ارائه روشی توزیعی و پایدار برای توازن بار بین کنترل‌کننده‌ها در شبکه‌های مبتنی بر نرم‌افزار

بنت‌الهدی احمدی قاجاری [۱]، کارشناسی ارشد؛ زینب موحدی [۲]، استادیار

۱- دانشکده مهندسی کامپیوتر - دانشگاه علم و صنعت ایران - تهران - ایران – hodastar_22@yahoo.com

۲- دانشکده مهندسی کامپیوتر - دانشگاه علم و صنعت ایران - تهران - ایران – zmovahedi@iust.ac.ir

چکیده: در سال‌های اخیر، شبکه‌های مبتنی بر نرم‌افزار به‌منظور مدیریت ساده‌تر، بهینه‌تر و قابلیت برنامه‌ریزی بیش‌تر شبکه‌های کامپیوتری مطرح شده‌اند. این شبکه‌ها از جداسازی لایه کنترل از داده و متمرکز سازی بخش کنترلی بهره می‌برند. با توجه به رشد سریع شبکه‌ها، افزایش تعداد سوئیچ‌ها و ترافیک موجود در شبکه، معماری‌های توزیع‌شده برای بخش کنترلی با حفظ دید شبکه‌ای متمرکز به‌منظور بهبود دسترسی‌پذیری، تحمل‌پذیری خطا و قابلیت اطمینان مطرح شدند. در این نوع از معماری‌ها، چگونگی تخصیص سوئیچ‌ها به کنترل‌کننده‌ها به‌منظور ایجاد توازن بار بین کنترل‌کننده‌ها و درنتیجه استفاده بهینه‌تر از منابع شبکه از اهمیت بالایی برخوردار است. برای پاسخگویی به این چالش‌ها، در این مقاله به ارائه روشی توزیعی و پایدار برای توازن بار بین کنترل‌کننده‌ها می‌پردازیم. روش پیشنهادی اطلاعات مربوط به میزان بار هر کنترل‌کننده را جمع‌آوری نموده و در صورت تجاوز از حد آستانه و نرخ توازن بار، سوئیچی را جهت مهاجرت به کنترل‌کننده با کم‌ترین میزان بار انتخاب می‌نماید که منجر به بهبود توازن بار شبکه گردد و کارایی حاصل از مهاجرت نسبت به هزینه تحمیلی به شبکه بهتر باشد. روش پیشنهادی با وجود عملکرد توزیعی از مهاجرت هم‌زمان بار دو کنترل‌کننده و ناپایداری حاصل از انتقال بار به یک کنترل‌کننده مقصد یکسان و نیاز به توزیع بار مجدد جلوگیری می‌کند. نتایج ارزیابی‌های حاصل از پیاده‌سازی روش پیشنهادی نشان می‌دهد که این روش سبب کاهش تا حدود ۷۰ درصد در زمان پردازش بسته‌های Packet-In در کنترل‌کننده دچار ازدحام، بهبود حدود ۱۵ درصد در میانگین مصرف حافظه و افزایش حدود ۵۰ درصد در توان گذردهی ترافیک کنترل‌کننده گشته است.

واژه‌های کلیدی: شبکه‌های مبتنی بر نرم‌افزار، پروتکل OpenFlow، کنترل‌کننده‌های توزیع‌شده، توازن بار.

Stable Distributed Load Balancing between Controllers in Software Defined Networks

B. Ahmadi, MSc student[1]; Z. Movahedi, Assistant Professor[2]

1- Faculty of Computer Engineering, Iran University of Science & Technology, Tehran, Iran, Email: hodastar_22@yahoo.com
2- Faculty of Computer Engineering, , Iran University of Science & Technology, Tehran, Iran, Email: zmovahedi@iust.ac.ir

Abstract: In recent years, Software Defined Networks (SDN) have been raised as a promising approach to improve the network programmability and management of computer networks. It consists in separating the control plane from the data plane and centralizing the control part of the network. Due to the rapid growth of computer networks in terms of number of switches and the amount of transiting traffic, the distributed architecture with centralized view on the network has been designed for control plane, enhancing the scalability, availability, fault tolerance and reliability. In such a distributed architecture, the load balancing between controllers plays an important role towards the optimal use of networking resources. To address the aforementioned challenges, we propose a stable distributed solution for load balancing between controllers in software defined networks. The proposed solution collects information on the amount of load of controllers and their corresponding switches. Based on this knowledge, the controller with the highest overload migrates the switch leading to the best enhancement in load balancing of the network to the least-loaded controller, if the network load is not balanced and the migration benefit is significant compared to its cost. The proposed solution inhibits simultaneous migrations triggered by distributed controllers to avoid cascade re-migrations and ensures the network stability. The results of the test-bed study of the proposed approach show that out solution outperforms other counterparts up to 15% in terms of average memory consumption, 50% in terms of controller traffic throughput and 70% in terms of processing time of the overloaded controllers.

Keywords: ***Software-Defined Networks (SDN)**, **OpenFlow, Distributed controllers, Load Balancing.***

تاریخ ارسال مقاله: ۱۳۹۶/۰۳/۰۹

تاریخ اصلاح مقاله: ۱۳۹۶/۰۸/۱۶

تاریخ پذیرش مقاله: ۱۳۹۷/۰۱/۰۱

نام نویسنده مسئول: زینب موحدی

نشانی نویسنده مسئول: ایران - تهران – میدان رسالت – خیابان هنگام – دانشگاه علم و صنعت- دانشکده مهندسی کامپیوتر.

۱ -مقدمه

امروزه شبکه‌های مبتنی بر نرم‌افزار[1] به‌منظور قابلیت برنامه‌ریزی بهتر، مدیریت آسان‌تر و نوآوری سریع‌تر شبکه‌های کامپیوتری مطرح شده‌اند [۱]. در این معماری برخلاف شبکه‌های سنتی، بخش کنترلی[2] از بخش داده[3] جدا شده و به یک کنترل‌کننده مرکزی به‌نام کنترل‌کننده SDN منتقل شده است. این کنترل‌کننده با ایجاد یک دید متمرکز جهانی نسبت به تمامی اجزای شبکه، امکان مدیریت و برنامه‌ریزی از راه دور شبکه را بدون ارتباط مستقیم با زیرساخت فراهم نموده است [۲]. با توجه به مزایای شبکه‌های مبتنی بر نرم‌افزار، امروزه این شبکه‌ها در موارد متعددی ازجمله مدیریت مراکز داده، شبکه‌های بی‌سیم، شبکه‌های ابری[4]، بخش‌بندی شبکه[5] و مجازی سازی[6] مورد استفاده قرار می‌گیرند.

با توجه به این‌که معماری‌های متمرکز شبکه‌های مبتنی بر نرم‌افزار از برخی جنبه‌ها از جمله مقیاس‌پذیری، امنیت، کیفیت سرویس و تبدیل کنترل‌کننده مرکزی به تنها نقطه شکست[7] با محدودیت‌هایی روبرو هستند، معماری‌های توزیع‌شده با حفظ دید منطقی متمرکز برای بخش کنترلی این شبکه‌ها مطرح شده‌اند. در معماری توزیع‌شده، شبکه به چند دامنه تقسیم می‌شود که هر دامنه متشکل از زیرساخت فیزیکی مانند سوئیچ‌های OpenFlow و اتصال‌های حامل ترافیک بخش کنترلی تحت مدیریت یک کنترل‌کننده می‌باشد. یک سوئیچ می‌تواند به‌طور هم‌زمان به چندین کنترل‌کننده متصل شود، اما فقط یک کنترل‌کننده به‌عنوان master و بقیه slave هستند.

در معماری توزیع‌شده بخش کنترلی، توزیع و توازن بار بین کنترل‌کننده‌ها به‌دلیل جلوگیری از ازدحام در برخی کنترل‌کننده‌ها، استفاده بهینه‌تر از منابع، حفظ پایداری شبکه، بهبود کیفیت سرویس و قابلیت مقیاس‌پذیری از اهمیت بالایی برخورداراست [۳،۴]. این روش‌ها با جلوگیری از بیکارماندن بعضی از کنترل‌کننده‌ها، از انتظار طولانی مدت درخواست‌های مربوط به سایر کنترل‌کننده‌ها به‌علت بار کاری[8] بالای آن‌ها جلوگیری می‌کنند. بدین ترتیب، با انتقال بار مدیریتی سوئیچ‌های عامل سربار در یک کنترل‌کننده سنگین (مشغول) به سایر کنترل‌کننده‌ها می‌توان بدون نیاز به نصب کنترل‌کننده جدید، ضمن تضمین کیفیت خدمات به استفاده بهتری از منابع دست یافت.

با وجود این که تاکنون روش‌هایی جهت توازن بار بین کنترل‌کننده‌ها مطرح شده‌اند [۸،۷،۶،۵]، اما این روش‌ها از مشکلاتی از جمله وجود مؤلفه کنترلی متمرکز در برخی معماری‌ها، پیچیدگی دستیابی به دید مطلوب جهانی توسط هر کنترل‌کننده، نگاشت ایستا بین کنترل‌کننده و سوئیچ و در نتیجه توزیع بار نامناسب در شرایط پویای شبکه، ایجاد ازدحام در کنترل‌کننده مقصد، وقوع مهاجرت‌های متوالی و ناپایداری ناشی از آن با انتخاب نامناسب سوئیچ مهاجر یا وقوع مهاجرت‌های همزمان، و در نظر نگرفتن هزینه مهاجرت بار رنج می‌برند [۹،۱۰].

برای پاسخگویی به این چالش‌ها در این مقاله، به ارائه راه‌حلی کارآمد، توزیعی و پایدار برای توازن بار پویا بین کنترل‌کننده‌ها در شبکه‌های SDN می‌پردازیم. راه حل پیشنهادی به جمع‌آوری اطلاعات مربوط به میزان بار هر کنترل‌کننده و مقایسه آن با حد آستانه بار[9](LT) و نرخ توازن بار[10] شبکه می‌پردازد. در صورت گذر بار یک کنترل‌کننده از این دو حد مجاز و بالاتر بودن میزان آن نسبت به سایر کنترل‌کننده‌ها، سوئیچی که مهاجرت آن منجر به بهبود توازن بار در شبکه می‌گردد، جهت مهاجرت به کنترل‌کننده‌ای با کم‌ترین میزان بار انتخاب می‌شود. در پایان، در صورتی عملیات مهاجرت انجام می‌گردد که کارایی حاصل از آن نسبت به هزینه تحمیلی به شبکه بهتر باشد. با توجه به امکان مهاجرت بار تنها یک کنترل‌کننده در هر زمان و انتخاب سوئیچ مهاجر با لحاظ بهبود در توازن بار، از وقوع سربار در کنترل‌کننده مقصد و ناپایداری و مهاجرت‌های متوالی ناشی از آن جلوگیری می‌شود. راه‌حل پیشنهادی با n کنترل‌کننده و m سوئیچ دارای پیچیدگی زمانی $o(n^2m)$ می‌باشد که می‌تواند منجر به بهبود توازن بار در شبکه، اجرای عملیات مهاجرت کم‌تر و بهینه‌تر با حفظ تعادل بین کارایی حاصل و هزینه مربوطه و هم‌چنین کاهش زمان پاسخ و حافظه مصرفی گردد.

ادامه مقاله به شرح زیر سازمان‌دهی شده است. در بخش دوم به بررسی کارهای پیشین در زمینه توازن بار بین کنترل‌کننده‌های شبکه‌های مبتنی بر نرم‌افزار می‌پردازیم. معماری و طریقه عملکرد روش پیشنهادی در بخش سوم بیان می‌گردد. در بخش چهارم، به ارزیابی نتایج حاصل از آزمایش‌ها و مقایسه با روش‌های پیشین پرداخته می‌شود. در بخش پنجم، به بحث و بررسی در مورد کارایی روش پیشنهادی و تحلیل پیچیدگی الگوریتم پرداخته می‌شود. در بخش ششم، نتیجه‌گیری و پیشنهادهایی برای کارهای آینده ارائه خواهد شد.

۲ -کارهای پیشین

هدف از روش [۷] ارائه پروتکلی به‌منظور اجتناب از نصب یک کنترل‌کننده جدید درزمان افزایش موقت بارکاری می‌باشد، به‌گونه‌ای که همه کنترل‌کننده‌ها در زمان پاسخ هدف[11] تعیین شده پاسخگو باشند. به‌این منظور، کنترل‌کننده‌ها بر اساس میزان اختلاف میانگین زمان پاسخشان با زمان پاسخ هدف در گروه‌های سبز، زرد و قرمز دسته‌بندی می‌شوند. هر کنترل‌کننده مکانیزم جمع‌آوری اطلاعات سایر کنترل‌کننده‌های شبکه را در صورت قرارگیری در دسته زرد به‌منظور انجام مهاجرت آغاز می‌کند. بر اساس این اطلاعات، کنترل‌کننده‌های موجود در محدوده چندگامی سوئیچ عامل سربار به‌عنوان کنترل‌کننده‌های محلی شناسایی شده و از میان آن‌ها نزدیک‌ترین کنترل‌کننده گروه سبز به سوئیچ عامل سربار که انتخاب آن منجر به بازگشت زمان پاسخ به دسته سبز می‌گردد، به‌عنوان کنترل‌کننده مقصد انتخاب می‌شود. با توجه به انتخاب کنترل‌کننده

مقصد به‌صورت محلی نسبت به سوئیچ عامل سربار، ممکن است این روش قادر به انجام مهاجرت نباشد. به‌علاوه، با توجه به عدم در نظر گرفتن شرایط آتی میزان بار کنترل‌کننده مقصد، ممکن است این روش با توجه به امکان قرارگیری این کنترل‌کننده در محدوده زرد، منجر به مهاجرت‌های متوالی گردد. هم‌چنین، با توجه به تصمیم مستقل هر یک از کنترل‌کننده‌ها جهت انتقال بار، امکان انتخاب کنترل‌کننده مقصد مشترک توسط چندین کنترل‌کننده دچار سربار وجود دارد که خود ممکن است منجر به انجام مهاجرت‌های متوالی و اتلاف منابع شبکه گردد.

روش Balance Flow [۸] با استفاده از یک روش اکتشافی به‌صورت پویا به توزیع ترافیک بین کنترل‌کننده‌های مختلف می‌پردازد و از تأخیر نامناسب انتشار جلوگیری می‌کند. در این روش، هر یک از کنترل‌کننده‌ها ماتریس درخواست‌های جریان بین سوئیچ‌های متصل به خود را نگه‌داری می‌کند. به‌علاوه، یکی از کنترل‌کننده‌ها به‌عنوان ابر کنترل‌کننده[۱۲] انتخاب می‌شود و سایر کنترل‌کننده‌ها از طریق ارتباطات بین کنترل‌کننده‌ها به‌طور دوره‌ای اطلاعات ماتریس جریان خود را به ابر کنترل‌کننده ارسال می‌نمایند. به‌منظور هماهنگی بین کنترل‌کننده‌ها، ابرکنترل کننده به‌محض تغییر در شرایط ترافیک به‌طور مجدد تنظیمات جریان متفاوتی را به هر کنترل‌کننده تخصیص می‌دهد، به‌طوری‌که هیچ کنترل‌کننده‌ای در زمان اجرا نامتوازن نباشد. به این منظور، میانگین تعداد درخواست‌های جریان تحت کنترل هر کنترل‌کننده و تعداد کل درخواست‌ها در شبکه محاسبه می‌شود و به‌محض شناسایی عدم توازن بار، ابر کنترل‌کننده الگوریتم تقسیم درخواست‌ها را اجرا می‌کند و تنظیمات جریان متفاوتی به کنترل‌کننده‌ها تخصیص می‌دهد. از معایب این روش می‌توان به نیاز به نظارت دائمی شرایط شبکه توسط ابر کنترل‌کننده برای تشخیص عدم توازن، سربار محاسبات و تنظیم جریان‌های متفاوت برای هر کنترل‌کننده و گلوگاه بودن ابر کنترل‌کننده اشاره کرد.

در روش[۱۱] یک طرح توزیع بار به‌ازای هر جریان[۱۳] بین کنترل‌کننده‌های متعدد پیشنهاد شده است. زمانی که ظرفیت یک کنترل‌کننده به مقدار آستانه می‌رسد، کنترل‌کننده آماده انتقال پیام‌های ورودی به سایر کنترل‌کننده‌ها برای جلوگیری از مسدود شدن پیام‌ها می‌گردد. در این روش، برای پیام‌های جدید و مهاجر، از نرخ ورود با توزیع پواسون و برای زمان پیام‌ها از توزیع نمایی استفاده می‌گردد. هم‌چنین، از مدل زنجیره مارکوف و احتمال مسدود شدن پیام‌ها به‌منظور مدل‌سازی مهاجرت‌های بین کنترل‌کننده‌ها استفاده می‌شود. نتایج حاصل از این روش بیانگر این است که احتمال مسدود شدن پیام‌ها در روش‌های قبلی بیشتر می‌باشد، زیرا زمانی که کنترل‌کننده پیش فرض ظرفیت قابل قبولی برای پذیرش پیام‌ها نداشته باشد، پیام‌های ورودی همواره باید مسدود شوند. اما این روش، با توجه به تعداد کنترل‌کننده‌ها و احتمال p_f (احتمال عدم پذیرش یک پیام توسط کنترل‌کننده پیش فرض) احتمال مسدود شدن پیام کم‌تری دارد، هم‌چنین احتمال این‌که حداقل یک کنترل‌کننده بتواند پیام مهاجر را پردازش کند افزایش می‌یابد.

روش ElastiCon [۱۲]، از توازن بار برای حفظ میزان بار کنترل‌کننده‌ها در محدوده مناسب از طریق نگاشت پویای بین سوئیچ و کنترل‌کننده استفاده می‌نماید. به‌این منظور، مهاجرتی که بیش‌ترین کاهش در انحراف استاندارد از میانگین بهره‌وری در کنترل‌کننده مبدأ را حاصل نماید به‌عنوان مهاجرت مناسب انتخاب می‌گردد. به‌علاوه، در صورتی که هیچ مهاجرت مناسبی برای کاهش بار کنترل‌کننده دچار ازدحام وجود نداشته باشد، از روشن کردن کنترل‌کننده جدید استفاده می‌نماید. هم‌چنین، این روش از خاموش کردن کنترل‌کننده‌ها برای صرفه‌جویی انرژی در شرایط پایین بودن میزان بهره‌وری کنترل‌کننده‌ها بهره می‌برد. با وجود مزایای روش ElastiCon در صرفه‌جویی انرژی در کنار کاهش زمان پردازش درخواست‌ها در کنترل‌کننده‌ها، این روش از پیچیدگی زمانی بالا جهت انتخاب مهاجرت مناسب رنج می‌برد. در ضمن، انتخاب کنترل‌کننده مقصد بدون در نظر گرفتن فاصله زمانی میان سوئیچ و کنترل‌کننده صورت می‌پذیرد که این امر می‌تواند منجر به افزایش زمان پاسخ‌گویی به درخواست‌های سوئیچ مهاجر گردد.

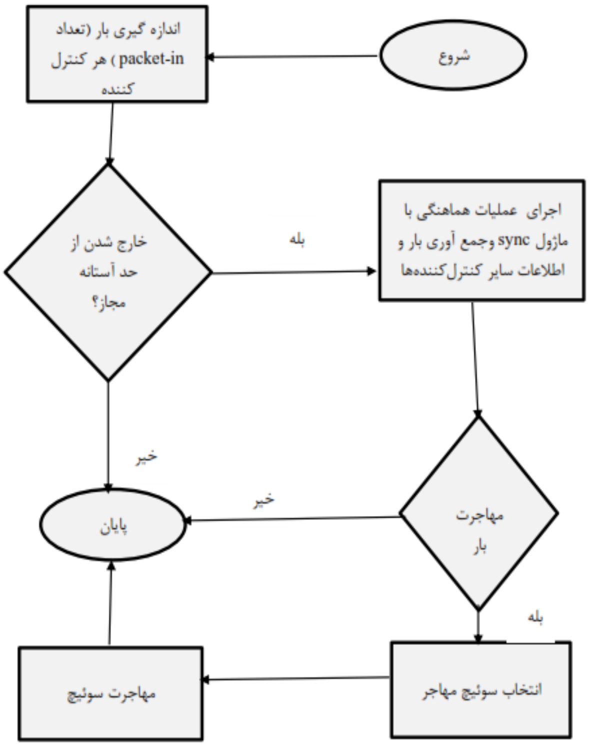
روش DALB[۱۴] [۱۳] یک معماری توزیع‌شده منطبق با شرایط و قابل انعطاف با نگاشت پویای بین سوئیچ و کنترل‌کننده ارائه می‌دهد. این روش یک مقدار آستانه بار برای هر کنترل‌کننده در نظر گرفته و در صورت تجاوز بار از این مقدار، سوئیچ مولد بیش‌ترین تعداد بسته‌های ارسالی را جهت مهاجرت به کنترل‌کننده با کم‌ترین بار در شبکه انتخاب می‌کند. در نهایت، عملیات مهاجرت در صورتی انجام می‌پذیرد که بار سوئیچ انتخابی از نصف تفاضل بار دو کنترل‌کننده کم‌تر باشد. چالش اصلی این روش انتخاب سوئیچ با بیش‌ترین بار جهت مهاجرت می‌باشد که با توجه به شرط انجام مهاجرت، ممکن است در نهایت این امر صورت نپذیرد. این در حالی است که می‌توانست با در نظر گرفتن معیار مناسب‌تری برای انتخاب سوئیچ، مهاجرت را انجام داده و منجر به بهبود توازن بار بین کنترل‌کننده‌ها گردد. به‌علاوه، این روش همانند روش‌های پیشین از مشکل ناپایداری و امکان ایجاد سربار در کنترل‌کننده مقصد در صورت انتخاب توسط چند کنترل‌کننده و نیاز به مهاجرت‌های متوالی رنج می‌برد.

۳ -الگوریتم پیشنهادی

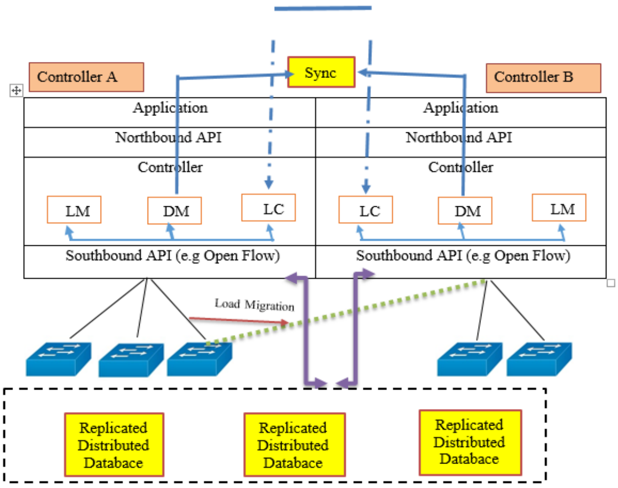
به‌منظور پاسخگویی به مشکلات روش‌های پیشین، در این بخش به ارائه روشی کارآمد، توزیعی و پایدار برای توازن بار پویا بین کنترل‌کننده‌های SDN می‌پردازیم. همان‌طور که فلوچارت روش پیشنهادی در شکل ۱ نشان می‌دهد، در این روش هر کنترل‌کننده میزان بار خود را به‌صورت دوره‌ای محاسبه نموده و در صورت تجاوز از حد آستانه بار، اطلاعات بار سایر کنترل‌کننده‌ها را به‌منظور محاسبه

نرخ توازن بار شبکه از پایگاه داده توزیع‌شده فیزیکی با دید منطقی جهانی جمع‌آوری می‌نماید. به‌منظور تضمین ایجاد بهبود در توازن بار شبکه پس از مهاجرت و جلوگیری از ناپایداری ناشی از مهاجرت هم‌زمان دو کنترل‌کننده، کنترل‌کننده تنها در صورتی اقدام به انجام عملیات توازن بار می‌نماید که میزان بار آن از نرخ توازن بار کم‌تر و از میزان بار سایر کنترل‌کننده‌ها بیش‌تر باشد. سپس سوئیچی که مهاجرت آن منجر به بهبود توازن بار شبکه می‌شود، جهت مهاجرت به کنترل‌کننده‌ای با کم‌ترین میزان بار انتخاب می‌شود. با توجه به این‌که روش پیشنهادی به‌صورت کاملاً توزیع‌شده و به‌عنوان بخشی از کنترل‌کننده SDN اجرا می‌شود، از مشکلات ناشی از تنها نقطه شکست جلوگیری کرده و سبب بهبود مقیاس‌پذیری، دسترس‌پذیری و قابلیت تحمل‌پذیری خطا می‌گردد. به‌علاوه، با توجه به این که روش پیشنهادی به‌صورت مبتنی بر رویداد به جمع‌آوری اطلاعات کنترل‌کننده‌ها می‌پردازد، نسبت به روش‌های موجود با سربار نظارت کم‌تری عملیات هماهنگ‌سازی صورت می‌پذیرد. هم‌چنین، از آنجائی که عملیات مهاجرت و انتخاب سوئیچ مهاجر تنها در صورت تأثیر مثبت در توازن بار شبکه صورت می‌پذیرد، از مهاجرت‌های نامناسب و ناپایداری حاصل از آن جلوگیری می‌گردد.

شکل ۲ معماری روش توازن بار پیشنهادی را نشان می‌دهد. همان‌طور که در شکل دیده می‌شود، مؤلفه‌های اصلی این معماری شامل واحد اندازه‌گیری بار (LM)[15]، واحد جمع‌آوری کننده بار (LC)[16]، واحد تصمیم گیرنده (DM)[17] و پایگاه داده تکرار شده (RBD)[18] می‌باشد که در ادامه هر یک به‌طور مفصل توضیح داده شده است.



شکل ۱: فلوچارت روش پیشنهادی



شکل ۲: معماری توازن بار پیشنهادی

۳-۱- واحد اندازه‌گیری بار

واحد اندازه‌گیری بار در کنترل‌کننده، میزان بار تحمیلی از هر سوئیچ به آن کنترل‌کننده و میزان کل بار کنونی کنترل‌کننده را به‌صورت محلی و به‌طور دوره‌ای اندازه‌گیری می‌کند. میزان بار تحمیلی از هر سوئیچ به کنترل‌کننده بر اساس نرخ ورود پیام‌های وارد شده از آن سوئیچ به کنترل‌کننده اندازه‌گیری می‌شود. بارکنونی کنترل‌کننده بر اساس مجموع تعداد پیام‌هایی که در واحد زمان از سوئیچ‌ها به آن کنترل‌کننده ارسال می‌شود ارزیابی می‌گردد. هرچند بر اساس پروتکل OpenFlow، یک کنترل‌کننده انواع متفاوتی درخواست از جمله پیام‌های Packet-In، Hello و Echo را از سوئیچ‌ها دریافت می‌کند، اما با توجه به اینکه نرخ پیام‌های Packet-In به‌صورت قابل توجهی نسبت به سایر پیام‌ها بالاتر است، در این مقاله میانگین نرخ ورود پیام‌های Packet-In را به‌عنوان بار یک کنترل‌کننده در نظر می‌گیریم.

۳-۲- واحد جمع‌آوری کننده بار

این واحد مسئول جمع‌آوری اطلاعات بار سایر کنترل‌کننده‌ها به‌منظور بهره‌برداری جهت تصمیم‌گیری درباره توزیع بار می‌باشد. با توجه به سربار ارسال یا دریافت پیام و تأثیر آن بر عملکرد کل سیستم، واحد جمع‌آوری بار پیشنهادی به‌صورت مبتنی بر رخداد[19] عمل می‌کند. به این منظور، صرفاً در صورتی که بار یک کنترل‌کننده از حد آستانه بار بیش‌تر شود، پیام هماهنگی[20] و درخواست اطلاعات به سایر کنترل‌کننده‌ها ارسال می‌شود.

اطلاعات جمع‌آوری شده توسط این واحد شامل بار هر کنترل‌کننده، سوئیچ‌های متصل به آن‌ها، میزان بار هر سوئیچ و سایر معیارهای مرتبط می‌باشد. در روش پیشنهادی از ماژول sync موجود در بسته Floodlight، به‌منظور به اشتراک‌گذاری اطلاعات کنترل‌کننده‌ها در یک پایگاه داده توزیع‌شده که دارای دید منطقی جهانی از شبکه می‌باشد، استفاده می‌گردد.

۳-۳- پایگاه داده توزیع‌شده

این پایگاه داده، پارامترهای مورد استفاده در تصمیم‌گیری و اطلاعات کنترل‌کننده‌های شبکه اعم از اطلاعات لینک، اطلاعات بار هر کنترل‌کننده و بار تحمیلی هر سوئیچ و توپولوژی را ذخیره می‌کند. مطابق شکل ۲ این پایگاه داده متشکل از مجموعه‌ای از پایگاه داده‌های توزیع‌شده فیزیکی با دید منطقی جهانی می‌باشد که به‌منظور نمایش خوشه‌ای از کنترل‌کننده‌ها به‌صورت یک کنترل‌کننده مرکزی از دید برنامه‌های کاربردی لایه بالا به کار می‌رود.

۳-۴- واحد تصمیم‌گیرنده

این واحد مسئول تصمیم‌گیری در خصوص لزوم انجام مهاجرت و انتخاب سوئیچ مناسب به این منظور می‌باشد. روند عملکرد این واحد در الگوریتم ۱ نشان داده شده است. در گام اول، هر کنترل‌کننده میزان بار خود را که توسط واحد اندازه‌گیری بار اندازه‌گیری شده و در پایگاه داده توزیعی ذخیره شده است با حد آستانه بار مقایسه می‌کند. در صورتی که این میزان از حد آستانه بار فراتر رفته باشد، این کنترل‌کننده دچار سربار شده است و جهت بررسی شرایط جهانی انجام توازن بار، پیامی برای جمع‌آوری اطلاعات به سایر کنترل‌کننده‌ها ارسال می‌کند. این عملیات منجر به به‌روزرسانی و هماهنگ‌سازی پایگاه داده توزیع‌شده موجود در کنترل‌کننده می‌گردد. اطلاعات جهانی جمع‌آوری شده در پایگاه داده توزیع‌شده به‌منظور اجتناب از مهاجرت هم‌زمان بار دو کنترل‌کننده دچار سربار و یا جلوگیری از مهاجرت بار در شرایط مناسب بودن توازن بار نسبی شبکه مورد استفاده قرار می‌گیرد. مهاجرت هم‌زمان بار دو کنترل‌کننده ممکن است منجر به انتقال سوئیچ مهاجر در هر دو کنترل‌کننده به کنترل‌کننده یکسان و در نتیجه ایجاد سربار در کنترل‌کننده مقصد شود. بر این اساس، یک کنترل‌کننده در صورتی تصمیم به انجام عملیات توازن بار خواهد گرفت که دارای بیش‌ترین بار بین همه کنترل‌کننده‌ها بوده و نرخ توازن بار کنونی شبکه از حد آستانه توازن بار کمتر باشد. نرخ توازن بار بر اساس نسبت میانگین بار به حداکثر بار شبکه از فرمول زیر محاسبه می‌شود:

$$Balance\_rate = \left(\frac{1}{n}\right)\sum_{i=1}^{n}\{L_i\} \,/\, max_{i=1}^{n}\{L_i\} \qquad (۱)$$

که در آن L_i نشان‌دهنده بار کنترل‌کننده i است. مقدار آستانه نرخ توازن بار با توجه به وضعیت شبکه و بررسی شرایطی که پایداری شبکه حفظ گردد تعیین می‌گردد.

در صورتی که بنا به شرایط یاد شده انجام توازن بار لازم باشد، فرآیند انتخاب کنترل‌کننده مقصد و سوئیچ مناسب جهت مهاجرت آغاز می‌گردد. در روش پیشنهادی، کنترل‌کننده‌ای با کم‌ترین میزان بار به‌عنوان کنترل‌کننده مقصد انتخاب شده و بار آن تحت عنوان L_{Target} در پایگاه داده ذخیره می‌گردد. برخلاف روش‌های پیشین که در آن‌ها سوئیچی با بیش‌ترین میزان بار برای مهاجرت انتخاب می‌گردید، در روش پیشنهادی سوئیچی که مهاجرت آن منجر به بهترین وضعیت توازن بار در شبکه گردد به‌عنوان سوئیچ مهاجر انتخاب می‌شود. به این منظور، ابتدا میانگین بار کنترل‌کننده‌ها بر اساس رابطه (۲) محاسبه می‌شود:

$$\text{Average\_load} = \sum_{i=1}^{n} L_i/\text{n} \qquad (۲)$$

در مرحله بعد، انحراف از میانگین بار کنترل‌کننده مقصد به‌صورت اختلاف بین بار کنترل‌کننده مقصد و میانگین بار کنترل‌کننده‌ها طبق رابطه (۳) محاسبه می‌گردد:

$$\text{Mean\_}Dev_{Target} = |\text{Average\_load} - L_{Target}| \qquad (۳)$$

در نهایت، مطابق رابطه (۴) سوئیچی از کنترل‌کننده مبدأ به‌منظور مهاجرت انتخاب می‌گردد که کم‌ترین میزان اختلاف را با $Mean\_Dev_{Target}$ داشته و منجر به بهترین میزان بهبود در توازن بار گردد:

$$\text{Migrated\_Switch} = min_{j1}^{m}\,(|l_{s_j} - \text{Mean\_}Dev_{Target}|) \qquad (۴)$$

در این رابطه، l_{s_j} میزان بار هر سوئیچ s_j و m تعداد کل سوئیچ‌هایی می‌باشد که کنترل‌کننده مبدأ برای آن‌ها master است. بنابراین، میزان بار برای انتقال از کنترل‌کننده مبدأ به کنترل‌کننده مقصد برابر با بار سوئیچ مهاجر خواهد بود.

پس از انتخاب سوئیچ مهاجر، عملیات مهاجرت سوئیچ منتخب به کنترل‌کننده مقصد در صورتی فعال خواهد شد که میزان بهبود حاصل شده در توازن بار در صورت انجام مهاجرت نسبت به میزان هزینه انجام آن قابل قبول باشد. به این منظور، در صورتی عملیات مهاجرت صورت می‌پذیرد که انجام آن (۱) منجر به کاهش انحراف بار هر دو کنترل‌کننده مبدأ و مقصد از میانگین بار شبکه نسبت به قبل از مهاجرت گردد، و (۲) میزان کاهش انحراف از میانگین بار هر دو کنترل‌کننده پس از مهاجرت از حد آستانه مهاجرت (T_M) بیش‌تر باشد. به این ترتیب، می‌توان بین هزینه مهاجرت و میزان کاهش در انحراف میانگین بعد از مهاجرت تعادل برقرار نمود. به‌عبارت دیگر، انجام مهاجرت در صورت بهبود کم‌تر از این مقدار در میزان توازن بار حاصل شده از نظر هزینه‌های مهاجرت مقرون به صرفه نخواهد بود. منظور از هزینه مهاجرت، هزینه میزان حافظه مصرفی و بار شبکه‌ای به‌منظور هماهنگ‌سازی بین کنترل‌کننده‌ها، هزینه پردازشی برای تصمیم‌گیری (مانند CPU) ، هزینه شبکه‌ای برای ارتباطات سوئیچ به کنترل‌کننده جدید، هزینه جابجایی بسته‌ها و سربار حاصل از این جابجایی می‌باشد. جزئیات این فرآیند در الگوریتم ۲ نمایش داده شده است.

در نهایت، در صورتی که انجام مهاجرت به صرفه باشد، کنترل‌کننده مبدأ با ارسال پیام تغییر نقش از master به slave، برای سوئیچ مهاجر، slave شده و این سوئیچ به کمک پروتکل مهاجرت سوئیچ به کنترل‌کننده مقصد منتقل می‌گردد. کنترل‌کننده مقصد، پس از کسب اطلاعات محلی خود از طریق پایگاه داده توزیع‌شده با

دید منطقی جهانی و سرویس‌های sync، از سوئیچ مهاجر مطلع می‌گردد و با ارسال درخواست تغییر نقش از slave به master، مسئول دریافت و پردازش پیام‌های این سوئیچ می‌شود. در طول فرآیند انجام مهاجرت، از اتصالات TCP با دسترس‌پذیری بالا (HA - TCP)[۲۱] به‌منظور تبادل پیام هماهنگی بین کنترل‌کننده‌ها ضمن تضمین دسترسی‌پذیری بالا و اطمینان از عدم از دست رفتن پیام‌های سوئیچ در حال مهاجرت استفاده می‌گردد.

در پایان فرایند، عملیات تنظیم خودکار و منطبق با شرایط برخی از پارامترهای تصمیم‌گیری از جمله حد آستانه بار انجام می‌شود. به این منظور، مشابه روش ارائه شده در مرجع [۱۳] چنانچه میانگین بار بیش‌تر از حد آستانه بار فعلی باشد، مقدار میانگین به‌عنوان حد آستانه جدید جایگزین می‌شود.

Input:
{L_1,…,L_n} = List of all controllers' load
{L_{S1}, ... L_{sn}} = List of all switches' load
$LT1$: Load measurement threshold
Balance_rate = Load balancing rate
Output:
0: no need for SDN controller load balancing
1: SDN controller load balancing success
-1: SDN controller load balancing fail
1: L_C= Load measurement()
2: If $L_C < LT1$:
3: Return 0
4: Else
5: {L_1,…,L_n} = Controllers_Load_Collect ()
6: L_{max} = max ({L1,…,Ln})
7: $L_{target} = L_{min}$ = min ({L1,…,Ln})
8: Cid_{target} = id of the target controller
9: Average_load = $\frac{\sum_{1=1}^{n} Li}{n}$
10: Balance_rate = $\frac{\frac{1}{n}\sum_{i=1}^{n} Li}{L_{max}}$
11: If Balance_rate> 0.8 or $L_C < L_{max}$
12: Return 0
13: Else
14: $Mean\_Dev_{Target}$ = |Averag_load - L_{target} |
15: {L_{S1}, ... L_{sn}} = Switches_Load_Collect()
16: For each switch i of this controller
DS_i=| $Mean\_Dev_{Target}$- L_{si}|
17: DS_{min} = min ({DS_1,…,DS_n})
18: Dpid = id of switch with DS_{min}
19: If (Successful_SwitchMigration(Dpid))
20: SendRoleRequest(Dpid, slave, Cid_{target})
21: If success
22: Return 1
23: Else
24: Throw Exception
25: Return -1
26: End if
27: End if
28: LT = AdaptiveLT({L1 , …., Ln})
29: End if

الگوریتم ۱: روش پیشنهادی

//before migration
Input:
BeforeMean_$Dev_{Overload}$ =| $L_{Overloaded} - average\_load$|
BeforeMean_Dev_{Target} = |L_{Target} - $average\_load$|
//assuming Doing migration for Migrated_Switch with $L_{migrated}$
$NewL_{OverLoaded}$ = | $L_{Overloaded}$ - $L_{migrated}$|
$NewL_{Target} = L_{Target} + L_{migrated}$
$AfterMean\_Dev_{OverLoaded}$ = |$NewL_{OverLoaded}$ - $average\_load$|
$AfterMean\_Dev_{Target}$ = |$NewL_{Tatget}$ - $average\_load$|
Output:
0: Migration is not efficient
1: Migratin success
1: If
($After$Mean_$Dev_{Overload}$ <
BeforeMean_$Dev_{Overload}$ && $AfterMean\_Dev_{Target}$ <
$Before$Mean_Dev_{Target})
2: If
(|$AfterMean\_Dev_{OverLoaded}$ −
$BeforeMean\_Dev_{OverLoaded}$| ≥
T_M && |$AfterMean_{Dev_{Target}}$ −
$Before$Mean_Dev_{Target}| ≥ T_M)
3: Do_Migration()
4: Return 1
5: Else
6: Return 0 // Migration is not efficient

الگوریتم ۲: انتخاب سوئیچ مناسب برای مهاجرت موفق

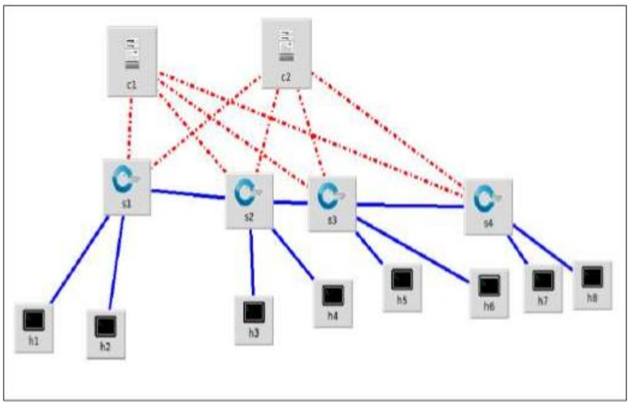
۴ -ارزیابی عملکرد روش پیشنهادی

به‌منظور بررسی و تحلیل عملکرد روش پیشنهادی، آن را پیاده‌سازی نموده و با روش DALB، که در قسمت کارهای پیشین شرح داده شده است، مقایسه نمودیم. علت انتخاب روش DALB جهت مقایسه، برتری آن نسبت به سایر روش‌های موجود از لحاظ ساختار توزیعی، پیچیدگی زمانی پایین، در نظر گرفتن نرخ توازن بار شبکه جهت تصمیم به انجام یا عدم انجام توازن بار و در نظر گرفتن آستانه بار قابل تنظیم بر اساس بار همه کنترل‌کننده‌ها می‌باشد. هر دو روش با استفاده از کنترل‌کننده floodlight [۱۳] نسخه floodlight master که یک کنترل‌کننده متن‌باز OpenFlow [۱۴] مبتنی بر جاوا و شاخه‌ای از کنترل‌کننده Beacon [۱۵] می‌باشد، پیاده‌سازی شده‌اند. این کنترل‌کننده توسط شبکه‌های سوئیچ بزرگ [۱۶] برای هماهنگی جریان‌های ترافیک در محیط SDN، با پروتکل OpenFlow توسعه یافته است. از OpenVSwitch 2.3.90 و Open JDK 1.8.0 بر روی سیستم عامل Ubuntu 14.04 با تخصیص ۸ گیگابایت رم (RAM) و ۲ هسته پردازنده در ماشین مجازی نصب شده بر روی VirtualBox استفاده شده است. علاوه براین، از نسخه mininet 2.2.1 [۱۷] به‌عنوان مقلد شبکه برای ایجاد توپولوژی‌های متفاوت، wireshark به‌عنوان ابزار نظارت بسته‌ها و شبکه و iperf به‌عنوان ابزار تولید ترافیک استفاده می‌گردد.

۴-۱ -شرح آزمایش‌ها

آزمون‌ها بر روی یک نمونه توپولوژی مطابق شکل ۳ انجام شده است. توپولوژی مورد استفاده متشکل از ۲ کنترل‌کننده، ۴ سوئیچ و ۸ پایانه می‌باشد. در این توپولوژی کنترل‌کننده C1 برای سوئیچ‌های S1، S2 و

S4 نقش master و برای سوئیچ S3 نقش slave را دارد. همچنین، کنترل‌کننده C2 دارای نقش master برای سوئیچ S3 و نقش slave برای سایر سوئیچ‌ها می‌باشد. بدیهی است با توجه به استفاده از پایگاه داده توزیع‌شده تکراری و وجود اطلاعات جهانی لازم در هر یک از کنترل‌کننده‌ها جهت انجام الگوریتم توازن بار، روش پیشنهادی در توپولوژی‌های مختلف با هر تعداد کنترل‌کننده قابل استفاده می‌باشد و انتخاب توپولوژی متشکل از دو کنترل‌کننده در این مقاله صرفاً به‌دلیل محدودیت‌های سخت‌افزاری با توجه به تست واقعی در محیط آزمایشگاهی صورت پذیرفته است.



شکل ۳: توپولوژی مورد استفاده جهت ارزیابی‌ها

در پیاده‌سازی‌های انجام شده، دوره زمانی مورد استفاده واحد اندازه‌گیری بار هر ۱ ثانیه یک‌بار می‌باشد. هم‌چنین، مقدار حد آستانه ۱۰۰۰ برای بار کنترل‌کننده‌ها و ۰٫۸ برای نرخ توازن بار در نظر گرفته‌شده است. به‌منظور مشاهده عملکرد روش پیشنهادی در مقایسه با روش DALB، آزمون‌های متعددی هر بار با میزان بار متفاوت از سوی سوئیچ‌ها به کنترل‌کننده‌ها طراحی کردیم، به‌طوری که شرایط مختلف انجام یا عدم انجام مهاجرت در یکی یا هر دو روش به دست آید. با توجه به این که همه این آزمون‌ها برتری روش پیشنهادی را ثابت می‌کرد، در ادامه صرفاً نتایج مربوط به آزمونی که مهاجرت در روش پیشنهادی انجام پذیرفته اما در روش DALB انجام نمی‌شود نمایش داده شده است. برای این آزمون، بار سوئیچ‌ها و کنترل‌کننده‌ها مطابق جدول ۱ توسط ابزار iperf تولید شده است.

جدول ۱: اطلاعات بار سوئیچ و کنترل‌کننده

سوئیچ/کنترل‌کننده	بار(تعداد packet_in)
S1	۵۰۵
S2	۳۳۵
S3	۲۷۵
S4	۱۶۰
C1	۱۰۰۰
C2	۲۷۵
Average - Load	۶۳۷٫۵

۴-۲- معیارهای ارزیابی

به‌منظور ارزیابی روش پیشنهادی، از معیارهای زیر استفاده نموده‌ایم:

۱ - *میانگین تعداد بسته‌های Packet-In دریافتی:* این معیار، میانگین تعداد بسته‌هایی که از طرف سوئیچ‌ها به کنترل‌کننده ارسال شده‌اند را محاسبه نموده و به‌منظور ارزیابی تأثیر اجرای الگوریتم توازن بار بر بار ورودی به کنترل‌کننده استفاده می‌شود.

۲ - *میانگین تعداد بسته‌های Packet-Out خروجی:* این معیار به‌صورت میانگین تعداد بسته‌هایی که توسط کنترل‌کننده به سمت سوئیچ‌ها ارسال می‌شوند محاسبه می‌شود. هدف از این معیار، بررسی توان پاسخ‌دهی کنترل‌کننده در طی زمان با توجه به نرخ ورودی پیام‌های Packet-In می‌باشد.

۳ - *میانگین زمان پردازش بسته‌های Packet-In دریافتی:* برای اندازه‌گیری میانگین زمان پردازش بسته‌های دریافتی، زمان پردازش هر بسته Packet-In را بر اساس اختلاف زمان دریافت بسته در کنترل‌کننده و اتمام پردازش آن محاسبه نموده و سپس مقدار میانگین این زمان برای همه بسته‌های دریافتی در طی دوره زمانی مربوط به آزمون موردنظر محاسبه می‌شود. به این منظور، زمان‌های موردنظر با قرار دادن timer به‌ازای دریافت و پایان پردازش هر بسته Packet-In در ماژول PktInProcessingTime در کنترل‌کننده floodlight که در مسیر src/main/java/net/floodlightcontroller/perfmon موجود می‌باشد استخراج شده و سپس برای کل بسته‌های دریافتی در بازه زمانی مربوط به آزمون میانگین محاسبه شده است.

۴ - *میانگین میزان حافظه مصرفی کنترل‌کننده:* هدف از این معیار، ارزیابی تأثیر توازن بار برکاهش میانگین میزان حافظه مصرفی کنترل‌کننده مبدأ می‌باشد. به این منظور، میانگین میزان حافظه مصرفی قبل و بعد از مهاجرت بر اساس میزان کل حافظه و میزان حافظه آزاد هر کنترل‌کننده محاسبه می‌گردد. در کنترل‌کننده floodlight، این مقادیر را می‌توان با استفاده از REST API به‌دست آورد یا از طریق web GUI کنترل‌کننده floodlight از بخش JVM total memory و JVM free memory مشاهده نمود. سپس، از طریق اختلاف این دو مقدار میزان حافظه مصرفی را در هر یک از زمان‌های قبل و بعد از مهاجرت اندازه‌گیری نمود.

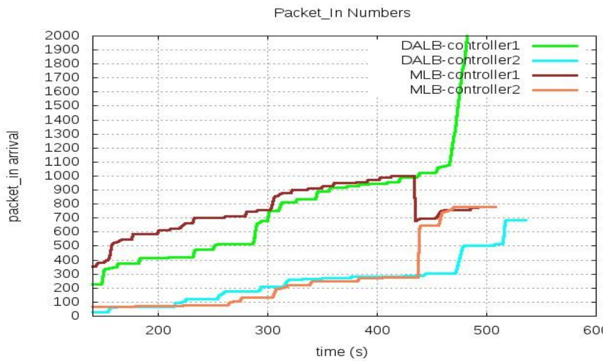
۴-۳- شرح نتایج

در ادامه، نتایج به‌دست آمده برای هر یک از معیارهای ارزیابی به‌طور جداگانه شرح داده شده است. در نمودارها روش پیشنهادی با عنوان MLB نمایش داده شده است.

۴-۳-۱- تأثیر بر میانگین بسته‌های Packet-In دریافتی

در شکل ۴، عملکرد روش پیشنهادی و روش DALB از لحاظ میانگین تعداد بسته‌های Packet-In ورودی مقایسه شده است. در طی آزمایش، در حدود لحظه ۴۴۰، بار کنترل‌کننده C1 از حد آستانه ۱۰۰۰ فراتر رفته است. هم‌چنین، نرخ توازن بار در این حالت برابر با ۰٫۶۴ می‌باشد.

الگوریتم DALB، در مرحله انتخاب سوئیچ به‌دلیل عدم احراز شرط کافی برای سوئیچ انتخاب شده، عملیات مهاجرت را انجام نداده و حتی با وجود اعمال بار زیاد به کنترل‌کننده اول و اختلاف بار بین دو کنترل‌کننده، هم‌چنان کنترل‌کننده اول برای سوئیچ S1 در حالت master می‌ماند. بنابراین، در نمودار شکل ۴، سربار کنترل‌کننده C1 در روش DALB و به‌عبارتی شیب نمودار مربوط به کنترل‌کننده C1 به‌طور صعودی افزایش یافته و اختلاف بار بین دو کنترل‌کننده به‌طور قابل ملاحظه‌ای بیشتر می‌شود. این در حالی است که در روش پیشنهادی، با توجه به برقراری شرایط توازن بار، سوئیچ S2 جهت مهاجرت انتخاب شده و با توجه به تضمین شرایط مهاجرت موفق، این سوئیچ از کنترل‌کننده C1 به C2 مهاجرت کرده و از این پس C2 برای آن master می‌باشد. در نتیجه، بار دو کنترل‌کننده پس از مهاجرت به هم نزدیک می‌شود. بنابراین، با انجام مهاجرت سوئیچ در روش پیشنهادی، سربار کنترل‌کننده C1 کاهش یافته و هم‌چنین با پذیرش بار سوئیچ S2 توسط کنترل‌کننده C2 به توازن بار مناسبی بین کنترل‌کننده‌ها دست می‌یابیم. همان‌طور که در شکل ۴ نشان داده شده است، نمودار کنترل‌کننده C1 تا لحظه رسیدن به حد آستانه شیب صعودی داشته و سپس با انجام مهاجرت، شیب نمودار C1 کاهش و شیب نمودار C2 افزایش می‌یابد. لحظاتی پس از توازن بار در هر دو کنترل‌کننده با شیب تقریباً ثابت و نزدیک به هم عمل می‌کنند.



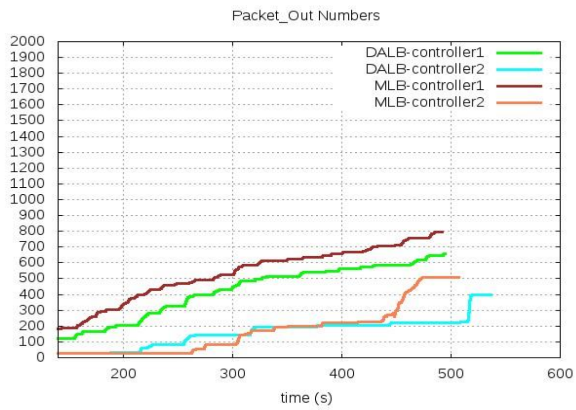
شکل ۴: مقایسه تعداد بسته‌های Packet - In دریافتی

۴-۳-۲- تأثیر بر میانگین بسته‌های Packet-Out خروجی

در شکل ۵، عملکرد روش پیشنهادی و روش DALB از لحاظ میانگین تعداد بسته‌های Packet-Out خروجی مقایسه شده است. همان‌طور که مشاهده می‌گردد در هر دو روش کنترل‌کننده اول تا لحظه تجاوز بار از حد آستانه (ثانیه ۴۴۰) با پردازش بسته‌های ارسالی از طرف سوئیچ‌های خودشان، دارای شیب صعودی در تولید بسته‌های Packet-Out می‌باشند. بعد از این زمان، در روش پیشنهادی به‌دلیل کاهش بار ورودی کنترل‌کننده اول در نتیجه انجام مهاجرت، با شیب کم‌تر از قبل به ارسال بسته‌های Packet-Out می‌پردازد. این در حالی است که در روش DALB، به‌دلیل وقوع سربار در کنترل‌کننده C1 و عدم اجرای توازن بار بین کنترل‌کننده‌ها نمودار Packet-Out کنترل‌کننده C1 از لحظه وقوع سربار با شیب افزایشی کم‌تری نسبت به Packet-In های ورودی پیش می‌رود. این امر بیانگر آن است که با توجه به عدم انجام مهاجرت، توان پاسخ‌دهی C1 پس از وقوع سربار کاهش یافته که سبب مصرف حافظه بالاتر و تأخیر پردازشی بیش‌تر در کنترل‌کننده C1 در روش DALB می‌گردد.

مطابق شکل، در روش DALB کنترل‌کننده C2 با همان نرخ ورود بسته‌های Packet-In دریافتی از سوئیچ S3 بسته‌های Packet-Out را ارسال می‌کند که نشان‌دهنده عدم انتقال بار سوئیچ دیگری جهت پردازش به این کنترل‌کننده می‌باشد. در مقایسه، در روش پیشنهادی به‌دلیل پذیرش بار سوئیچ مهاجر نمودار مربوط به کنترل‌کننده دوم با شیب صعودی و افزایش نرخ بسته‌های Packet-Out روبرو می‌گردد. هم‌چنین، پس از ارسال بسته‌های Packet-Out مربوطه، با توجه به تعداد بسته‌های دریافتی باقی‌مانده با شیب ثابت به پردازش آن‌ها می‌پردازد.

علت اختلاف شیب نمودار مربوط به کنترل‌کننده دوم در روش پیشنهادی با بار سوئیچ مهاجر پس از انجام توازن بار در این است که ممکن است در زمان مهاجرت، تعدادی از بسته‌های مربوط به سوئیچ مهاجر توسط کنترل‌کننده C1 پردازش شده باشند. در نتیجه، کنترل‌کننده C2 بسته‌هایی را پردازش می‌کند که تاکنون Packet-Out برای آن‌ها ارسال نشده باشد.

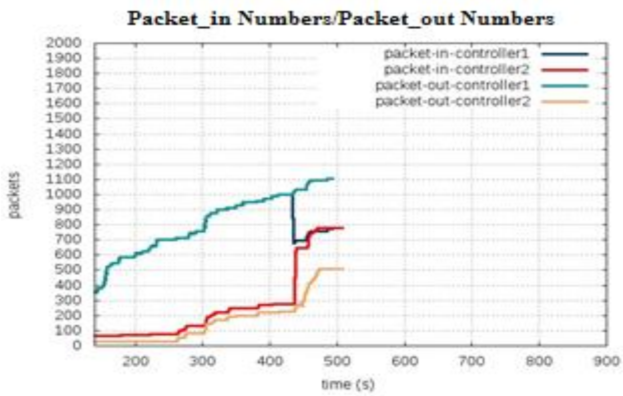


شکل ۵: مقایسه تعداد بسته‌های **packet-out** ارسالی

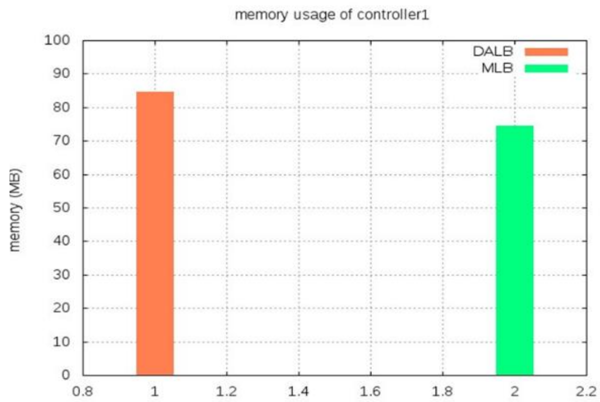
به‌منظور مقایسه ساده‌تر نرخ ارسال بسته‌های Packet-Out نسبت به بسته‌های Packet-In، شکل ۶ نمودار Packet-Out را برحسب Packet - In برای روش پیشنهادی نشان می‌دهد.

۴-۳-۳- تأثیر بر میانگین حافظه مصرفی در کنترل‌کننده اول

در شکل ۷، میانگین حافظه مصرفی کنترل‌کننده اول در دو روش بعد از انجام مهاجرت با هم مقایسه شده است. مطابق شکل، با توجه به عدم انجام مهاجرت در روش DALB، میزان حافظه مصرفی C1 در طی زمان و پس از این‌که با سربار روبرو شد افزایش می‌یابد. اما در روش پیشنهادی با اجرای توازن بار و مهاجرت سوئیچ، میزان حافظه مصرفی لحظاتی پس از انتقال سوئیچ، نسبت به قبل از مهاجرت کاهش می‌یابد. به‌عبارتی، الگوریتم پیشنهادی با کاهش حدود ۴۰ درصد حافظه مصرفی کنترل‌کننده C1 پس از مهاجرت نسبت به قبل از آن (مطابق نتایج دیگر) و کاهش حدود ۱۵ درصد میانگین مصرف حافظه نسبت به روش DALB سبب بهره‌وری مناسب‌تری از منابع موجود می‌گردد.



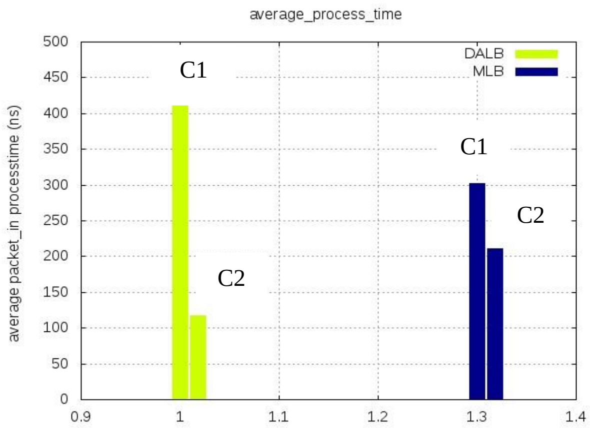
شکل ۶: نمودار تعداد بسته‌های Packet-Out نسبت به تعداد بسته‌های Packet-In برای روش پیشنهادی



شکل ۷: مقایسه تأثیر بر میانگین میزان حافظه مصرفی کنترل‌کننده C1

۴-۳-۴- تأثیر بر زمان پردازش بسته‌ها

نمودار شکل ۸، به بررسی تأثیر روش پیشنهادی و روش DALB بر روی میانگین زمان پردازش بسته‌ها می‌پردازد. در روش پیشنهادی، میانگین زمان پردازش یا به‌عبارتی تأخیر پردازش در کنترل‌کننده C1 نسبت به روش DALB به‌طور قابل توجهی کم‌تر بوده است. دلیل این امر عدم انجام مهاجرت سوئیچ در روش DALB و ازدحام ناشی از آن در کنترل‌کننده اول می‌باشد. هم‌چنین، با مقایسه نمودار کنترل‌کننده C2 در هر دو روش می‌توان مشاهده نمود که میانگین زمان پردازش در روش پیشنهادی با توجه به پذیرش بار سوئیچ مهاجر توسط کنترل‌کننده دوم از روش DALB بیش‌تر است. این امر با توجه به قرار گرفتن میانگین زمان پردازش در محدوده قابل قبول به‌عنوان نکته منفی برای روش پیشنهادی محسوب نمی‌شود و صرفاً مؤید پذیرش مقداری از بار کنترل‌کننده اول توسط کنترل‌کننده دوم در این روش می‌باشد.



شکل ۸: مقایسه تأثیر بر میانگین زمان پردازش بسته‌ها

۵ -بحث و بررسی

نتایج ارزیابی‌های صورت گرفته در محیط تست واقعی نشان می‌دهد که روش پیشنهادی منجر به توازن تعداد بسته‌های دریافتی در دو کنترل‌کننده، نرخ خروج بسته‌های Packet-Out متناسب با بسته‌های ورودی در دو کنترل‌کننده، بهبود میانگین میزان حافظه مصرفی کنترل‌کننده مبدأ و توازن زمان پردازش بسته‌ها در دو کنترل‌کننده شده است.

سربار محاسباتی الگوریتم پیشنهادی و الگوریتم مورد مقایسه به‌ازای هر کنترل‌کننده و در هر بازه دوره‌ای بررسی حد آستانه میزان بار هر کنترل‌کننده، در بهترین حالت (عدم احراز شرایط مربوط به حد آستانه بار و نرخ توازن بار) برابر با ثابت C در هر دو الگوریتم می‌باشد. در بدترین حالت ممکن (مثبت بودن همه شرایط لحاظ شده جهت اجرای فرآیند توازن بار و انجام مهاجرت)، این سربار محاسباتی در هر کنترل‌کننده و در هر بازه دوره‌ای بررسی حد آستانه میزان بار، شامل مقایسه با بار سایر کنترل‌کننده‌ها جهت انجام توازن بار در هر زمان توسط تنها یک کنترل‌کننده، انتخاب سوئیچ مهاجر و انتخاب کنترل‌کننده مقصد می‌باشد که با n کنترل‌کننده و m سوئیچ در بدترین حالت از مرتبه $o(n^2m)$ خواهد بود. در مقابل، سربار محاسباتی روش مورد مقایسه در بدترین حالت از مرتبه $o(nm)$ می‌باشد. این سربار محاسباتی بیشتر در الگوریتم پیشنهادی در بدترین حالت ناشی از توجه به انجام تنها یک مهاجرت در هر زمان (انتخاب کنترل‌کننده با

بیشترین بار جهت توازن بار) به‌منظور جلوگیری از انتخاب یک کنترل‌کننده مقصد توسط چند کنترل‌کننده مبدأ و ایجاد سربار در کنترل‌کننده مقصد و متعاقب آن نیاز به مهاجرت‌های متوالی می‌باشد. لذا، با وجود بالاتر بودن مرتبه زمانی الگوریتم پیشنهادی نسبت به الگوریتم مورد مقایسه در بدترین حالت، انتظار می‌رود تعداد بارهایی که الگوریتم پیشنهادی در شرایط انجام مهاجرت (بدترین حالت) قرار می‌گیرد به‌مراتب کمتر از الگوریتم مورد مقایسه باشد. بنابراین، پیچیدگی محاسباتی الگوریتم پیشنهادی نسبت به الگوریتم مورد مقایسه در کل زمان قابل قبول یا حتی بهتر خواهد بود. این در حالی است که روش پیشنهادی معیارهای ارزیابی مطرح شده در مقاله را نیز نسبت به روش مورد مقایسه به‌طور چشمگیری بهبود داده است.

به‌عنوان جمع‌بندی ویژگی‌های منحصر به فرد روش پیشنهادی می‌توان به نکات زیر اشاره نمود. روش پیشنهادی، تنها به مهاجرت بار یک کنترل‌کننده در هر بازه زمانی می‌پردازد و بدین ترتیب از مهاجرت‌های احتمالی متوالی ناشی از وقوع سربار در کنترل‌کننده مقصد جلوگیری می‌نماید. به‌علاوه، به‌منظور اطمینان از کاهش بار در کنترل‌کننده‌ای که دچار سربار شده و هم‌چنین جلوگیری از بروز سربار در کنترل‌کننده مقصد، پس از انتخاب سوئیچ در صورتی عمل مهاجرت انجام می‌گردد که پس از مهاجرت و انتقال بار سوئیچ منتخب، انحراف بار از میانگین بار برای هر دو کنترل‌کننده مبدأ و مقصد نسبت به قبل از مهاجرت کاهش یابد. در غیر این صورت، اگر انحراف از میانگین بعد از مهاجرت بیش‌تر یا مساوی قبل از مهاجرت گردد، عملیات مهاجرت کارآمد نخواهد بود و ترجیحاً عملیات توازن بار صورت نمی‌گیرد. هم‌چنین به‌منظور اعمال کارایی قابل قبول و بهبود معیارهای شبکه، می‌توان با در نظر گرفتن یک حد آستانه مهاجرت (T_M) بین هزینه مهاجرت و میزان کاهش در انحراف میانگین بعد از مهاجرت، تعادل برقرار نمود. در این صورت، در صورتی یک مهاجرت قابل قبول خواهد بود که میزان کاهش انحراف از میانگین بار هر دو کنترل‌کننده پس از مهاجرت از این حد آستانه بیش‌تر باشد. به‌عبارت دیگر کاهش کم‌تر از این مقدار از نظر هزینه زمانی و پردازش‌های موردنیاز مقرون به صرفه نخواهد بود.

در این روش تضمین می‌گردد که با انتخاب سوئیچ بهینه برای مهاجرت، پس از مهاجرت، بار کنترل‌کننده درگیر سربار از LT کم‌تر است. هم‌چنین بار همه کنترل‌کننده‌ها پس از عملیات مهاجرت و توازن بار، در محدوده میانگین می‌باشد. درنتیجه، انتخاب سوئیچ برای مهاجرت موفق است و عمل توازن بار به‌درستی صورت گرفته است.

۶- نتیجه‌گیری

در این مقاله، به ارائه روشی کارآمد، توزیعی و پایدار برای توازن بار پویا بین کنترل‌کننده‌ها در شبکه‌های مبتنی بر نرم‌افزار پرداختیم. راه‌حل پیشنهادی با توجه به امکان مهاجرت بار تنها یک کنترل‌کننده در هر زمان و انتخاب سوئیچ مهاجر با لحاظ بهبود در توازن بار، از وقوع سربار در کنترل‌کننده مقصد و ناپایداری و مهاجرت‌های متوالی ناشی از آن جلوگیری می‌نماید. هم‌چنین، با لحاظ هزینه ناشی از مهاجرت، تعادل بین کارایی شبکه و هزینه انجام مهاجرت را حفظ می‌نماید.

به‌عنوان کارهای آتی در نظر داریم مسئله صرفه‌جویی انرژی را به‌طور هم‌زمان با مسئله توازن بار در نظر بگیریم. به‌عبارتی، با توجه به پویایی ترافیک شبکه، در برخی مواقع می‌توان با خاموش کردن تعدادی از کنترل‌کننده‌ها ضمن حفظ کیفیت سرویس به صرفه‌جویی انرژی پرداخت. در این صورت، روشن کردن مجدد تعدادی از کنترل‌کننده‌های خاموش در صورت افزایش ترافیک و توازن بار مجدد پس از این عملیات باید مدنظر گرفته شود. به‌عنوان کار آتی دیگر در نظر داریم به ارائه روشی جهت توازن بار بین سرورها در مراکز داده‌ای که به‌واسطه کنترل‌کننده‌های توزیعی مدیریت می‌شوند بپردازیم.

مراجع

[1] N. McKeown, T. Anderson., H. Balakrishnan., G. Parulkar, L. Peterson., J. Rexford., S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," SIGCOMM CCR, vol. 38, no.2, pp 69-74, 2008.

[2] W. Xia, Y. Wen., C.H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp 27-51, 2015.

[3] S. Kaur, K. Kumar, J. Singh., and N.S. Ghumman, "Round-Robin Based Load Balancing in Software Defined Networking," IEEE 2nd International Conference (INDIACom), pp. 2136–2139, 2015.

[4] H. Zhang, and G. Xiao, "SDN-based load balancing strategy for server cluster In Cloud Computing and Intelligence Systems (CCIS)," IEEE 3rd International Conference, pp. 662-667, 2014.

[5] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, "Onix: A distributed control platform for large-scale production networks," OSID, vol. 10, pp. 1-6, 2010.

[6] A. Tootoonchian and Y. Ganjali, "Hyper flow: A distributed control plane for OpenFlow," In Proceedings of the 2010 internet network management conference on Research on enterprise networking, USENIX Association, pp. 3-3, 2010.

[7] D. Vinayagamurthy and J. Balasundaram, "Load Balancing between Controllers," Technical report, Department of Computer Science, University of Toronto, 2012.

[8] Y. Hu, W. Wang, X. yang Gong, X. Que, and S. Cheng. "Balance flow: controller load balancing for OpenFlow networks," Cloud Computing and Intelligent Systems (CCIS), IEEE 2nd International Conference on, vol. 2, pp. 780-785, 2012.

[9] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, T. Gayraud, "Software-Defined Networking: Challenges and research opportunities for Future Internet," Elsevier Journal of computer networks, vol. 75, pp. 453-471, 2014.

[10] D. L. Tennenhouse, D. Wetherall, ”Towards an active network architecture,” IEEE DARPA Active Networks Conference and Exposition, pp. 2-15, 2002.

[11] Y. Kyung, K. Hong, T.M. Nguyen, S. Park, and J. Park, “A load distribution scheme over multiple controllers for scalable SDN," In *Ubiquitous and Future Networks (ICUFN), IEEE Seventh International Conference on*, pp. 808-810, 2016.

[12] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R.R. Kompella, “ElastiCon: an elastic distributed SDN controller,” In Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems, pp. 17-28, 2014.

[13] Y. Zhou, M. Zhu, L. Xiao, L.Ruan, W. Duan, D. Li., R. Liu, and M. Zhu, “A Load Balancing Strategy of SDN Controller Based on Distributed Decision,” Trust, Security and Privacy in Computing and Communications (TrustCom), IEE 13th International Conference, pp. 851-856, 2014.

[14] Floodlight Project. http://www.projectfloodlight.org/.

[15] “OpenFlow Switch Specification Version 1.3.3 (Protocol version 0x04)”, ONF TS-015, September 27, 2013.

[16] D. Erickson, "The beacon OpenFlow controller," In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 13-18, 2013.

[17] www.bigswitch.com

[18] www.mininet.org

زیرنویس‌ها

[1] Software Defined Networks (SDN)

[2] Control Plane

[3] Data Plane

[4] Cloud Networking

[5] Slicing

[6] Virtualization

[7] Single Point of Failure

[8] Work Load

[9] Load Threshold

[10] Balance Rate

[11] Target Response Time

[12] Super Controller

[13] Per Flow

[14] Distributed Adaptive Load Balancing

[15] Load Measurement

[16] Load Collector

[17] Decision Maker

[18] Replicated Data Base

[19] Event Based

[20] Synchronization Message

[21] High Availability Transport Connection Protocol