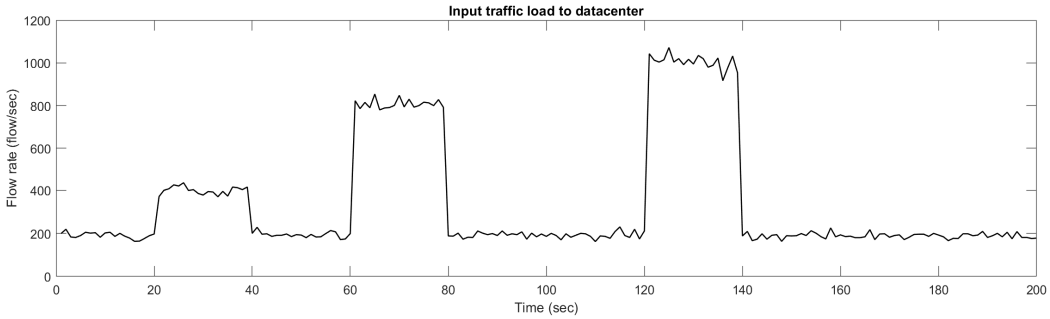
زمانی که جریان جدیدی به سوئیچ مدخل وارد می‌شود برای اولین بسته از جریان، بررسی جهت پیداکردن انطباقی با قوانین نصب‌شده در جدول جریان سوئیچ صورت می‌گیرد. اگر در جدول جریان سوئیچ انطباقی برای آن بسته پیدا نشد ورودی از جدول جریان که دارای پایین‌ترین اولویت بوده و با هر جریان ورودی انطباق دارد (ورودی جریان فقدان جدول) باعث می‌شود تا یک پیام Packet-in به کنترل‌کننده ارسال شود. این پیام، درخواستی جهت آغاز برقراری مسیر برای جریان واردشده است. کنترل‌کننده OpenFlow بر اساس داده‌های پایگاه اطلاعات شبکه، مسیر مناسبی را برای جریان واردشده محاسبه کرده و با نصب قوانین جریان مناسب در جداول جریان سوئیچ‌های واقع در مسیر سبب برقراری مسیر و در نتیجه هدایت تمامی بسته‌های جریان به مقصد مورد نظر می‌شود.

در یک مرکز داده با N سوئیچ، با فرض این که متوسط تعداد سوئیچ‌های واقع در مسیر $[N/۲]$ باشد و برای هر جریان در جدول جریان سوئیچ‌های واقع در مسیر دو قانون جریان (به منظور پیش‌راندن در دو جهت) نصب شود، بدین ترتیب بار ورودی به کنترل‌کننده حاصل از ورود جریان‌ها با سرعت ورودی λ جریان در ثانیه با رابطه زیر مشخص می‌شود

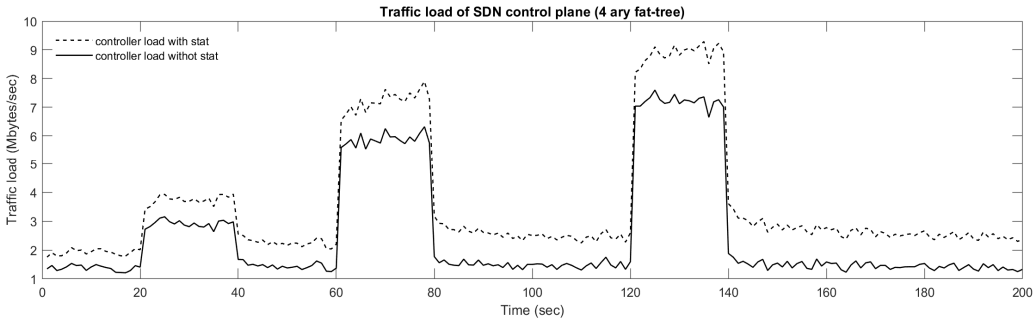
$$L_{arrival_flows} = \lambda \times ([m_{pi}] + [m_{po}] + [\frac{N}{۲}] \times ۲ \times [m_{fm}]) \qquad (۱)$$

که در آن m_{pi} معرف اندازه بسته packet_in، m_{po} معرف اندازه بسته packet_out و m_{fm} معرف اندازه بسته flow_mod (بسته‌ای که جهت نصب قانون جریان جدید از کنترل‌کننده به سوئیچ‌های OpenFlow ارسال می‌شود) است. بنابراین در روش واکنشی که برای هر جریان واردشده جهت پیش‌راندنش در کنترل‌کننده تصمیم‌گیری می‌شود، این امکان وجود دارد که با افزایش سرعت ورود جریان‌های جدید به سوئیچ مدخل، کنترل‌کننده دچار پرباری شود.

1. Flow Migration Defense
2. Safe-Guard Scheme



(الف)



(ب)

شکل ۲: (الف) ترافیک ورودی به مرکز داده و (ب) بار ورودی به کنترل‌کننده در یک مرکز داده با ساختار ۴_Ary_Fat_tree ((---) با بار حاصل از جمع‌آوری اطلاعات آماری و (-) بدون آن).

جدول ۱: پارامترهای استفاده‌شده در تولید ترافیک ورودی به مرکز داده.

Name	Model	Parameters
New flow arrivals	Poisson	$\lambda(t)=۱۹۲$
Internal rack flow ratio	Bernoulli	$R_{\text{int}}=۰٫۸$
TCP flow ratio	Bernoulli	$R_{tcp}=۰٫۸۵$
Flow duration, D_n	Pareto	$a_p=۱٫۵۰۴$ $M_p=۱٫۰۰۰۱$
Flow transmission rate, Y_n	Gaussian	$E[Y_n]=۴۳۰۳٫۶۹$ $Var[Y_n]=۶۹۹۳۶٫۳۷$
Flow size, S_n	$Y_n \times D_n$	$E[S_n]=۸۵۱۷٫۹۷$

عامل دیگر پربارشدن کنترل‌کننده شبکه نرم‌افزار محور، بار ورودی ناشی از جمع‌آوری اطلاعات آماری از سطح شبکه است. در یک مرکز داده، اطلاعات آماری حایز اهمیتی که از سوئیچ‌های OpenFlow موجود در آن می‌توان جمع‌آوری کرد شامل اطلاعات آماری مربوط به درگاه‌های متصل به سرویس‌دهنده‌ها و اطلاعات آماری مربوط به جریان‌های نصب‌شده در جداول جریان سوئیچ‌ها است. بنابراین در یک مرکز داده، بار ورودی به سطح کنترلی ناشی از جمع‌آوری اطلاعات آماری از سوئیچ‌های OpenFlow از رابطه زیر به دست می‌آید

$$L_{statistics}=N\times(([m_{portreq}]+H\times[m_{portrep}])+ ([m_{flowreq}]+F\times[m_{flowrep}]))\times\frac{۱}{T} \quad (۲)$$

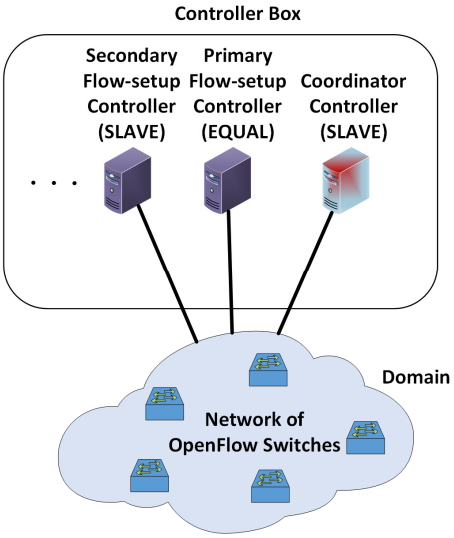
که در آن N معرف تعداد کل سوئیچ‌های OpenFlow در مرکز داده و H معرف مجموع کل درگاه‌های سوئیچ‌ها است که به سرویس‌دهنده‌ها متصل شده است. F تعداد کل جریان‌های نصب‌شده در سوئیچ‌ها و T دوره تناوبی درخواست اطلاعات آماری توسط کنترل‌کننده است.

جهت نمایان‌کردن تأثیر بار ناشی از جمع‌آوری اطلاعات آماری (رابطه (۲)) و ورود جریان‌های جدید به مرکز داده (رابطه (۱)) بر روی کنترل‌کننده شبکه نرم‌افزار محور، شبیه‌سازی در Matlab ver. R۲۰۱۸b [۲۳] انجام شده است. در این شبیه‌سازی، مرکز داده دارای ساختار ۴ Ary Fat_tree است. ترافیک ورودی به مرکز داده با توجه به نتایج به دست آمده توسط Yoonseon Han و همکارانش [۲۴] بر اساس پارامترهای جدول ۱ ایجاد شده است. نتیجه شبیه‌سازی که بر اساس (۱) و (۲) و همچنین روابط ارائه‌شده توسط Bong-yeol Yu و همکارانش [۲۵] به دست آمده است، در شکل ۲ آورده شده است.

همان طور که در شکل ۲- الف دیده می‌شود، در حالت عادی میزان ترافیک ورودی به مرکز داده ۱۹۲ جریان بر ثانیه است که برای مشاهده تأثیر افزایش سرعت آن بر میزان بار ورودی به کنترل‌کننده شبکه نرم‌افزار محور، این مقدار در بازه‌های زمانی ۲۰ تا ۴۰ ثانیه، ۶۰ تا ۸۰ ثانیه و ۱۲۰ تا ۱۴۰ ثانیه به ترتیب به ۴۰۰، ۸۰۰ و ۱۰۰۰ جریان بر ثانیه افزایش یافته است. در شکل ۲- ب میزان بار ورودی به کنترل‌کننده شبکه نرم‌افزار محور یک مرکز داده با ساختار ۴_Ary_Fat_tree در دو حالت (---) با بار حاصل از جمع‌آوری اطلاعات آماری و (-) بدون آن نشان داده شده است. همان طور که انتظار می‌رود، با افزایش سرعت ترافیک ورودی به مرکز داده، بار ورودی به کنترل‌کننده نیز به همان نسبت افزایش می‌یابد. نکته قابل توجه در قسمت پایین شکل ۲، میزان قابل ملاحظه بار ورودی حاصل از جمع‌آوری اطلاعات آماری به کنترل‌کننده است که تأثیر زیادی در بروز تداخل با ارسال پیام‌های پایه کنترل‌کننده (مانند نصب جریان) داشته و باعث ایجاد تأخیر در رسیدن اطلاعات به کنترل‌کننده و در نتیجه کاهش کارایی کنترل‌کننده می‌شود. بنابراین با جداکردن بار ناشی از جمع‌آوری اطلاعات آماری از بار حاصل از ورود بسته‌های packet_in می‌توان کارایی کنترل‌کننده و همچنین دسترس‌پذیری آن را افزایش داد.

٤- SAHAR، معماری پیشنهادی جهت مقاوم‌کردن سطح کنترلی SDN

در شبکه‌های نرم‌افزار محور، جهت استفاده از مزیت مدیریت مرکزی



شکل ۳: SAHAR، ساختار معماری پیشنهادی جهت کنترل‌کننده‌های شبکه نرم‌افزار محور.

آن، کنترل‌کننده تنها وظایف یک سوئیچ ساده جهت ارسال جریان‌ها در شبکه را بر عهده ندارد، بلکه با اجرای برنامه‌های مؤثری نظیر توزیع متوازن بار بر روی خطوط ارتباطی و سرویس‌دهنده‌های موجود در شبکه، یا کنترل کیفیت ارائه سرویس به جریان‌ها به صورت سیستم به سیستم نیاز به جمع‌آوری اطلاعات آماری از سطح شبکه دارد که همین امر نیاز به دسترس‌پذیری بالای کنترل‌کننده در شبکه‌های نرم‌افزار محور را جهت مدیریت مؤثر و یکپارچه آن الزامی می‌کند.

برای این که در شبکه‌های نرم‌افزار محور، کنترل‌کننده‌های OpenFlow از دسترس‌پذیری بالایی برخوردار باشند، دو راهکار وجود دارد [۲۶]. راهکار اول کاهش بار کنترل‌کننده‌ها است. همان طور که در بخش قبل مطرح شد، کنترل‌کننده OpenFlow پیام‌های زیادی را با سوئیچ‌های OpenFlow تبادل می‌کند، مخصوصاً زمانی که در حالت واکنشی عمل می‌کند. همین امر باعث پرباری کنترل‌کننده شده که نتیجه آن عدم توانایی در پردازش پیام‌های واردشده به آن است. راهکار دوم ایجاد افزونگی با جایگزین‌کردن یک کنترل‌کننده با چندین کنترل‌کننده است.

در جعبه کنترلی پیشنهادی که در معماری SAHAR به کار رفته است هر دو راهکار فوق جهت بالابردن دسترس‌پذیری کنترل‌کننده‌های OpenFlow در شبکه‌های نرم‌افزار محور مد نظر قرار گرفته است (شکل ۳). در این جعبه کنترلی، یکی از کنترل‌کننده‌ها نقش هماهنگ‌کننده را ایفا می‌کند که ارتباط آن با سوئیچ‌های OpenFlow تنها به منظور جمع‌آوری اطلاعات آماری آنها است. همچنین در این جعبه کنترلی، یک کنترل‌کننده اصلی و یک یا چند (بر حسب نیاز) کنترل‌کننده فرعی دیگر وجود دارد که وظیفه نصب قوانین جریان در جداول جریان سوئیچ‌های OpenFlow را بر عهده دارند. بدین ترتیب با تقسیم وظایف بین کنترل‌کننده هماهنگ‌کننده و کنترل‌کننده‌های نصاب قوانین جریان، علاوه بر کاهش بار کنترل‌کننده‌ها، از کنترل‌کننده‌های اضافی نیز استفاده شده است. برای اجرای مفهوم جعبه کنترلی در شبکه‌های نرم‌افزار محور نیاز به استفاده از چندین کنترل‌کننده جهت کنترل‌کردن سوئیچ‌های OpenFlow است که این امکان در نسخه ۱.۲ به بعد OpenFlow آورده شده است.

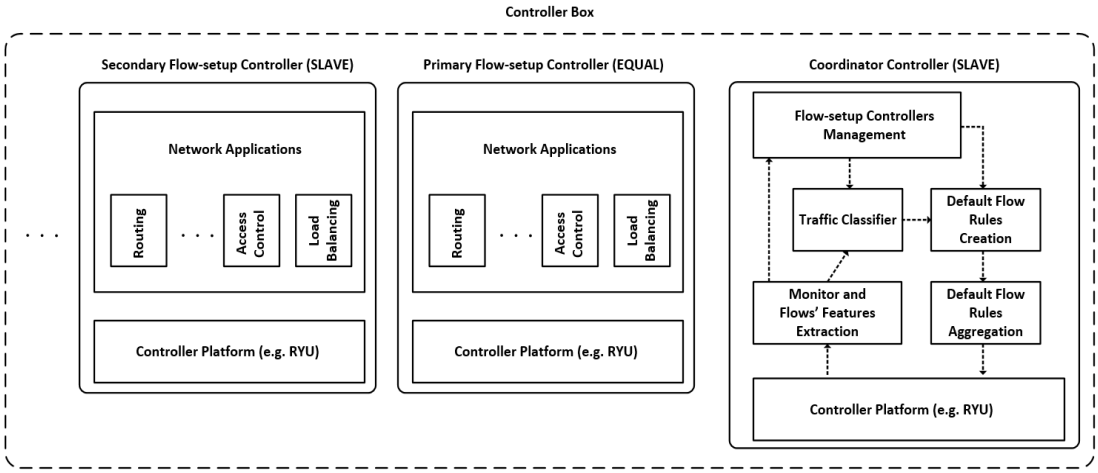
از نسخه ۱.۲ به بعد OpenFlow، قابلیت چندین کنترل‌کننده که در آن هر کنترل‌کننده می‌تواند یکی از سه حالت MASTER، SLAVE یا EQUAL را جهت کنترل‌کردن سوئیچ‌های OpenFlow داشته باشد در نظر گرفته شده است. از بین این سه حالت، حالت‌های MASTER و EQUAL به طور کامل به سوئیچ دسترسی داشته و توانایی دریافت تمامی پیام‌های غیر هم‌زمان (مثل packet-in) را از سوئیچ دارا می‌باشند. هر سوئیچ OpenFlow می‌تواند با چندین کنترل‌کننده EQUAL ارتباط برقرار کند، ولی فقط اجازه دارد تا با یک کنترل‌کننده MASTER ارتباط داشته باشد. در حالت SLAVE، دسترسی کنترل‌کننده به سوئیچ‌ها تنها دسترسی جهت خواندن از سوئیچ است و به غیر از پاسخ پیام چندبخشی که حاوی وضعیت سوئیچ است پیام غیر هم‌زمان دیگری را نمی‌تواند دریافت کند. با ارسال پیام OFPT_ROLE_REQUEST به سوئیچ‌ها، هر کنترل‌کننده می‌تواند حالت خود را تغییر دهد. با دریافت این پیام، سوئیچ پیام OFPT_ROLE_REPLY را به کنترل‌کننده ارسال می‌کند. اگر سوئیچ پیامی مبنی بر تغییر حالت کنترل‌کننده به MASTER را از آن دریافت کرد، حالت بقیه کنترل‌کننده‌های آن سوئیچ به SLAVE تغییر خواهد کرد. این عمل این امکان را به سوئیچ می‌دهد تا با داشتن چندین مجرای OpenFlow، در صورت از کار افتادن هر یک از کنترل‌کننده‌های مرتبط با آن دیگر نیازی به برقراری مجدد مجرای OpenFlow نباشد.

با دادن شناسه یکتا به هر یک از کنترل‌کننده‌های نصاب قوانین جریان موجود در جعبه کنترلی، سوئیچ‌های OpenFlow توانایی آن را پیدا می‌کنند که پیام‌های Packet-in را تنها به کنترل‌کننده‌ای ارسال کنند که شناسه آن در ورودی تطبیق‌یافته جدول جریانشان مشخص شده است. این قابلیتی است که می‌توان با استفاده از عملکردهای Nicira ضمیمه‌شده به سیستم عامل شبکه RYU انجام داد [۲۷]. برای این منظور، جهت دادن شناسه یکتا به هر یک از کنترل‌کننده‌های نصاب قوانین جریان از روال NXTSetControllerId استفاده می‌شود. بدین ترتیب، دیگر محدودیتی برای تعداد کنترل‌کننده‌های نصاب قوانین جریان موجود در جعبه کنترلی وجود نخواهد داشت. این امر، باعث مقیاس‌پذیر شدن جعبه کنترلی می‌شود تا با توجه به افزایش بار ترافیک ورودی به شبکه، تعداد کنترل‌کننده‌های فعال نصاب قوانین جریان نیز بر حسب نیاز افزایش یابد. بدین ترتیب، با توزیع بار مربوط به پیام‌های Packet-in بین کنترل‌کننده‌های نصاب قوانین جریان، بار تک‌تک آنها کاهش یافته و باعث افزایش دسترس‌پذیری سطح کنترلی شبکه نرم‌افزار محور می‌شود.

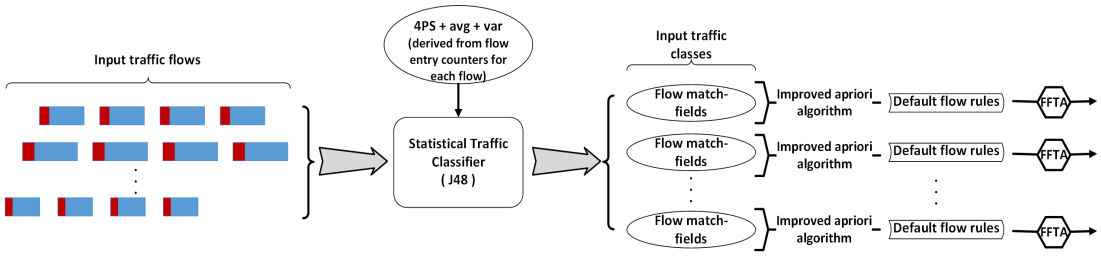
اشاره شد که در معماری SAHAR یکی از کنترل‌کننده‌ها نقش هماهنگ‌کننده را ایفا می‌کند که ارتباط آن با سوئیچ‌های OpenFlow تنها به منظور جمع‌آوری اطلاعات آماری آنها است و بنابراین حالت این کنترل‌کننده SLAVE است. کنترل‌کننده هماهنگ‌کننده بر اساس اطلاعات آماری جمع‌آوری شده، احتمال پربارشدن کنترل‌کننده اصلی نصاب جریان را تشخیص داده و با فعال‌کردن یک یا چند (بر حسب نیاز) کنترل‌کننده فرعی نصاب جریان، بار اضافی حاصل از افزایش ترافیک ورودی به شبکه را به کنترل‌کننده(های) فرعی نصاب جریان هدایت می‌کند. در ادامه، تشریح عملکرد کنترل‌کننده هماهنگ‌کننده و واحدهای تشکیل‌دهنده آن آورده شده است.

۴-۱ کنترل‌کننده هماهنگ‌کننده

کنترل‌کننده هماهنگ‌کننده که در حالت SLAVE قرار دارد به تمامی سوئیچ‌های OpenFlow واقع در دامنه‌اش متصل است. همان طور که در شکل ۴ نشان داده شده است، کنترل‌کننده هماهنگ‌کننده شامل تعدادی واحد برای نظارت بر شبکه و مدیریت عملکرد کنترل‌کننده‌های نصاب قوانین جریان است. وظیفه اصلی کنترل‌کننده هماهنگ‌کننده، تخصیص



شکل ۴: اجزای تشکیل‌دهنده معماری SAHAR.



شکل ۵: روند دسته‌بندی آماری ترافیک ورودی به شبکه و تولید قوانین جریان پیش‌فرض.

بخشی از ترافیک ورودی به کنترل‌کننده فرعی نصاب جریان است. این کار زمانی انجام می‌شود که کنترل‌کننده هماهنگ‌کننده تشخیص دهد که کنترل‌کننده اصلی نصاب جریان نزدیک به پربارشدن است. بدین ترتیب، بار ترافیکی کنترل‌کننده اصلی نصاب جریان کاهش یافته و دوباره می‌تواند به کارش ادامه دهد.

با ادامه‌یافتن افزایش بار ترافیک ورودی به شبکه، کنترل‌کننده هماهنگ‌کننده با فعال‌کردن تعداد بیشتری از کنترل‌کننده‌های فرعی نصاب جریان و تخصیص اضافه بار ترافیک ورودی به آنها این کار را تکرار می‌کند، تا بدین ترتیب از پربارشدن کنترل‌کننده‌های نصاب جریان جلوگیری شود. از طرف دیگر، با کاهش بار ترافیک ورودی به شبکه، کنترل‌کننده هماهنگ‌کننده ضمن غیر فعال‌کردن کنترل‌کننده فرعی نصاب جریان، تمامی بار ترافیک ورودی به شبکه را به سمت کنترل‌کننده اصلی نصاب جریان هدایت می‌کند. تشریح عملکرد کنترل‌کننده هماهنگ‌کننده و جزئیات واحدهای تشکیل‌دهنده آن در ادامه آمده است.

۴-۲ واحدهای تشکیل‌دهنده کنترل‌کننده هماهنگ‌کننده

کنترل‌کننده هماهنگ‌کننده با استفاده از تکنیک دسته‌بندی آماری ترافیک، ترافیک ورودی به شبکه را دسته‌بندی کرده و بخشی از آن را زمانی که تشخیص داد کنترل‌کننده اصلی نصاب جریان نزدیک به پربارشدن است به کنترل‌کننده فرعی نصاب جریان تخصیص می‌دهد. تشخیص پربارشدن کنترل‌کننده نصاب جریان با بررسی میزان زمان نصب جریان در آن صورت می‌گیرد. با توجه به مقاله نوشته‌شده توسط Md. Faizul Bari و همکارانش [۲۸]، برای این که شبکه SDN عملکرد قابل قبولی داشته باشد حداکثر زمان نصب جریان در شبکه SDN نباید از ۲۰۰ میلی‌ثانیه بیشتر شود. به طور کلی سه روش برای دسته‌بندی ترافیک ورودی به شبکه‌های SDN وجود دارد: ۱) روش دسته‌بندی مبتنی بر درگاه، ۲) روش دسته‌بندی مبتنی بر بار بسته و ۳) روش دسته‌بندی مبتنی بر آمار جریان (روش دسته‌بندی آماری). با توجه به برتری‌های روش دسته‌بندی آماری نسبت به دو روش دیگر که در [۲۹] تا [۳۱] عنوان شده است، از روش دسته‌بندی آماری در کنترل‌کننده هماهنگ‌کننده جهت دسته‌بندی ترافیک ورودی استفاده شده است.

در روش دسته‌بندی آماری، با استفاده از مشخصات جریان (به عنوان مثال: مدت جریان، طول بسته، میزان بایت‌های ارسال‌شده و زمان بین بسته‌های ورودی) که از طریق شمارنده‌های ذاتی OpenFlow به دست می‌آید، ترافیک ورودی به شبکه دسته‌بندی می‌شود. سپس در کنترل‌کننده هماهنگ‌کننده، قوانین جریان پیش‌فرض (قوانین جریانی که دارای پایین‌ترین اولویت مناسب بوده و با تمامی جریان‌های متعلق به یک دسته خاص از ترافیک ورودی مطابقت دارد) منطبق با هر یک از دسته‌های ترافیک ورودی به دست می‌آید. این کار با استفاده از الگوریتم بهبودیافته apriori [۳۲]، از روی قوانین جریان وابسته به هم موجود در آن دسته انجام می‌شود. با توجه به بار ترافیک ورودی به شبکه، کنترل‌کننده هماهنگ‌کننده یک یا چند قانون پیش‌فرض جریان را به کنترل‌کننده(های) فرعی نصاب جریان نسبت می‌دهد تا از این طریق، بار کنترل‌کننده اصلی نصاب جریان کاهش یابد (شکل ۵). در ادامه، واحدهای تشکیل‌دهنده کنترل‌کننده هماهنگ‌کننده تشریح شده‌اند.

- **واحد نظارت و استخراج‌کننده خصوصیات جریان:** این واحد با ارسال متناوب پیام‌های چندبخشی [۲۲] به سوئیچ‌های OpenFlow، اطلاعات آماری آنها را جمع‌آوری می‌کند. با استفاده از اطلاعات آماری به دست آمده از سوئیچ‌ها که شامل اطلاعات آماری مرتبط با هر جریان، مجموعه‌ای از جریان‌ها، جدول جریان و شمارنده‌های متناظر با درگاه‌های سوئیچ‌ها می‌باشد، پایگاه داده شبکه (NIB) ایجاد می‌شود. همچنین، این واحد ویژگی جریان‌های موجود در سوئیچ دروازه را از شمارنده‌های مدخل جریان‌های موجود در آن (شمارنده‌های بیانگر تعداد بسته‌های دریافتی،

تعداد بایت‌های دریافتی و مدت زمان جریان) استخراج می‌کند. با توجه به مطالعات انجام‌شده در [۲۹] و [۳۰]، ویژگی‌های مؤثر جریان‌ها جهت شناسایی ترافیک در مراحل ابتدایی شکل‌گیری آن (با توجه به چند بسته ابتدایی جریان) عبارتند از طول بسته و مشتقات آماری آن (مانند میانگین، حداکثر، انحراف معیار و واریانس)، زمان بین ورود بسته‌ها و میانگین و واریانس آنها و مدت زمان جریان و طول آن بر حسب تعداد بسته‌ها یا تعداد بایت‌ها. به منظور کاهش پیچیدگی محاسباتی و زمانی مورد نیاز جهت شناسایی ترافیک در این واحد، همان گونه که L. Peng و همکارانش در [۲۹] اشاره کرده‌اند، تنها به تعداد کمی از ویژگی‌های آماری جریان (حداکثر دو یا سه ترکیب از آن) نیاز است تا با دقت بالایی ترافیک ورودی شناسایی شود. با توجه به نتایج به دست آمده توسط L. Peng و همکارانش در [۲۹]، در این واحد تنها از طول چهار بسته ابتدایی هر جریان ورودی به همراه میانگین و واریانس آنها به عنوان ویژگی‌های آماری جریان جهت دسته‌بندی آماری ترافیک ورودی به شبکه استفاده شده است.

• **واحد مدیریت کنترل‌کننده‌های نصاب جریان:** کار اصلی این واحد، صدور سیگنال‌هایی جهت فعال و غیر فعال‌کردن کنترل‌کننده‌های فرعی نصاب جریان است. بر اساس اطلاعات دریافتی از واحد نظارت، این واحد تشخیص می‌دهد که کنترل‌کننده اصلی نصاب جریان نزدیک به پربارشدن است و در نتیجه یکی از کنترل‌کننده‌های فرعی نصاب جریان را فعال می‌کند. با افزایش بار ترافیک ورودی به شبکه، این عمل برای یکی دیگر از کنترل‌کننده‌های فرعی نصاب جریان تکرار می‌شود. با کاهش یافتن بار ترافیک ورودی به شبکه، این واحد با صدور سیگنال‌هایی یک یا چند کنترل‌کننده فرعی نصاب جریان فعال را غیر فعال می‌کند. برنامه‌های شبکه که بر روی کنترل‌کننده‌های نصاب جریان اجرا می‌شوند، جهت تصمیم‌گیری مناسب و عملکرد صحیح نیاز به اطلاعات پایگاه داده شبکه (NIB) یا نتایج حاصل از پردازش آن دارند. این واحد پایگاه داده شبکه را از واحد نظارت دریافت کرده و پس از پردازش آن، پایگاه داده پردازش‌شده، بخشی از پایگاه داده پردازش‌شده، یا نتیجه حاصل از پردازش پایگاه داده شبکه را (بر اساس توافق ابتدایی حاصل‌شده بین کنترل‌کننده هماهنگ‌کننده و کنترل‌کننده‌های نصاب جریان) به کنترل‌کننده‌های نصاب جریان ارسال می‌کند.

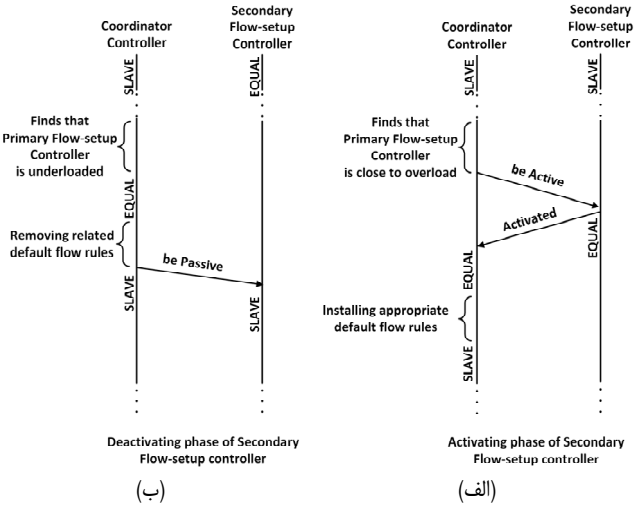
• **واحد دسته‌بندی ترافیک:** این واحد بر اساس اطلاعات دریافتی از واحد نظارت و استخراج‌کننده خصوصیات جریان، ترافیک ورودی به شبکه را دسته‌بندی می‌کند. این دسته‌بندی بر روی فیلد انطباق مداخل جریان‌های ثبت‌شده در واحد نظارت با توجه به خصوصیات آماری استخراج‌شده در آن واحد انجام می‌شود. خروجی واحد دسته‌بندی ترافیک مجموعه‌ای از فیلدهای انطباق مرتبط با هر یک از دسته‌های ترافیک ورودی به شبکه است. این دسته‌بندی زمانی انجام می‌شود که واحد مدیریت کنترل‌کننده‌های نصاب جریان تشخیص دهد که کنترل‌کننده اصلی/ فرعی نصاب جریان نزدیک به پربارشدن است. در این واحد با توجه به یافته‌های L. Peng و همکارانش [۲۹] و همچنین Y. Zhao و همکارانش [۳۳]، از الگوریتم دسته‌بندی J۴۸ [۳۴] جهت دسته‌بندی ترافیک ورودی استفاده شده است. الگوریتم دسته‌بندی J۴۸ فیلد انطباق مداخل جریان‌های ثبت‌شده را با توجه به خصوصیات آماری جریان‌ها (طول چهار بسته ابتدایی از هر جریان ورودی، میانگین و واریانس آنها) دسته‌بندی می‌کند. مجموعه‌ای از فیلدهای انطباق مرتبط با هر یک از دسته‌های ترافیک ورودی به شبکه، خروجی الگوریتم دسته‌بندی J۴۸ می‌باشد.

• **واحد تولید قوانین جریان پیش‌فرض:** وظیفه این واحد تولید قوانین جریان پیش‌فرض متناظر با هر یک از دسته‌های ترافیک ورودی تشخیص داده شده در واحد دسته‌بندی ترافیک برای هدایت بار اضافی ترافیک ورودی به کنترل‌کننده فرعی نصاب جریان انتخاب‌شده است. هر دسته تشخیص داده شده از ترافیک ورودی شامل مجموعه‌ای از فیلدهای انطباق قوانین جریان است که این قوانین جریان از جدول جریان سوئیچ‌های دروازه استخراج شده و مورد بررسی قرار گرفته‌اند. این واحد با استفاده از الگوریتم بهبودیافته apriori [۳۲]، وابستگی بین قوانین جریان در هر دسته از ترافیک ورودی را تشخیص می‌دهد. برای این منظور، این الگوریتم فراوانی وقوع فیلدهای انطباق مشخصی نظیر آدرس IP مبدأ و مقصد، نوع پروتکل انتقال و شماره درگاه انتقال مبدأ و مقصد را اندازه‌گیری کرده و علاوه بر آن ترکیبی از این فیلدها را به عنوان بخشی از قوانین جریان که از فراوانی وقوع بالایی برخوردار هستند در نظر می‌گیرد. با شکستن فیلد آدرس IP مبدأ و مقصد به فیلدهای کوچک‌تر، این الگوریتم می‌تواند از مفهوم subnet masks و wildcard نیز پشتیبانی کند. این واحد برای تولید قوانین جریان پیش‌فرض نیاز به شناسه کنترل‌کننده فرعی نصاب جریان و اولویت مناسبی برای قانون جریان دارد که این اطلاعات را واحد مدیریت کنترل‌کننده‌های نصاب جریان در اختیارش می‌گذارد.

• **واحد تجمیع قوانین جریان پیش‌فرض:** وظیفه این واحد، تجمیع قوانین جریان پیش‌فرض متعلق به هر دسته تشخیص داده شده از ترافیک ورودی است تا بدین ترتیب تعداد قوانین جریان پیش‌فرض مورد نیاز جهت نصب در سوئیچ‌های OpenFlow کاهش یابد [۳۵] تا [۳۷]. در این واحد جهت تجمیع قوانین جریان از الگوریتم FFTA که توسط S. Luo و همکارانش در [۳۶] معرفی شده است استفاده می‌شود. از آنجایی که برای تمامی قوانین پیش‌فرض متعلق به هر دسته از ترافیک ورودی تنها یک نوع عمل در نظر گرفته شده است (به عنوان مثال دورریختن بسته‌های حمله DDoS و ارسال بسته‌های ترافیک FTP به کنترل‌کننده فرعی نصاب جریان)، الگوریتم FFTA با دقت و سرعت بیشتری نسبت به حالت معمول می‌تواند تجمیع قوانین جریان را انجام دهد.

۴-۳ نحوه فعال و غیر فعال کردن کنترل‌کننده فرعی نصاب جریان

در اثر تغییرات نرخ ترافیک ورودی به شبکه، امکان پربارشدن کنترل‌کننده اصلی نصاب جریان وجود دارد. زمانی که کنترل‌کننده هماهنگ‌کننده تشخیص دهد که کنترل‌کننده اصلی نصاب جریان نزدیک به پرباری است، کنترل‌کننده فرعی نصاب جریان را فعال می‌کند (شکل ۶- الف). برای این منظور، کنترل‌کننده هماهنگ‌کننده سیگنال be-Active را به کنترل‌کننده فرعی نصاب جریان ارسال می‌کند. با دریافت سیگنال be-Active، کنترل‌کننده فرعی نصاب جریان پیام OFPT_ROLE_REQUEST را به تمامی سوئیچ‌های OpenFlow متصل به خودش جهت اعلان تغییر حالتش به EQUAL ارسال می‌کند. با دریافت سیگنال OFPT_ROLE_REPLY توسط کنترل‌کننده فرعی نصاب جریان از سوئیچ‌های OpenFlow، این کنترل‌کننده در حالت EQUAL قرار می‌گیرد. کنترل‌کننده هماهنگ‌کننده با دریافت سیگنال Activated از کنترل‌کننده فرعی نصاب جریان، حالتش را از SLAVE به EQUAL تغییر داده و پس از نصب قوانین جریان پیش‌فرض مرتبط با کنترل‌کننده فرعی نصاب جریان فعال‌شده در سوئیچ‌های دروازه، دوباره به حالت SLAVE می‌رود.



شکل ۶: نحوه تغییر حالت کنترل‌کننده فرعی نصاب جریان در SAHAR: (الف) فعال‌شدن و (ب) غیر فعال شدن.

زمانی که کنترل‌کننده هماهنگ‌کننده تشخیص دهد که کنترل‌کننده اصلی نصاب جریان از حالت پرباری خارج شده است مجدداً کنترل‌کننده فرعی نصاب جریان را غیر فعال می‌کند (شکل ۶- ب). برای انجام این کار، کنترل‌کننده هماهنگ‌کننده پس از تغییر حالت از SLAVE به EQUAL، قوانین جریان پیش‌فرض مرتبط با کنترل‌کننده فرعی نصاب جریان فعال را از سوئیچ‌های دروازه حذف کرده و سپس سیگنال be-Passive را به کنترل‌کننده فرعی نصاب جریان مورد نظر ارسال می‌کند. سپس، کنترل‌کننده هماهنگ‌کننده دوباره حالتش را از EQUAL به SLAVE تغییر می‌دهد. با دریافت سیگنال be-Passive، کنترل‌کننده فرعی نصاب جریان پیام‌های درخواست Barrier را به تمامی سوئیچ‌های متصل به خودش ارسال می‌کند تا بدین ترتیب، این سوئیچ‌ها پردازش تمامی پیام‌های دریافتی از کنترل‌کننده فرعی نصاب جریان را به پایان برسانند. پس از آن، کنترل‌کننده فرعی نصاب جریان با ارسال پیام OFPT_ROLE_REQUEST به تمامی سوئیچ‌های متصل به خودش، حالتش را از EQUAL به SLAVE تغییر می‌دهد. به این ترتیب، کنترل‌کننده فرعی نصاب جریان غیر فعال می‌شود.

٤-٤ اعتبارسنجی پروتکل تغییر حالت کنترل‌کننده‌های نصاب قوانین جریان

در پروتکل تغییر حالت کنترل‌کننده‌های اصلی و فرعی توسط کنترل‌کننده هماهنگ‌کننده، سه خاصیت استاندارد زنده‌بودن، ایمنی و اتمام عملیات که در هر پروتکل مهاجرتی برای سوئیچ‌ها باید در نظر گرفته شود رعایت شده است. در این پروتکل، هر سوئیچ OpenFlow در تمامی لحظات حداقل توسط یک کنترل‌کننده فعال که در حالت EQUAL است (کنترل‌کننده اصلی) کنترل می‌شود که این همان خاصیت زنده‌بودن است. به هنگام نصب‌کردن قوانین جریان در جدول جریان سوئیچ‌ها، کنترل‌کننده‌های اصلی و فرعی از پیام Flow-mod که در آن پرچم OFPFF_CHECK_OVERLAP فعال شده است استفاده می‌کنند. این عمل باعث می‌شود که از نصب قوانین جریان دوگانه برای پیام‌های Packet-in یکسانی که به کنترل‌کننده‌های اصلی و فرعی ارسال می‌شود توسط سوئیچ OpenFlow جلوگیری شود که این امر باعث رعایت خاصیت ایمنی می‌شود. قبل از این که کنترل‌کننده فرعی به حالت آماده به کار برود (با رفتن به حالت SLAVE)، با ارسال پیام Barrier-Request به سوئیچ‌های OpenFlow تحت کنترل خود باعث می‌شود که تمامی سوئیچ‌ها پس از اتمام رسیدگی به تمام پیام‌هایی که قبل از این پیام (Barrier-Request) دریافت کرده‌اند پیام Barrier-Reply را به کنترل‌کننده فرعی ارسال کنند. پس از دریافت پیام Barrier-Reply است که کنترل‌کننده فرعی به حالت SLAVE می‌رود که این همان خاصیت اتمام عملیات است که توسط این پروتکل رعایت شده است.

٥- بررسی اثر حمله اشباع‌کردن سطح کنترلی توسط سطح داده بر روی SAHAR

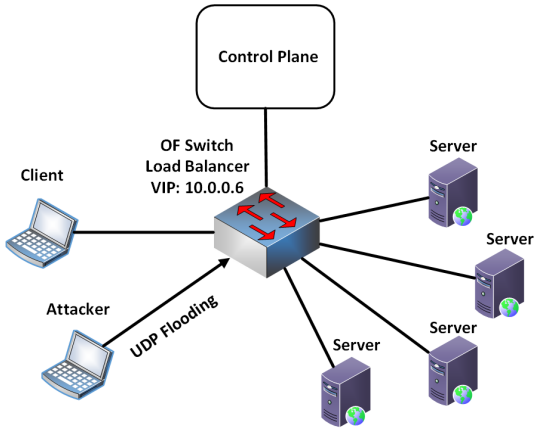
به منظور نشان‌دادن تأثیر افزونگی بر دسترس‌پذیری معماری SAHAR، یک حمله سیلابی UDP بر علیه سه سناریو (سناریوی اول: استفاده از یک کنترل‌کننده، سناریوی دوم: استفاده از کنترل‌کننده هماهنگ‌کننده و یک عدد کنترل‌کننده نصاب قوانین جریان و سناریوی سوم: استفاده از معماری SAHAR) اعمال شده تا بدین ترتیب اثر حمله اشباع‌کردن سطح کنترلی توسط سطح داده مورد ارزیابی قرار گیرد. برای انجام این آزمایش از ساختار نشان داده شده در شکل ۷ استفاده شده که در محیط مقلد Mininet ۲.۲.۲ [۳۸] و [۳۹] پیاده‌سازی گردیده است. در این پیاده‌سازی همان طور که در جدول ۲ آمده است، کنترل‌کننده‌ها در بستر RYU [۴۰] اجرا می‌شوند و سوئیچ‌های OpenFlow نیز با استفاده از سوئیچ‌های نرم‌افزاری OpenVSwitch [۴۱] پیاده‌سازی شده‌اند. همچنین به منظور تولید ترافیک حمله سیلابی UDP جهت ارزیابی سناریوها، از ابزار iPerf [۴۲] استفاده شده است.

در حالی که یک مشتری با سوئیچ OpenFlow که متعادل‌کننده بار بین سرویس‌دهنده‌ها است با سرعت ثابت ۵۰ جریان در ثانیه در حال ارتباط است، حمله‌کننده یک حمله سیلابی UDP به سوئیچ OpenFlow اعمال می‌کند. در این آزمایش، با اندازه‌گیری زمان رفت و برگشت (RTT) درخواست‌های مشتری که با استفاده از اجرای دستور ping به دست می‌آید عملکرد سناریوهای مختلف مورد ارزیابی قرار گرفته است. همان گونه که در شکل ۸ نشان داده شده است، در حالی که سرعت حمله سیلابی UDP به صورت خطی تا ۸۰۰ بسته در ثانیه (PPS) افزایش می‌یابد، زمان رفت و برگشت درخواست‌های مشتری برای هر سه سناریو اندازه‌گیری شده است.

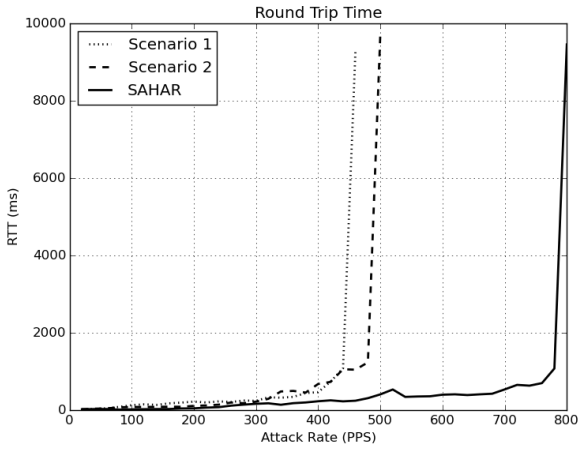
همان گونه که در شکل ۸ دیده می‌شود، SAHAR با رسیدن سرعت حمله به ۸۰۰ بسته در ثانیه اشباع می‌شود، در حالی که این حالت برای سناریوهای ۱ و ۲ به ترتیب در ۴۵۰ و ۵۰۰ بسته در ثانیه اتفاق افتاده است. در حقیقت با رسیدن سرعت حمله به نقطه اشباع هر یک از سناریوها، کنترل‌کننده‌های نصاب قوانین جریان از دسترس خارج شده و دیگر قادر به نصب مسیر جدید بر روی سوئیچ OpenFlow به ازای ورود جریان‌های جدید نیستند. با توجه به نتایج به دست آمده، کاملاً مشهود است که SAHAR در برابر حمله اشباع‌کردن سطح کنترلی بسیار مقاوم بوده و همچنین به دلیل استفاده از افزونگی در کنترل‌کننده‌های نصاب قوانین جریان از دسترس‌پذیری بالایی نسبت به سناریوهای دیگر برخوردار است.

٦- بررسی تأثیر حملات منع سرویس بر روی SAHAR

از عوامل تأثیرگذار بر دسترس‌پذیری شبکه‌های نرم‌افزار محور، حملات منع سرویس است. در این قسمت تأثیر سه نوع حمله منع سرویس بر روی SAHAR مورد بررسی قرار گرفته است. این حملات عبارتند از: ۱) حمله سیلابی ICMP/Ping، ۲) حمله سیلابی UDP و ۳) حمله سیلابی



شکل ۷: ساختار استفاده‌شده در انجام آزمایش حمله سیلابی UDP.



شکل ۸: اندازه‌گیری زمان رفت و برگشت درخواست‌های مشتری برای SAHAR و سناریوهای ۱ و ۲ در طول اعمال حمله سیلابی UDP. (الف) سناریوی اول: استفاده از یک کنترل‌کننده، (ب) سناریوی دوم: استفاده از کنترل‌کننده هماهنگ‌کننده و یک عدد کنترل‌کننده نصاب قوانین جریان و (ج) سناریوی سوم: استفاده از SAHAR.

جدول ۲: پارامترهای آزمایشگاهی جهت ارزیابی SAHAR.

Virtual Machine	Oracle VM VirtualBox (version ۵.۱.۱۰ r۱۱۲۰۲۶)
Guest OS	Ubuntu ۱۶.۰۴
RAM	۱۶ GB
CPU	Intel Core i۷ – ۴۷۲۰HQ CPU@۲٫۶۰ GHz with four cores
Emulator	Mininet ۲.۲.۲ [۳۹] و [۳۸]
Network Operating System	RYU ۴.۱۵ [۴۰]
OpenFlow Switch	OpenVSwitch ۲.۴.۰ [۴۱]
Network Traffic Generator	iPerf ۲.۰.۸b [۴۲]

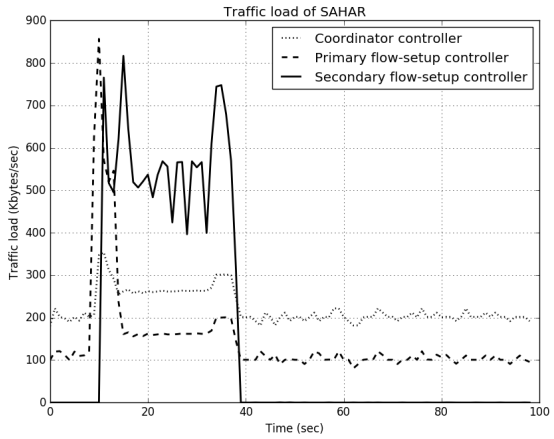
TCP_SYN. برای تولید این حملات که جزء حملات حجمی منع سرویس هستند از ابزار hyenae نسخه ۰.۳۶ [۴۳] استفاده شده است. در این قسمت نیز از ساختار نشان داده شده در شکل ۷ که با استفاده از مقلد Mininet ۲.۲.۲ [۳۸] و [۳۹] پیاده‌سازی شده، جهت انجام آزمایش‌ها استفاده شده است. همچنین در حالی که یک مشتری با سوئیچ OpenFlow (متعادل‌کننده بار بین سرویس‌دهنده‌ها) با سرعت ثابت ۵۰ جریان در ثانیه در حال ارتباط است، حمله‌کننده حملات منع سرویس را به سوئیچ OpenFlow اعمال می‌کند. در این آزمایش، پارامترهای مورد نظر جهت ارزیابی نحوه تأثیرگذاری حملات منع سرویس بر روی عملکرد معماری SAHAR و دسترس‌پذیری آن، میزان بار ورودی به کنترل‌کننده‌های موجود در معماری SAHAR و زمان رفت و برگشت (RTT) درخواست‌های مشتری است. این پارامترها به ترتیب با استفاده از ابزار bwm-ng v۰.۶.۲ [۴۴] و اجرای دستور ping به دست می‌آیند. در ادامه به این حملات و بررسی نتایج اجرای آنها پرداخته شده است.

۶-۱ بررسی تأثیر حمله سیلابی ICMP/Ping

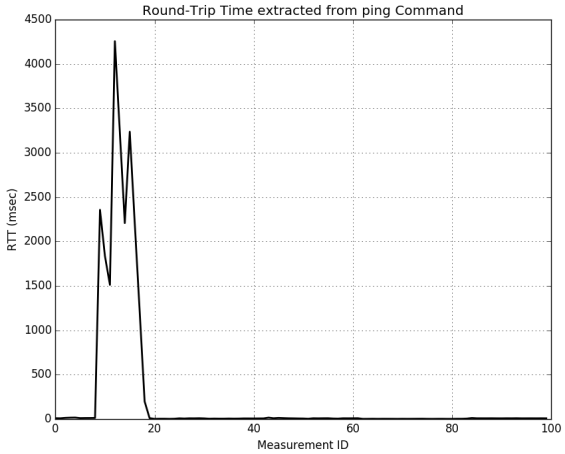
در این نوع حمله، مهاجم، قربانی را با درخواست‌های ICMP echo غرق می‌کند که این امر بر روی سرویس‌های دیگری که توسط سایر برنامه‌ها بر روی قربانی استفاده می‌شود اثر می‌گذارد. برای انجام این نوع حمله، مهاجم باید تعداد زیادی از بسته‌های echo_request را که به نظر می‌رسد از منابع متفاوت با آدرس‌های تصادفی آمده‌اند با نهایت سرعت به سمت قربانی ارسال کند. مهاجم با برخورداری از تعداد زیادی جریان، می‌تواند شبکه را مختل کند. هنگامی که مهاجم از منابع متعدد با آدرس‌های تصادفی برای حمله به قربانی استفاده می‌کند، هر بار بسته packet-in جدیدی تولید می‌شود که این امر منجر به ایجاد ترافیک عظیمی در کنترل‌کننده خواهد شد. کنترل‌کننده با دریافت تعداد زیادی از بسته‌های packet-in شروع به غرق‌شدن کرده و بعد از مدتی شروع به نادیده‌گرفتن بسته‌های ورودی جدید می‌کند. این نوع از حملات را با استفاده از حجم ترافیک تولیدی که بر حسب تعداد بایت‌ها، بسته‌ها یا جریان‌ها بیان می‌شود می‌توان تشخیص داد.

در شکل ۹ نتایج اعمال حمله سیلابی ICMP/Ping به شبکه نرم‌افزار محور مجهز به معماری SAHAR که ساختار آن در شکل ۷ نشان داده شده است آورده شده است. شکل ۹- الف میزان بار ترافیک ورودی به معماری SAHAR را قبل از حمله، در حین حمله و پس از آن نشان می‌دهد. همچنین، شکل ۹- ب زمان رفت و برگشت درخواست‌های مشتری را نشان می‌دهد. همان طور که مشاهده می‌شود، حمله سیلابی ICMP/Ping در ثانیه ۱۰ شروع شده و در ثانیه ۴۰ خاتمه می‌یابد. با شروع حمله، در اثر پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان و عدم توانایی آن در رسیدگی به درخواست‌های واردشده در قالب بسته‌های packet-in، زمان رفت و برگشت درخواست‌های مشتری به شدت افزایش می‌یابد.

با فعال‌شدن کنترل‌کننده فرعی نصاب قوانین جریان توسط کنترل‌کننده هماهنگ‌کننده به دلیل تشخیص پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان و تخصیص بار اضافی ناشی از حمله صورت‌گرفته به آن، میزان بار ترافیک ورودی به کنترل‌کننده اصلی نصاب قوانین جریان کاهش می‌یابد. با کاهش‌یافتن میزان بار ترافیک ورودی به کنترل‌کننده اصلی نصاب قوانین جریان، این کنترل‌کننده دوباره قادر به رسیدگی و پردازش درخواست‌های رسیده در قالب بسته‌های packet-in بوده و در

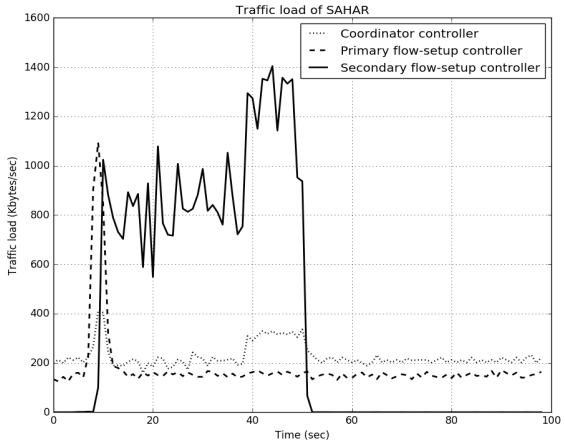


(الف)

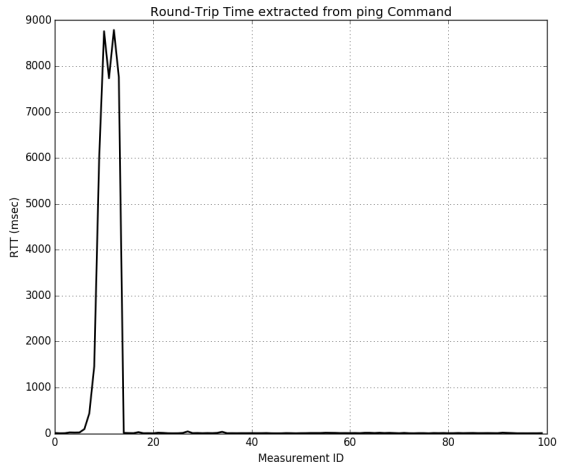


(ب)

شکل ۹: (الف) میزان بار ترافیک ورودی به معماری SAHAR قبل از حمله سیلابی ICMP/Ping، در حین حمله و پس از آن و (ب) زمان رفت و برگشت (RTT) درخواست‌های مشتری.



(الف)



(ب)

شکل ۱۰: (الف) میزان بار ترافیک ورودی به معماری SAHAR قبل از حمله سیلابی UDP، در حین حمله و پس از آن و (ب) زمان رفت و برگشت (RTT) درخواست‌های مشتری.

نتیجه زمان رفت و برگشت درخواست‌های مشتری به مقدار معقول خود برمی‌گردد. این آزمایش نشان می‌دهد که دسترس‌پذیری معماری SAHAR به جز زمان بسیار کوتاهی (زمان لازم برای تشخیص پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان توسط کنترل‌کننده هماهنگ‌کننده و به دنبال آن فعال‌شدن کنترل‌کننده فرعی نصاب قوانین جریان و تخصیص بار اضافی به آن توسط کنترل‌کننده هماهنگ‌کننده) تحت تأثیر حمله سیلابی ICMP/Ping قرار نمی‌گیرد.

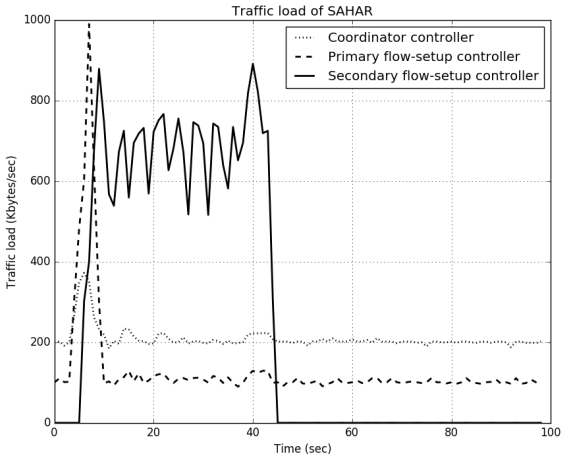
۶-۲ بررسی تأثیر حمله سیلابی UDP

حمله سیلابی UDP تلاش می‌کند تا ترافیک عظیمی از بسته‌های UDP را به قربانی بفرستد. زمانی که حمله سیلابی UDP توسط یک میزبان انجام می‌شود، از تعداد زیادی آدرس‌های IP مبدأ جعلی استفاده می‌شود. ورود تعداد زیادی از بسته‌های UDP به قربانی منجر به سرریزشدن بافر در قربانی می‌شود.

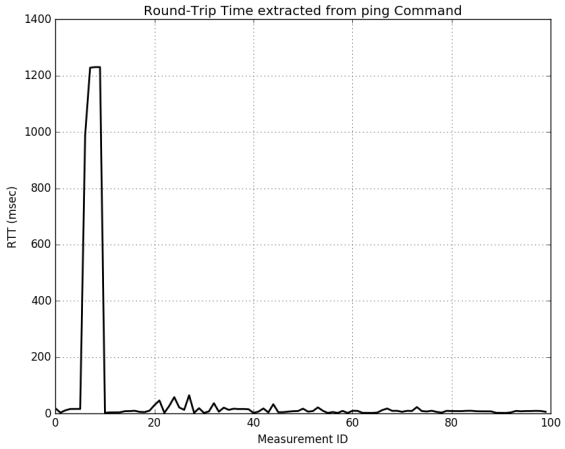
مشخصه اصلی حمله سیلابی UDP ارسال حجم عظیمی از بایت‌ها به قربانی است و بنابراین، این حمله بیشترین میزان ترافیک را در بین حملات منع سرویس ایجاد می‌کند. در این حمله، بسته‌های UDP با استفاده از آدرس‌های IP مبدأ جعلی ایجاد می‌شوند که به نظر می‌رسد تعداد زیادی جریان جدید در ثانیه به قربانی ارسال می‌شود. از این رو، پس از مدت معینی کنترل‌کننده شبکه نرم‌افزار محور شروع به ریزش جریان‌های جدید می‌کند.

در شکل ۱۰ نتایج اعمال حمله سیلابی UDP به شبکه نرم‌افزار محور مجهز به معماری SAHAR که ساختار آن در شکل ۷ نشان داده شده آمده است. شکل ۱۰- الف میزان بار ترافیک ورودی به معماری SAHAR را قبل از حمله، در حین حمله و پس از آن نشان می‌دهد. همچنین، شکل ۱۰- ب زمان رفت و برگشت درخواست‌های مشتری را نشان می‌دهد. همان طور که مشاهده می‌شود، حمله سیلابی UDP در ثانیه ۱۰ شروع شده و در ثانیه ۵۰ خاتمه می‌یابد. با شروع حمله، در اثر پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان و عدم توانایی آن در رسیدگی به درخواست‌های واردشده در قالب بسته‌های packet-in، زمان رفت و برگشت درخواست‌های مشتری به شدت افزایش می‌یابد. این افزایش زمان همان طور که در شکل ۱۰- ب دیده می‌شود، به دلیل بیشتربودن حجم ترافیک ورودی به کنترل‌کننده شبکه نرم‌افزار محور در مقایسه با حمله سیلابی ICMP/Ping بیشتر است.

با فعال‌شدن کنترل‌کننده فرعی نصاب قوانین جریان توسط کنترل‌کننده هماهنگ‌کننده به دلیل تشخیص پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان و تخصیص بار اضافی ناشی از حمله صورت‌گرفته به آن، میزان بار ترافیک ورودی به کنترل‌کننده اصلی نصاب قوانین جریان کاهش می‌یابد. با کاهش‌یافتن میزان بار ترافیک ورودی به کنترل‌کننده اصلی نصاب قوانین جریان، این کنترل‌کننده دوباره قادر به رسیدگی و پردازش درخواست‌های رسیده در قالب بسته‌های packet-in بوده و در نتیجه زمان رفت و برگشت درخواست‌های مشتری به مقدار معقول خود



(الف)



(ب)

شکل ۱۱: (الف) میزان بار ترافیک ورودی به معماری SAHAR قبل از حمله سیلابی TCP_SYN، در حین حمله و پس از آن و (ب) زمان رفت و برگشت (RTT) درخواست‌های مشتری.

برمی‌گردد. بنابراین در این حالت نیز همانند حمله سیلابی ICMP/Ping، دسترس‌پذیری معماری SAHAR به جزء زمان بسیار کوتاهی (زمان لازم برای تشخیص پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان توسط کنترل‌کننده هماهنگ‌کننده و به دنبال آن فعال‌شدن کنترل‌کننده فرعی نصاب قوانین جریان و تخصیص بار اضافی به آن توسط کنترل‌کننده هماهنگ‌کننده) تحت تأثیر حمله سیلابی UDP قرار نمی‌گیرد.

۶-۳ بررسی تأثیر حمله سیلابی TCP_SYN

در حمله سیلابی TCP_SYN، قربانی با درخواست‌های ساختگی SYN که با استفاده از IPهای مبدأ جعلی ایجاد شده‌اند غرق می‌شود. از آنجایی که این IPهای مبدأ جعل شده‌اند، قربانی هرگز پاسخی برای بسته‌های SYN/ACK خود دریافت نمی‌کند. بنابراین، درگاهی که در این حمله درگیر شده است بی‌جهت باز می‌ماند. با تعداد زیادی از درخواست‌های ساختگی SYN، تمام درگاه‌های قربانی مسدود شده و دیگر قربانی قادر به اتصال با کاربران قانونی نمی‌باشد. این نوع حمله به ترافیک زیادی جهت در دسترس نبودن قربانی نیاز ندارد، فقط باز نگه داشتن اتصالات جعلی کار را انجام می‌دهد.

جهت ایجاد حمله سیلابی TCP_SYN از ابزار hyenae نسخه ۰.۳۶ [۴۳] استفاده شده که این ابزار شروع به ارسال تعداد زیادی از جریان‌های با سرعت پایین حاوی بسته‌های TCP_SYN به سمت قربانی می‌کند. همچنین برای این که فقط تأثیر حمله سیلابی TCP_SYN بر روی کنترل‌کننده شبکه نرم‌افزار محور مورد بررسی قرار گیرد، سوئیچ OpenFlow متعادل‌کننده بار، بار ترافیک ورودی را تنها بین سه سرویس‌دهنده توزیع کرده و از سرویس‌دهنده چهارم برای بررسی زمان رفت و برگشت (RTT) درخواست‌های مشتری استفاده شده است.

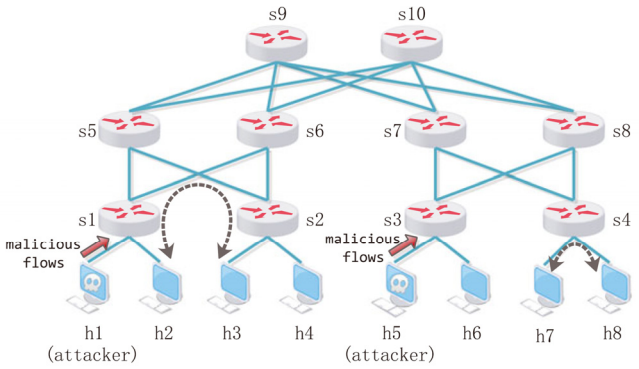
در این حمله، از آنجا که از IPهای مبدأ تصادفی برای تولید جریان به سمت قربانی استفاده می‌شود، تعداد زیادی جریان توسط کنترل‌کننده شبکه نرم‌افزار محور مشاهده می‌شود. این امر باعث ورود بار ترافیکی عظیمی به کنترل‌کننده شبکه نرم‌افزار محور شده و عملکرد آن را مختل می‌کند.

شکل ۱۱ نتایج اعمال حمله سیلابی TCP_SYN به شبکه نرم‌افزار محور مجهز به معماری SAHAR را که ساختار آن در شکل ۷ نشان داده شده است نمایش می‌دهد. شکل ۱۱- الف میزان بار ترافیک ورودی به معماری SAHAR را قبل از حمله، در حین حمله و پس از آن نشان می‌دهد. همچنین شکل ۱۱- ب زمان رفت و برگشت درخواست‌های مشتری را نشان می‌دهد. همان طور که مشاهده می‌شود، حمله سیلابی TCP_SYN در ثانیه ۳ شروع شده و در ثانیه ۴۵ خاتمه می‌یابد. با شروع حمله، در اثر پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان و عدم توانایی آن در رسیدگی به درخواست‌های واردشده در قالب بسته‌های packet-in، زمان رفت و برگشت درخواست‌های مشتری به شدت افزایش می‌یابد. این افزایش زمان همان طور که در شکل ۱۱- ب دیده می‌شود، به دلیل کمتربودن حجم ترافیک ورودی به کنترل‌کننده شبکه نرم‌افزار محور در مقایسه با حمله سیلابی UDP خیلی کمتر است.

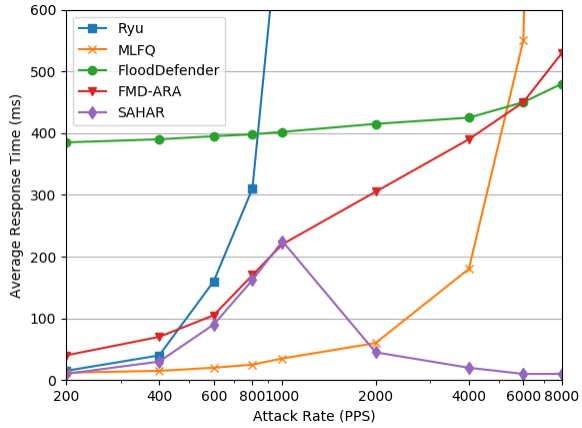
با فعال‌شدن کنترل‌کننده فرعی نصاب قوانین جریان توسط کنترل‌کننده هماهنگ‌کننده به دلیل تشخیص پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان و تخصیص بار اضافی ناشی از حمله صورت‌گرفته به آن، میزان بار ترافیک ورودی به کنترل‌کننده اصلی نصاب قوانین جریان کاهش می‌یابد. با کاهش‌یافتن میزان بار ترافیک ورودی به کنترل‌کننده اصلی نصاب قوانین جریان، این کنترل‌کننده دوباره قادر به رسیدگی و پردازش درخواست‌های رسیده در قالب بسته‌های packet-in بوده و در نتیجه زمان رفت و برگشت درخواست‌های مشتری به مقدار معقول خود برمی‌گردد. بنابراین در این حالت نیز همانند حملات سیلابی ICMP/Ping و UDP، دسترس‌پذیری معماری SAHAR به جز زمان بسیار کوتاهی (زمان لازم برای تشخیص پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان توسط کنترل‌کننده هماهنگ‌کننده و به دنبال آن فعال‌شدن کنترل‌کننده فرعی نصاب قوانین جریان و تخصیص بار اضافی به آن توسط کنترل‌کننده هماهنگ‌کننده) تحت تأثیر حمله سیلابی TCP_SYN قرار نمی‌گیرد.

۷- بررسی کارایی SAHAR در مقایسه با راهکارهای موجود

در این بخش، کارایی معماری SAHAR در برابر حملات منع سرویس با چهار راهکار دیگر مقایسه شده است. این چهار راهکار عبارتند از: ۱) کنترل‌کننده Ryu [۴۰] بدون هر گونه ساز و کار محافظتی، ۲) کنترل‌کننده Ryu مجهز به ساز و کار MLFQ [۱۸]، ۳) کنترل‌کننده Ryu مجهز به ساز و کار FloodDefender [۱۹] و ۴) کنترل‌کننده Ryu مجهز به ساز و کار FMD-ARA [۲۰]. این مقایسه بر اساس روشی که توسط Pengpeng Wu و همکارانش [۲۰] انجام داده‌اند صورت گرفته است. ساختار شبکه استفاده‌شده برای انجام این مقایسه در شکل ۱۲ نشان داده شده که این شبکه در مقلد Mininet ۲.۲.۲ [۳۸] و [۳۹]



شکل ۱۲: ساختار شبکه متشکل از سوئیچ‌های OpenFlow جهت بررسی کارایی SAHAR در برابر حمله منع سرویس [۲۰].

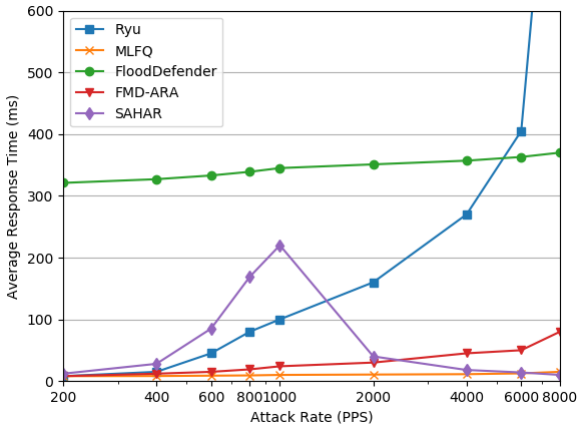


شکل ۱۳: میانگین زمان پاسخ در برقراری ارتباط بین h۲ و h۳.

پیاده‌سازی شده است. در این آزمایش، برخلاف آزمایش‌های قبلی انجام‌شده که در آنها سطح کنترلی دارای شبکه in-band (استفاده از شبکه سطح داده برای ارتباط با سوئیچ‌ها و دیگر کنترل‌کننده‌ها) بود، شبکه سطح کنترلی به صورت out-of-band (شبکه‌ای مجزا از شبکه سطح داده) است.

همان طور که در شکل ۱۲ دیده می‌شود، در سناریوی در نظر گرفته شده برای انجام آزمایش، در حالی که h۱ و h۵ به عنوان حمله‌کننده اقدام به حمله سیلابی UDP می‌کنند و مرتباً سرعت این حمله را افزایش می‌دهند، h۲ با h۳ و h۷ با h۸ ارتباط برقرار می‌کنند. در این حین، زمان درخواست پاسخ (RRT) (میانگین زمان پاسخ از کنترل‌کننده که بیانگر کارایی کنترل‌کننده در ایجاد ارتباط شبکه‌ای است) در ارتباط h۲ با h۳ و h۷ با h۸ به عنوان معیاری جهت ارزیابی کارایی هر یک از راهکارها اندازه‌گیری می‌شود.

زمان درخواست پاسخ در ارتباط h۲ با h۳ در شکل ۱۳ نشان می‌دهد که SAHAR در برابر حمله منع سرویس کارا است. با افزایش سرعت حمله و رسیدن آن به ۱۰۰۰ بسته در ثانیه (PPS)، کنترل‌کننده فرعی نصاب قوانین جریان توسط کنترل‌کننده هماهنگ‌کننده فعال شده و ترافیک حاصل از حمله سیلابی UDP به آن ارسال می‌شود. این عمل باعث کاهش بار ترافیکی کنترل‌کننده اصلی نصاب قوانین جریان شده و همان گونه که در شکل ۱۳ مشخص است، میانگین زمان پاسخ به درخواست‌های قانونی و بی‌خطر جهت برقراری جریان به شدت کاهش می‌یابد. با توجه به نتایج نشان داده شده در شکل ۱۳ که از مقاله نوشته‌شده توسط Pengpeng Wu و همکارانش [۲۰] آورده شده است، در حالی که چهار راهکار دیگر با افزایش سرعت حمله، میانگین زمان پاسخ در آنها افزایش یافته و بدین ترتیب کارایی خود را از دست می‌دهند،



شکل ۱۴: میانگین زمان پاسخ در برقراری ارتباط بین h۷ و h۸.

SAHAR با بهره‌گیری از ساختار خاص خود (وجود کنترل‌کننده‌های فرعی نصاب قوانین جریان در کنار کنترل‌کننده‌های اصلی و هماهنگ‌کننده) سطح کنترلی را در برابر حمله منع سرویس مقاوم کرده و کارایی بهتری از خود نشان می‌دهد.

شکل ۱۴ میانگین زمان پاسخ در برقراری ارتباط بین h۷ و h۸ را نشان می‌دهد. در این حالت برخلاف حالت قبل، ارتباط بین دو میزبانی برقرار شده که به سوئیچی که تحت حمله قرار گرفته است متصل نیستند. در این حالت، تغییر محسوسی در کارایی SAHAR نسبت به حالت قبل دیده نمی‌شود. این در حالی است که چهار راهکار دیگر به غیر از راهکار FloodDefender که مبتنی بر سیستم پیش‌پردازش بوده و نیاز دارد تا قبل از پاسخ به هر درخواستی، درخواست جدید OpenFlow را در بافر بسته‌های packet_in موقتاً نگهداری کند، کارایی خیلی بهتری نسبت به حالت قبل داشته باشند.

این وضعیت به این دلیل است که در سه راهکار Ryu، MLFQ و FMD-ARA بروز ازدحام در کانال ارتباطی سوئیچ با کنترل‌کننده، عامل اصلی افزایش تأخیر در زمان پاسخ است. در صورتی که در SAHAR، با بروز ازدحام در کانال ارتباطی سوئیچ با کنترل‌کننده اصلی نصاب قوانین جریان و پربارشدن کنترل‌کننده اصلی که عامل افزایش تأخیر در زمان پاسخ است، بار اضافی که باعث به وجود آمدن ازدحام شده است از طریق کانال ارتباطی مجزا به یک کنترل‌کننده فرعی نصاب قوانین جریان هدایت می‌شود. کنترل‌کننده هماهنگ‌کننده با تشخیص پربارشدن کنترل‌کننده اصلی نصاب قوانین جریان، کنترل‌کننده فرعی نصاب قوانین جریان را فعال می‌کند که این امر نه تنها باعث کم‌بارشدن کنترل‌کننده اصلی می‌شود، بلکه کانال ارتباطی مجزایی بین کنترل‌کننده فرعی و سوئیچ برقرار می‌شود تا از بروز ازدحام در کانال ارتباطی سوئیچ با کنترل‌کننده اصلی نیز جلوگیری شود.

۸- نتیجه‌گیری

در این مقاله با هدف افزایش دسترس‌پذیری شبکه‌های نرم‌افزار محور از طریق مقاوم‌کردن سطح کنترلی آن، معماری جدیدی برای سطح کنترلی با نام SAHAR معرفی شده است. معماری SAHAR از یک جعبه کنترلی تشکیل گردیده است. این جعبه کنترلی شامل یک کنترل‌کننده هماهنگ‌کننده، یک کنترل‌کننده اصلی نصاب قوانین جریان و یک یا چند کنترل‌کننده فرعی نصاب قوانین جریان است که بر حسب نیاز فعال می‌شوند. وظیفه کنترل‌کننده هماهنگ‌کننده جمع‌آوری اطلاعات آماری از سوئیچ‌های OpenFlow جهت نظارت بر شبکه و مدیریت

کنترل‌کننده‌های نصاب قوانین جریان است. وظیفه نصب قوانین جریان نیز بر عهده کنترل‌کننده‌های نصاب قوانین جریان است. این قوانین جریان توسط برنامه‌های شبکه‌ای که بر روی کنترل‌کننده‌های نصاب قوانین جریان در حال اجرا هستند با توجه به اطلاعات پایگاه داده شبکه (NIB) که از کنترل‌کننده هماهنگ‌کننده می‌آید تولید می‌شوند. کنترل‌کننده هماهنگ‌کننده نیز بر اساس بار ترافیکی ورودی به شبکه، عملکرد کنترل‌کننده‌های نصاب اصلی و فرعی قوانین جریان را هماهنگ می‌کند.

در معماری SAHAR با استفاده از قابلیت چند کنترل‌کننده معرفی‌شده در نسخه ۱.۲ پروتکل OpenFlow و عملکردهای Nicira ضمیمه‌شده به سیستم عامل شبکه RYU، نه تنها وظایف سطح کنترلی شبکه SDN بین کنترل‌کننده‌های هماهنگ‌کننده و نصاب قوانین جریان تقسیم شده است بلکه ترافیک ورودی به سطح کنترلی نیز بین کنترل‌کننده‌های نصاب قوانین جریان توزیع می‌شود.

با توجه به این که در معماری SAHAR، افزونگی کنترل‌کننده‌های نصاب قوانین جریان وجود دارد، تحمل‌پذیری SAHAR در برابر حملاتی که منجر به اشباع سطح کنترلی توسط سطح داده شبکه‌های SDN می‌شود زیاد است. همچنین، از آنجایی که معماری SAHAR قادر به دسته‌بندی ترافیک ورودی به شبکه است، با تشخیص ترافیک حمله، رسیدگی به آن را بر عهده یکی از کنترل‌کننده‌های فرعی نصاب قوانین جریان می‌گذارد. در این حال، نصب قوانین جریان توسط کنترل‌کننده اصلی و یک یا چند کنترل‌کننده فرعی نصاب قوانین جریان انجام می‌شود و بدین ترتیب در دسترس‌پذیری آن خللی ایجاد نمی‌شود.

در SAHAR با ارسال‌شدن ترافیک مشکوک به حمله به یکی از کنترل‌کننده‌های فرعی نصاب قوانین جریان، این ترافیک می‌تواند مورد بررسی دقیق‌تر قرار گرفته و در صورت تشخیص حمله‌بودن آن با نصب قوانین جریان مناسب بر روی سوئیچ(های) ارسال‌کننده ترافیک از ورود آن ترافیک به سطح کنترلی جلوگیری شود. بدین ترتیب از اشباع‌شدن کنترل‌کننده و در نهایت سطح کنترلی جلوگیری می‌شود.

با این که SAHAR نشان داده که در برابر حملات منع سرویس در مقایسه با راهکارهای قبلی (مانند: MLFQ، FloodDefender و FMD-ARA) از کارایی بهتری برخوردار است، اما دسترسی به این کارایی به قیمت استفاده از منابع بیشتر که منجر به افزایش توان مصرفی و هزینه می‌شود حاصل شده است. همچنین محدودیت در اضافه‌کردن کنترل‌کننده‌های فرعی نصاب قوانین جریان به دلیل محدودبودن تعداد درگاه‌های سوئیچ‌ها نیز یکی دیگر از نقاط ضعف این معماری است. با این وجود، SAHAR با بهره‌گیری از ساختار چند کنترل‌کننده و وجود افزونگی در سطح کنترلی به خوبی توانسته تا با مقاوم‌کردن سطح کنترلی، دسترس‌پذیری شبکه نرم‌افزار محور را در برابر افزایش بار ترافیک ورودی و به خصوص ترافیک حاصل از حملات منع سرویس افزایش دهد.

مراجع

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

[2] H. Farhady, H. Lee, and N. Akihiro, "Software-defined networking: a survey," *Computer Networks*, vol. 81, pp. 79-95, 2015.

[3] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: a survey," *IEEE Commun Surveys Tutorials*, vol. 17, no. 4, pp. 2317-2346, 4th Quarter 2015.

[4] W. Li, W. Meng, and L. F. Kwok, "A survey on OpenFlow-based software defined networks: security challenges and countermeasures," *J. Network Comput Appl.*, vol. 68, pp. 126-139, 2016.

[5] J. Benabbou, K. Elbaamrani, and N. Idboufker, "Security in OpenFlow-based SDN, opportunities and challenges," *Photon Netw Commun*, vol. 37, no. 1, pp. 1-23, 2019.

[6] S. Shin and G. Gu, "Attacking software-defined networks: a first feasibility study," in *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, *HotSDN '13,* pp. 165-166, New York, NY, USA, Aug. 2013.

[7] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments," *IEEE Access*, vol. 7, pp. 80813-80828, 2019.

[8] R. Swami, M. Dave, and V. Ranga, "Software defined networking based DDoS defense mechanisms," *ACM Computing Surveys*, vol. 52, no. 2, Article No.: 28, May 2019.

[9] Open Networking Foundation, *OpenFlow Switch Specification*, version 1.2, Dec. 2011.

[10] H. Wang, L. Xu, and G. Gu, OF-GUARD: a DoS attack prevention extension in software-defined networks. Poster presented at: The Open Networking Summit 2014; 2014; Santa Clara, CA.

[11] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks," in *Proc. ACM SIGSAC Conf. on Computer & Communications Security, CCS'13,* pp. 413-424, New York, NY, USA, Nov. 2013.

[12] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "LineSwitch: tackling control plane saturation attacks in software-defined networking," *IEEE/ACM Trans. Networking*, vol. 25, no. 2, pp. 1206-1219, Apr. 2017.

[13] K. Y. Chen, A. R. Junuthula, I. K. Siddhrau, Y. Xu, and H. J. Chao, "SDNShield: towards more comprehensive defense against DDoS attacks on SDN control plane," in *Proc. IEEE Conf. on Communications and Network Security, CNS'16,* pp. 28-36, Philadelphia, PA, USA, 17-19 Oct. 2016.

[14] H. Wang, L. Xu, and G. Gu, "FloodGuard: a DoS attack prevention extension in software-defined networks," in *Proc. 45th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks,* pp. 239-250, Rio de Janeiro, Brazil, 22-25 Jun. 2015.

[15] L. Wei and C. Fung, "FlowRanger: a request prioritizing algorithm for controller DoS attacks in software defined networks," in *Proc. IEEE Int. Conf. on Communications, ICC'15,* pp. 5254-5259, London, UK, 8-12 Jun. 2015.

[16] T. Wang and H. Chen, "SGuard: a lightweight SDN safe-guard architecture for DoS attacks," *China Commun.*, vol. 14, no. 6, pp. 113-125, 2017.

[17] S. Lim, S. Yang, Y. Kim, S. Yang, and H. Kim, "Controller scheduling for continued SDN operation under DDoS attacks," *Electron Lett.*, vol. 51, no. 16, pp. 1259-1261, Aug. 2015.

[18] P. Zhang, H. Wang, C. Hu, and C. Lin, "On denial of service attacks in software defined networks," *IEEE Network*, vol. 30, no. 6, pp. 28-33, Nov./Dec. 2016.

[19] S. Gao, Z. Peng, B. Xiao, A. Hu, and K. Ren, "FloodDefender: protecting data and control plane resources under SDN-aimed DoS attacks," in *Proc. INFOCOM IEEE Conf. on Computer Communications,* 9 pp., Atlanta, GA, USA, 1-4 May 2017.

[20] P. Wu, L. Yao, C. Lin, G. Wu, and M. S. Obaidat, "FMD: a DoS mitigation scheme based on flow migration in software-defined-networking," *Int. J. Commun. Syst.*, vol. 31, no. 9, Article No.: e3543, Jun. 2018.

[21] Y. Wang, T. Hu, G. Tang, J. Xie, and J. Lu, "SGS: safe-guard scheme for protecting control plane against DDoS attacks in software-defined networking," *IEEE Access*, vol. 7, pp. 34699-34710, 2019.

[22] Open Networking Foundation, OpenFlow Switch Specification, Version 1.3.5, Mar. 2015.

[23] https://se.mathworks.com/products/new_products/release2018b.html (Accessed Apr. 2019)

[24] Y. Han, J. H. Yoo, and J. Won-Ki Hong, "Poisson shot-noise process based flow-level traffic matrix generation for data center networks," in *Proc. IFIP/IEEE Int. Symp. on Integrated Network Management, IM'15*, pp. 450-457, Ottawa, Canada, 11-15 May 2015.

[25] B. Y. Yu, G. Yang, and C. Yoo, "Comprehensive prediction models of control traffic for SDN controllers," in *Proc. IEEE Int. Conf. on Network Softwarization (NetSoft)-Technical Sessions*, pp. 62-266, Montreal, Canada, 25-29 Jun. 2018.

[26] K. Kuroki, *et al.*, "Redundancy method for highly available openflow controller," *International J. on Advances in Internet Technology*, vol. 7, no. 1-2, pp. 114-123, 2014.

[27] http://ryu.readthedocs.io/en/latest/nicira_ext_ref.html, (Accessed Apr. 2019).

[28] M. Faizul Bari, *et al*., "Dynamic controller provisioning in software defined networks," in *Proc. of the 9th Int.l Conf. on Network and Service Management, CNSM'13*, pp. 18-25, Zurich, Switzerland, 14-18 Oct. 2013.

[29] L. Peng, B. Yang, Y. Chen, and Z. Chen, "Effectiveness of statistical features for early stage internet traffic identification," *Int. J. Parallel Program.*, vol. 44, no. 1, pp. 181-197, 2016.

[30] A. S. da Silva, C. C. Machado, R. V. Bisol, L. Z. Granville, and A. Schaeffer-Filho, "Identification and selection of flow features for accurate traffic classification in SDN," in *Proc. IEEE 14th Int. Symp. Netw. Comput. Appl.,* pp. 134-141, Cambridge, MA, USA, 28-30 Sept. 2015.

[31] M. Hayes, B. Ng, A. Pekar, and W. K. G. Seah, "Scalable architecture for SDN traffic classification," *IEEE Systems J.*, vol. 12, no. 4, pp. 3203-3214, Dec. 2017.

[32] M. Al-Maolegi and B. Arkok, "An improved apriori algorithm for association rules," *International J. on Natural Language Computing*, vol. 3, no. 1, pp. 22-29, Feb. 2014.

[33] Y. Zhao and Y. Zhang, "Comparison of decision tree methods for finding active objects," *Advances in Space Research*, vol. 41, no. 12, pp. 1955-1959, 2008.

[34] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[35] N. Shelly, E. J. Jackson, T. Koponen, N. McKeown, and J. Rajahalme, "Flow caching for high entropy packet fields," *ACM SIGCOMM Computer Communication Review.*, vol. 44, no. 4, pp. 151-156, Oct. 2014.

[36] S. Luo, H. Yu, and L. M. Li, "Fast incremental flow table aggregation in SDN," in *Proc. 23rd Int. Conf. Computer Communication and Networks, ICCCN'14*, 8 pp., Shanhai, China, 4-7 Aug. 2014.

[37] M. Rifai, *et al.*, "Too many SDN rules? Compress them with MINNIE," in *Proc. IEEE Global Communications Conference, GLOBECOM'15*, 7 pp., San Diego, CA, USA, 6-10 Dec. 2015.

[38] Mininet Team 2018, "Mininet: an Instant Virtual Network on your Laptop (or other PC)," Available: http://mininet.org/

[39] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks*Article No.: 19, 6 pp., Oct. 2010.

[40] RYU SDN Framework, *Ryubook 1.0 Documentation*, Available: http://osrg.github.io/ryu/ (Accessed Apr. 2019).

[41] B. Pfaff, *et al*., "Extending networking into the virtualization layer," in *Proc. of the ACM SIGCOMM HotNets Workshop*, 6 pp., 2009.

[42] J. Dugan, *et al*., "Iperf-The TCP, UDP and SCTP network bandwidth measurement tool," [Online]. Available: https://iperf.fr/

[43] Robin Richter, *Hyenae*, Available: https:// sourceforge.net/projects/hyenae/ (Dec. 2010)

[44] bwm-ng v0.6.2 Copyright (C) 2004-2019 Volker Gropp (bwmng@gropp.org) http://www.gropp.org/?id=projects&sub=bwm-ng, Available: https://github.com/vgropp/bwm-ng

مهران شتابی در سال ۱۳۷۵ مدرک کارشناسی مهندسی کامپیوتر- سخت‌افزار خود را از دانشگاه صنعتی اصفهان و در سال ۱۳۸۰ مدرک کارشناسی ارشد مهندسی کامپیوتر- معماری سیستم‌های کامپیوتری خود را از دانشگاه علم و صنعت ایران دریافت نمود. از سال ۱۳۸۲ تا کنون نامبرده به عنوان عضو هیأت علمی در دانشکده مهندسی کامپیوتر دانشگاه یزد مشغول به فعالیت است. نام‌برده هم‌اکنون دانشجوی دکترای مهندسی کامپیوتر- شبکه‌های کامپیوتری دانشگاه علم و صنعت ایران می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: شبکه‌های کامپیوتری، شبکه‌های کامپیوتری نرم‌افزار محور و امنیت شبکه‌های کامپیوتری.

احمد اکبری تحصیلات خود را کارشناسی الکترونیک و کارشناسی ارشد مهندسی مخابرات در سال‌های ۱۳۶۶ و ۱۳۶۹ در دانشگاه صنعتی اصفهان به پایان رساند و دکترای خود را در زمینه پردازش سیگنال از دانشگاه رن در فرانسه در سال ۱۹۹۵ اخذ نمود. ایشان قبل از اخذ دکترا موفق به اخذ مدرک کارشناسی ارشد در زمینه شبکه‌های کامپیوتری از دانشگاه پلی تکنیک تولوز گردید. ایشان از سال ۱۳۷۴ تا کنون عضو هیأت علمی دانشکده کامپیوتر دانشگاه علم و صنعت ایران بوده است و آزمایشگاه‌های تحقیقاتی مختلفی از جمله مرکز تحقیقات فناوری اطلاعات دانشگاه را راه اندازی و مدیریت کرده است و از سال ۱۳۹۳ تا ۱۳۹۹ ریاست دانشکده مهندسی کامپیوتر دانشگاه را به عهده داشته‌است. ایشان بیش از ۱۰۰ مقاله در مجلات و کنفرانس‌های معتبر علمی منتشر کرده‌است و هم‌اکنون عضو هیأت مدیره انجمن کامپیوتر ایران و نیز دبیر قطب علمی شبکه‌های ارتباطی و اطلاعاتی نسل جدید در دانشگاه علم و صنعت ایران است. زمینه‌های تحقیقاتی ایشان شبکه‌های کامپیوتری، امنیت شبکه و پردازش سیگنال می‌باشد.