Journal of Applied And Basic Machine Intelligence Research (ABMIR)

پژوهش‌های نظری و کاربردی هوش ماشینی

سال دوم، شماره دوم، پاییز و زمستان ۱۴۰۳، صفحات ۱۱۰ - ۹۵

ارائه روشی جهت کنترل ازدحام مبتنی بر مهاجرت سوئیچ در شبکه‌های نرم‌افزار محور با استفاده از الگوریتم بهبودیافته کرکس آفریقائی

محمدرضا جناب زاده[۱]، وحید آیت‌اللهی تفتی[۲*]، محمدرضا ملاخلیلی میبدی[۳]، محمدرضا ملاحسینی اردکانی[۳]

[۱] دانشجوی دکترای گروه مهندسی کامپیوتر، واحد میبد، دانشگاه آزاد اسلامی، میبد، ایران

[۲] استادیار گروه مهندسی کامپیوتر، واحد تفت، دانشگاه آزاد اسلامی، تفت، ایران

[۳] استادیار گروه مهندسی کامپیوتر، واحد میبد، دانشگاه آزاد اسلامی، میبد، ایران

مقاله پژوهشی

تاریخ دریافت:
۱۴۰۳/۱۰/۲۹
تاریخ پذیرش:
۱۴۰۴/۱/۱۷

کلیدواژه‌ها:
الگوریتم کرکس آفریقایی، بهینه‌سازی، شبکه‌های نرم‌افزار محور، کنترل ازدحام، مهاجرت سوئیچ

نویسنده‌مسئول:
vahid.ayat@iau.ac.ir

چکیده

شبکه‌های نرم‌افزارمحور به دلیل قابلیت برنامه‌ریزی و انعطاف‌پذیری بالا، به‌عنوان راهکاری نوین در مدیریت شبکه‌های بزرگ و پیچیده مطرح‌شده‌اند. بااین‌حال، در شبکه‌های چنددامنه که از چندین کنترل‌کننده برای بهبود عملکرد و افزایش مقیاس‌پذیری استفاده می‌کنند، چالش‌های جدی ازجمله مدیریت ازدحام و توازن بار میان کنترل‌کننده‌ها پدید می‌آید. این مقاله، باهدف حل مسئله ازدحام، روشی مبتنی بر مهاجرت پویا و هدفمند سوئیچ‌ها را برای توزیع ترافیک و کاهش فشار بر کنترل‌کننده‌ها ارائه می‌دهد. روش پیشنهادی از یک الگوریتم بهبودیافته کرکس آفریقایی برای مدیریت بار کنترل‌کننده‌ها بهره می‌گیرد. در این روش، ابتدا میزان بار سوئیچ‌ها و کنترل‌کننده‌ها اندازه‌گیری شده و در صورت نیاز، سوئیچ‌های پُربار به کنترل‌کننده‌هایی با ظرفیت بیشتر منتقل می‌شوند. عملکرد روش پیشنهادی با دو روش مشابه مقایسه شده است. نتایج شبیه‌سازی نشان می‌دهد که روش پیشنهادی توان عملیاتی شبکه را به میزان ۱۵ درصد افزایش داده و لرزش را حدود ۲۰ درصد بهبود داده است. این نتایج نشان‌دهنده کارایی بالاتر روش پیشنهادی در بهبود عملکرد شبکه است.

: 10.22034/ABMIR.2025.22659.1093

E-ISSN: 2821-2037

۱- مقدمه

شبکه‌های نرم‌افزار محور[1] به‌عنوان یک الگوواره جدید در مهندسی شبکه معرفی‌شده‌اند که به دلیل قابلیت برنامه‌ریزی و کنترل متمرکز، توانسته‌اند چالش‌های موجود در شبکه‌های سنتی را کاهش دهند. شبکه‌های نرم‌افزار محور با جدا کردن لایه کنترل از لایه داده و استفاده از کنترل‌کننده‌های مرکزی، مدیریت بهینه‌تری بر منابع شبکه فراهم می‌کنند و امکان کنترل دقیق و انعطاف‌پذیر بر ترافیک شبکه را به وجود می‌آورند. این خصوصیت، همچنین باعث می‌شود مدیریت و کنترل شبکه‌های پیچیده و مقیاس بزرگ راحت‌تر انجام شود [۱].

یکی از چالش‌های اساسی در این شبکه‌ها، کنترل ترافیک و ازدحام است [۲]. ازدحام، یک وضعیت شبکه است که در آن میزان نیاز منابع، بیش از ظرفیت موجود باشد و این امر منجر به از دست رفتن بسته‌ها و درنتیجه ارسال مجدد بسته‌ها می‌شود. کنترل ازدحام در شبکه‌های نرم افزارمحور به‌ویژه در شرایطی که بار ترافیکی به‌طور ناگهانی افزایش می‌یابد، اهمیت بیشتری پیدا می‌کند. روش‌های مختلفی برای کنترل ازدحام در این شبکه‌ها پیشنهادشده است. تاکنون روش‌های مختلفی برای کنترل ازدحام پیشنهادشده است که می‌توان به روش‌هایی نظیر تغییر توپولوژی شبکه، استفاده از جدول جریان و گروه، یافتن مسیر جایگزین با کمترین بار ترافیکی و یا روش مهاجرت سوئیچ اشاره نمود. درروش مهاجرت سوئیچ، انتقال بار ترافیکی از سوئیچ‌های شلوغ به سوئیچ‌های کم‌بارتر انجام می‌شود. این فن می‌تواند به بهبود عملکرد شبکه و کاهش تأخیر کمک کند و درنتیجه کیفیت خدمات را افزایش دهد [۳]. تحقیقات اخیر نشان داده‌اند که استفاده از فن‌های یادگیری ماشین و هوش مصنوعی در کنترل ازدحام نیز می‌تواند به بهبود عملکرد شبکه کمک کند [٤]. بااین‌حال، چالش‌هایی همچون پیچیدگی پیاده‌سازی و نیاز به منابع محاسباتی بیشتر در این زمینه وجود دارد که نیاز است روش‌های بهتری در این زمینه طراحی گردند.

این پژوهش با تمرکز بر مسئله کنترل ازدحام در شبکه‌های نرم‌افزارمحور چند دامنه، روشی مبتنی بر الگوریتم بهبودیافته کرکس آفریقایی (AVOA) ارائه می‌دهد. این الگوریتم به‌گونه‌ای طراحی‌شده است که با بررسی مداوم بار کنترل‌کننده‌ها و سوئیچ‌ها، در صورت بروز ازدحام، سوئیچ‌های پُربار را به کنترل‌کننده‌های دیگر منتقل می‌کند و بدین ترتیب از کاهش کارایی و افزایش تأخیر جلوگیری می‌نماید. نتایج حاصل از شبیه‌سازی‌ها نشان می‌دهد که روش پیشنهادی ازنظر معیارهایی مانند لرزش، تأخیر، توان عملیاتی، و زمان رفت‌وبرگشت بهبود قابل‌توجهی نسبت به روش‌های مورد مقایسه دارد.

مشارکت‌های اصلی این تحقیق در زیر خلاصه می‌شود:

- ارائه یک چارچوب پیشنهادی برای کنترل بهینه ازدحام در شبکه‌های نرم‌افزارمحور چند دامنه
- انتخاب بهینه سوئیچ (ها) برای مهاجرت در لایه داده
- ارائه یک مکانیسم مهاجرت سوئیچ بین کنترل‌کننده‌ها با رعایت تناسب بین هزینه مهاجرت و آستانه ازدحام با استفاده از الگوریتم بهبودیافته کرکس آفریقائی در لایه کنترل
- مقایسه روش پیشنهادی برای کنترل ازدحام با روش‌های مشابه

ادامه ساختار مقاله به شرح زیر است. در بخش دوم ادبیات تحقیق موردبررسی قرارگرفته است. بیان مسئله و روش پیشنهادی در بخش سوم ارائه‌شده است. بخش چهارم به نتایج حاصل از پیاده‌سازی روش پیشنهادی و ارزیابی نتایج و مقایسه با روش‌های پیشین اختصاص دارد و درنهایت در بخش پنجم به بحث و نتیجه‌گیری و پیشنهادی برای کارهای آینده پرداخته می‌شود.

۲- ادبیات تحقیق

در سال‌های اخیر، ظهور شبکه‌های نرم‌افزارمحور چشم‌انداز مدیریت شبکه را تغییر داده است، به‌ویژه درزمینه توازن بار و کنترل ازدحام. حوزه قابل‌توجهی از تحقیقات بر مهاجرت سوئیچ به‌عنوان یک استراتژی برای افزایش مقیاس‌پذیری و کارایی معماری‌های شبکه نرم‌افزارمحور تمرکز دارد. این مقاله، یافته‌های اخیر در مورد فن‌های مهاجرت سوئیچ و پیامدهای آن‌ها برای کنترل ازدحام و

[1] Software Defined Networks (SDN)

توازن بار در شبکه‌های نرم‌افزارمحور چند دامنه‌ای را بررسی می‌کند. مهاجرت سوئیچ به‌عنوان یک مکانیسم حیاتی برای حل مشکل ازدحام و عدم توازن بار در میان کنترل‌کننده‌های توزیع‌شده شناسایی‌شده است.

در پژوهش [۵] آدکویا و همکاران به مسئله مهاجرت سوئیچ‌ها در جریان‌های پرترافیک پرداخته است. الگوریتم ISMDA برای شناسایی کنترل‌کننده‌های کم‌بار و انتخاب بهینه سوئیچ‌ها برای مهاجرت طراحی‌شده است. این روش با استفاده از ماژول‌های خاصی که بر روی کنترل‌کننده‌ها اجرا می‌شوند، وضعیت بار کنترل‌کننده‌ها را ارزیابی کرده و تصمیمات مهاجرت را بر اساس کاهش هزینه و افزایش توازن بار اتخاذ می‌کند. این روش باعث بهبود توان عملیاتی کنترل‌کننده‌ها و کاهش افت بسته‌ها در مقایسه با روش‌های مشابه شده ولی مشکل پیچیدگی محاسباتی بالا برای جریان‌های پرترافیک در آن وجود دارد. بختیاری و همکاران در [٦] استفاده از الگوریتم گرگ خاکستری را برای توازن بار در کنترل‌کننده‌های شبکه‌های نرم‌افزارمحور پیشنهاد داده‌اند. در این روش، مهاجرت سوئیچ‌ها به‌صورت پویا انجام می‌شود و از الگوریتم گرگ خاکستری برای انتخاب کنترل‌کننده مقصد استفاده می‌شود. معیارهای اصلی تصمیم‌گیری شامل وضعیت ترافیک شبکه و هزینه مهاجرت هستند. این روش زمان پاسخگویی را کاهش و توان عملیاتی کنترل‌کننده‌ها را افزایش می‌دهد. محدودیت در کنترل توپولوژی‌های پیچیده و همچنین پیاده‌سازی توسط شبیه‌سازها و استفاده از کتابخانه‌های کنترل‌کننده و شبیه‌ساز شبکه بجای نصب و پیاده‌سازی در کنترل‌کننده واقعی از نقاط ضعف این تحقیق است.

پارجاپاتی و همکاران نیز در پژوهش [۷] روشی مبتنی بر الگوریتم حریصانه برای کاهش تعداد مهاجرت‌ها و بهبود کارایی شبکه را پیشنهاد کرده‌اند. این روش با تعریف آستانه‌ای برای شناسایی کنترل‌کننده‌های پربار و کم‌بار، مهاجرت سوئیچ‌ها را مدیریت می‌کند. به‌جای مهاجرت بی‌رویه، روش OptiGSM سوئیچ‌های خاصی را که بیشترین تأثیر را در توازن بار دارند، انتخاب می‌کند بنابراین پژوهش کاهش تعداد مهاجرت‌ها و کاهش زمان پاسخگویی شبکه از دست‌آوردهای روش OptiGSM بوده ولی در آن حساسیت به نوسانات ترافیکی مشاهده می‌گردد. در تحقیق [۸] از مدل مهاجرت‌های جزئی سوئیچ‌ها برای بهبود توازن بار بین کنترل‌کننده‌ها استفاده می‌کند. روش FractionalLB بر اساس مدل بهینه‌سازی خطی طراحی‌شده و هدف آن کاهش هزینه‌های همگام‌سازی و افزایش بهره‌وری کنترل‌کننده‌ها است ولی دارای پیچیدگی در پیاده‌سازی بوده و نیاز به تنظیم دقیق پارامترها دارد.

لی و همکاران در پژوهش [۹] الگوریتمی ارائه کرده‌اند که با استفاده از اشتراک زمانی، ترافیک سوئیچ‌ها را بین چند کنترل‌کننده تقسیم می‌کند. این روش باهدف کاهش پدیده "پینگ‌پونگ" در مهاجرت‌های مکرر، تعداد مهاجرت‌های غیرضروری را کاهش داده و کارایی شبکه را افزایش می‌دهد ولی باعث مصرف بالای منابع سیستم می‌گردد.

یئو و همکاران در [۱۰] با استفاده ازروشی بر اساس یادگیری تقویتی و مدل‌سازی خطی، تصمیمات بهینه‌ای برای مهاجرت سوئیچ‌ها ارائه می‌دهند. این روش توانایی تنظیم بار کنترل‌کننده‌ها را با دقت بالا دارد و باعث بهبود دقت تصمیم‌گیری و کاهش هزینه‌های همگام‌سازی نسبت به کارهای مشابه می‌گردد ولی برای آموزش مدل نیازمند محاسبات پیچیده و تنظیمات دقیق پارامترها است. بابار و همکاران در [۱۱] روشی را باهدف مدیریت کیفیت خدمات در سامانه‌های حمل‌ونقل هوشمند پیشنهاد کرده‌اند. این روش با شبیه‌سازی شبکه خودروهای هوشمند به یک SDN و با استفاده از فن مهاجرت سوئیچ الگوریتم LBSMT را ارائه کرده که با بهینه‌سازی استفاده از منابع سخت‌افزاری مانندCPU و حافظه، تأخیر شبکه را کاهش می‌دهد ولی کاربرد آن محدود به شبکه‌های معادل است. ژو و همکاران نیز در پژوهش [۱۲] برای مدیریت بار دینامیک در کنترل‌کننده‌های توزیع‌شده روش BalConPlus را باهدف کاهش قطعی سرویس در هنگام مهاجرت و پایداری بیشتری در شبکه ارائه می‌دهند تا در هنگام مهاجرت سوئیچ‌ها از یک کنترل‌کننده به کنترل‌کننده دیگر درخواستی بی‌پاسخ نماند که این کار نیازمند محاسبات پیچیده و تنظیم دقیق پارامترهای شبکه است. در پژوهش [۱۳] ژیرو و همکاران، الگوریتم ISMM برای مهاجرت سوئیچ‌ها در شرایط بار بالا را طراحی کرده‌اند. این روش با کاهش افت بسته‌ها و افزایش سرعت پاسخگویی، عملکرد شبکه را بهبود می‌بخشد و باعث افزایش توان عملیاتی و کاهش افت

بسته‌ها در شبکه می‌گردد. در پژوهش [۱۴]، یک رویکرد یادگیری تقویتی عمیق آگاه از ازدحام و کیفیت خدمات برای مسیریابی چندمسیری در شبکه‌های نرم‌افزارمحور پیشنهادشده است. این پژوهش با استفاده از یادگیری تقویتی عمیق، به‌طور هوشمندانه چند مسیر بهینه را انتخاب کرده و ترافیک را بر اساس نیازهای جریان تخصیص می‌دهد. انتخاب بهینه مسیر با استفاده از یک تابع چند هدفه برای ایجاد یک خط‌مشی مسیریابی کارآمد است انجام می‌شود.

جدول (۱): مقایسه روش‌های پیشین

نویسندگان	سال	مسئله موردبررسی	راهکار پیشنهادی	مزایا	معایب
Xu et al. [12]	۲۰۱۹	مشکلات نگاشت استاتیک سوئیچ‌ها در توازن بار	روش‌های BalCon و BalConPlus	کاهش قطعی سرویس	نیازمند تنظیم دقیق پارامترهای شبکه
Adekoya et al. [5]	۲۰۲۰	مدیریت ازدحام جریان‌های پرترافیک	الگوریتم ISMDA	کاهش تأخیر و افت بسته‌ها	پیچیدگی در جریان‌های پرترافیک
Yeo et al. [10]	۲۰۲۱	تصمیم‌گیری دقیق مهاجرت سوئیچ برای توازن بار	الگوریتم SAR-LB	دقت بالا در تصمیم‌گیری	هزینه محاسباتی زیاد
Babbar et al. [11]	۲۰۲۲	بهبود کیفیت خدمات در سامانه‌های هوشمند	الگوریتم LBSMT	بهبود استفاده از CPU و کاهش زمان پاسخ	محدودیت در کاربرد
Bakhtiari et al. [6]	۲۰۲۲	کارایی پایین کنترل‌کننده‌ها در بارهای دینامیک	استفاده از الگوریتم گرگ خاکستری	بهبود زمان پاسخ و بهره‌وری	محدودیت در پیچیدگی توپولوژی
Prajapati et al. [7]	۲۰۲۳	کاهش سربار محاسباتی در مهاجرت سوئیچ	روش OptiGSM	کاهش تعداد مهاجرت‌ها	حساسیت به شرایط پرترافیک
Lei et al. [9]	۲۰۲۳	مدیریت بار دینامیک و کاهش مهاجرت‌های غیرضروری	الگوریتم TSSM	کاهش مهاجرت‌های اضافی	مصرف منابع بالا
Jiru et al. [13]	۲۰۲۳	مشکلات کنترل‌کننده‌های پرترافیک	الگوریتم ISMM	کاهش هزینه مهاجرت	وابستگی به ابزارهای شبیه‌سازی
Prajapati et al. [8]	۲۰۲۳	کاهش هزینه‌های همگام‌سازی	مهاجرت‌های جزئی سوئیچ	افزایش پایداری	هزینه پیاده‌سازی بالا
et al. Sanchez [14]	۲۰۲۵	ازدحام و کیفیت خدمات	روش یادگیری تقویتی عمیق	دقت بالا در تصمیم‌گیری	کارآیی پایین برای کنترل‌کننده‌های چند دامنه

۳- روش پیشنهادی

ازدحام، یک وضعیت شبکه است که در آن میزان نیاز منابع، بیش از ظرفیت موجود باشد و این امر منجر به از دست رفتن بسته‌ها و درنتیجه ارسال مجدد بسته‌ها می‌شود [۱۵]. وقتی‌که ازدحام در یک شبکه کامپیوتری وجود داشته باشد، هم‌زمان افزایش تأخیر در صف، افزایش از دست دادن و گم‌شدن بسته‌ها و افزایش ارسال مجدد بسته‌ها رخ خواهد داد. تاکنون روش‌های مختلفی برای کنترل ازدحام پیشنهادشده است که می‌توان به روش‌هایی نظیر تغییر توپولوژی شبکه، استفاده از جدول جریان و گروه، یافتن مسیر جایگزین با کمترین بار ترافیکی و درنهایت روش مهاجرت سوئیچ اشاره نمود. در این فصل به تشریح روش پیشنهادی جهت استفاده از مهاجرت سوئیچ باهدف کنترل ازدحام در یک شبکه نرم‌افزارمحور چند دامنه‌ای خواهیم پرداخت.

۳-۱ ساختار کلی مدل پیشنهادی

یک عامل کلیدی برای دستیابی به نگاشت پویا، نظارت بر منابع کنترل‌کننده‌ها و انتقال سوئیچ‌ها از کنترل‌کننده‌های دچار اضافه‌بار به کنترل‌کننده‌های کم‌بار است. انتقال سوئیچ پس از پشتیبانی از چند کنترل‌کننده در پروتکل OpenFlow V1.2 و توسعه یک پروتکل مهاجرت سوئیچ چهار فازه امکان‌پذیر شد [۱۶]. در شبکه‌های نرم‌افزارمحور چند کنترل‌کننده‌ای، هر کنترل‌کننده می‌تواند دریکی از نقش‌های Master، Slave یا Equal قرارگیرد. کنترل‌کننده اصلی تنها کنترل‌کننده‌ای در دامنه است که می‌تواند

جداول جریان سوئیچ‌ها را به‌روزرسانی کند. یعنی هر سوئیچ نمی‌تواند بیش از یک کنترل‌کننده اصلی داشته باشد، اما می‌تواند یک یا چند کنترل‌کننده Equal و Slave داشته باشد [۱۷] هنگامی‌که یک کنترل‌کننده اصلی از کار بیفتد، یک کنترل‌کننده Equal و Slave می‌تواند جایگزین شود. تغییر کنترل‌کننده اصلی یک سوئیچ، مهاجرت سوئیچ نامیده می‌شود.

با توجه به وضعیت غیر بهینه شبکه در زمان t (به‌عنوان‌مثال کم بودن منابع شبکه یا خرابی کنترل‌کننده)، مسئله مهاجرت سوئیچ (SMP) باید در زمان t+1 تعیین کند که کدام سوئیچ (ها) باید از کدام کنترل‌کننده (ها) به کدام کنترل‌کننده (ها) مهاجرت کنند به‌طوری‌که مجموعه‌ای از توابع شایستگی (مانند توان عملیاتی) به حداکثر برسد و مجموعه‌ای از توابع هزینه (مانند سربار لایه کنترل) به حداقل برسد، درحالی‌که از برخی محدودیت‌ها تبعیت می‌کند (مانند حداکثر ظرفیت کنترل‌کننده‌ها).

بنابراین، یک راه‌حل برای SMP قرار است سه مجموعه ناشناخته را پیدا کند: کنترل‌کننده (های) مبدأ مهاجرت، سوئیچ (ها) برای مهاجرت، و کنترل‌کننده (های) مقصد مهاجرت، که یک مشکل NP-hard است [۱۸] و حداقل به یک الگوریتم اکتشافی برای حل آن نیاز است.

پژوهش‌های زیادی وجود دارد که در آن‌ها از روش مهاجرت سوئیچ استفاده‌شده است. محققان با در نظر گرفتن موارد گوناگون و استفاده از آمارهای دریافتی از شبکه سعی در کنترل ازدحام با استفاده از مهاجرت سوئیچ داشته و هرکدام با در نظر گرفتن برخی از پارامترهای ایجاد سربار در شبکه به حل مشکل اشاره نموده و راهکارهایی برای رفع آن‌ها پیشنهاد کرده‌اند. لازم به ذکر است اکثر آن‌ها از برنامه متلب برای شبیه‌سازی شبکه و پیاده‌سازی روش پیشنهادی خود استفاده نموده و روش آن‌ها در محیط واقعی و یا شبیه‌سازهای شبکه پیاده‌سازی و آزمایش نشده است.

در این پژوهش، برای پیاده‌سازی شبکه از کنترل‌کننده RYU و شبیه‌ساز Mininet استفاده‌شده است. همچنین، درروش پیشنهادی، برای کنترل ازدحام از مکانیزم مهاجرت سوئیچ مبتنی بر الگوریتم بهینه‌سازی کرکس آفریقایی (AVOA) بهره گرفته‌شده است. انتخاب الگوریتم کرکس آفریقایی درروش پیشنهادی به دلایل زیر صورت پذیرفته است:

-این الگوریتم به‌طور هم‌زمان به اکتشاف (جستجوی مناطق جدید) و بهره‌برداری (جستجوی موضعی در مناطق امیدوارکننده) در فضای جستجو می‌پردازد و این ویژگی باعث می‌شود AVOA در حل مسائل پیچیده با فضای جستجوی بزرگ عملکرد خوبی داشته باشد.

- AVOA با استفاده از مکانیسم‌های رفتاری کرکس‌ها تعادل بین اکتشاف و بهره‌برداری را به‌خوبی مدیریت می‌کند و از گیرکردن در بهینه‌های محلی جلوگیری می‌کند.
- AVOA ازنظر ساختاری ساده است و پارامترهای کمی برای تنظیم نیاز دارد. این موضوع باعث می‌شود اجرا و پیاده‌سازی آن نسبت به برخی الگوریتم‌های دیگر مانند الگوریتم ژنتیک یا PSO آسان‌تر باشد.
- سرعت همگرایی بالا
- تخصیص سوئیچ‌ها به کنترل‌کننده‌ها را می‌توان به‌عنوان یک کرکس درنظر گرفت و درنتیجه انتخاب بهینه سوئیچ‌ها جهت مهاجرت با توجه به‌سرعت همگرایی بالای این الگوریتم سریع‌تر انجام خواهد شد.

روش پیشنهادی شامل سه مرحله است. در مرحله ۱ به جمع‌آوری آمار شبکه که در ایجاد ازدحام و یا کنترل آن نقش دارند می‌پردازیم. مرحله ۲ شامل اعمال تابع هدف برای شناسایی کنترل‌کننده (های) درخطر ازدحام، سوئیچ (های) کاندید مهاجرت و کنترل‌کننده (های) هدف مهاجرت بوده و در مرحله سوم مهاجرت و درنتیجه جلوگیری از ایجاد ازدحام انجام می‌گردد.

۳-۲ مدل سیستم و فرموله کردن مسئله

فرمول چالش موردبحث و مدل سیستم در این بخش ارائه می‌گردند. برخی علائم ریاضی که در روابط استفاده‌شده در جدول ۲ نمایش داده‌شده‌اند.

جدول (۲): خلاصه‌ای از نمادهای ریاضی کلیدی

Notation	Description
G	Multidomain SDN network
U	Set of Controllers
V	Set of Switches
P_i	Number of packet-in messages from switch i

d_i	Propagation delay time
q_i	Packet-in process time
r_i	Rule installs time
LC_j	The load of controller j
LC_jS_i	The load of controller j from switch i
$LCpu_j$	CPU usage in controller j
$LMem_j$	Memory usage in controller j
γ_{max}	Maximum controller load threshold
γ_{min}	Minimum controller load threshold
$\delta_{C_iC_j}$	Load rate between controller i and controller j
LS_{ij}	The load of switch i in domain of controller j
S_M	Set of migration candidate switches
C_s	Set of migration source controllers
C_d	Set of migration destination controllers

۳-۲-۱ مدل سیستم

در این پژوهش از SDN با کنترل‌کننده‌های RYU و سوئیچ‌های OpenVSwitch استفاده‌شده است که به‌صورت G(U,V) تعریف‌شده و U مجموعه کنترل‌کننده‌ها و V مجموعه سوئیچ‌ها هستند. هر کنترل‌کننده نقش Master را برای سوئیچ‌های تحت نظارت خود و نقش Slave را برای سایر سوئیچ‌های شبکه دارد و S_{ij} به سوئیچ i ام از دامنه کنترل‌کننده j ام اشاره می‌کند. هر جریانی که وارد سوئیچ می‌شود ابتدا با قوانین موجود در جدول جریان که توسط کنترل‌کننده در آن پیاده شده، مقایسه شده و چنانچه با قانونی مطابقت داشت، سوئیچ طبق قانون موجود عمل می‌کند. چنانچه قانونی برای این جریان نبود، سوئیچ این جریان را به‌صورت Packet-in به کنترل‌کننده Master خودش ارسال می‌کند تا کنترل‌کننده قانونی برای آن تعیین و به‌صورت Packet-out جهت نصب در جدول جریان به سوئیچ ارسال نماید. علاوه بر بار تحمیلی از طرف سوئیچ‌های یک دامنه به کنترل‌کننده مسترشان که در این مقاله با LC_jS_i نشان داده‌شده وضعیت Cpu و $Memory$ کنترل‌کننده نیز در تعیین میزان بار کنترل‌کننده تأثیر مستقیم دارند.

۳-۲-۲ فرموله سازی

بار تحمیلی سوئیچ i به کنترل‌کننده j را تعداد بسته‌های ارسالی از سوئیچ در نظر گرفته‌ایم:

$$LC_jS_i = \sum PI_i \qquad (۱)$$

همچنین با توجه به تأثیر میزان Cpu و $Memory$ کنترل‌کننده بر بار آن و با در نظر گرفتن بار تحمیلی به کنترل‌کننده j از طرف دامنه آن‌که برابراست با مجموع بار تحمیلی از طرف سوئیچ‌های موجود در این دامنه، می‌توان بار کنترل‌کننده را طبق رابطه زیر تعریف نمود :

$$LC_j = \sum_{i=1}^{k} LC_jS_i + LCpu_j + LMem_j \qquad (۲)$$

که در آن k تعداد سوئیچ‌های متصل به کنترل‌کننده j است. با انجام مهاجرت در صورت تشخیص ازدحام، بار بعد از مهاجرت یک کنترل‌کننده ($\widehat{LC_j}$) برابر خواهد بود با بار فعلی به‌اضافه اختلاف بین بار سوئیچ خارج‌شده از دامنه کنترل‌کننده و بار سوئیچ واردشده به دامنه آن:

$$\widehat{LC_j} = LC_j + \Big(\sum_{j' \in U | j \neq j'} \sum_{i \in V} \Big(LC_{j'}S_i \times x_i^{j'j} - LC_jS_i \times x_i^{jj'}\Big)\Big) \qquad (۳)$$

که در آن U مجموعه کنترل‌کننده‌ها و V مجموعه سوئیچ‌های موجود در شبکه هستند. همچنین $x_i^{jj'}$ زمانی ۱ است که سوئیچ i از کنترل‌کننده j به کنترل‌کننده j' مهاجرت کرده باشد و در غیر این صورت صفر است.

اگر هزینه مهاجرت به‌عنوان واریانس بار تمامی کنترل‌کننده‌ها در نظر گرفته شود:

$$\vartheta = \frac{1}{n}\sum_{j=1}^{n}(\widehat{LC_j} - \overline{LC}) \qquad (٤)$$

هدف به حداقل رساندن هزینه مهاجرت خواهد بود این‌یک مسئله ILP است:

$$\text{Minimize:} \frac{1}{n}\sum_{j=1}^{n}(\widehat{LC_j} - \overline{LC}) \qquad (٥)$$

$$\text{Subject to:} \sum_{j \in U}\sum_{j' \in U} x_i^{jj'} \leq 1 \quad \forall i \in V \qquad (٦)$$

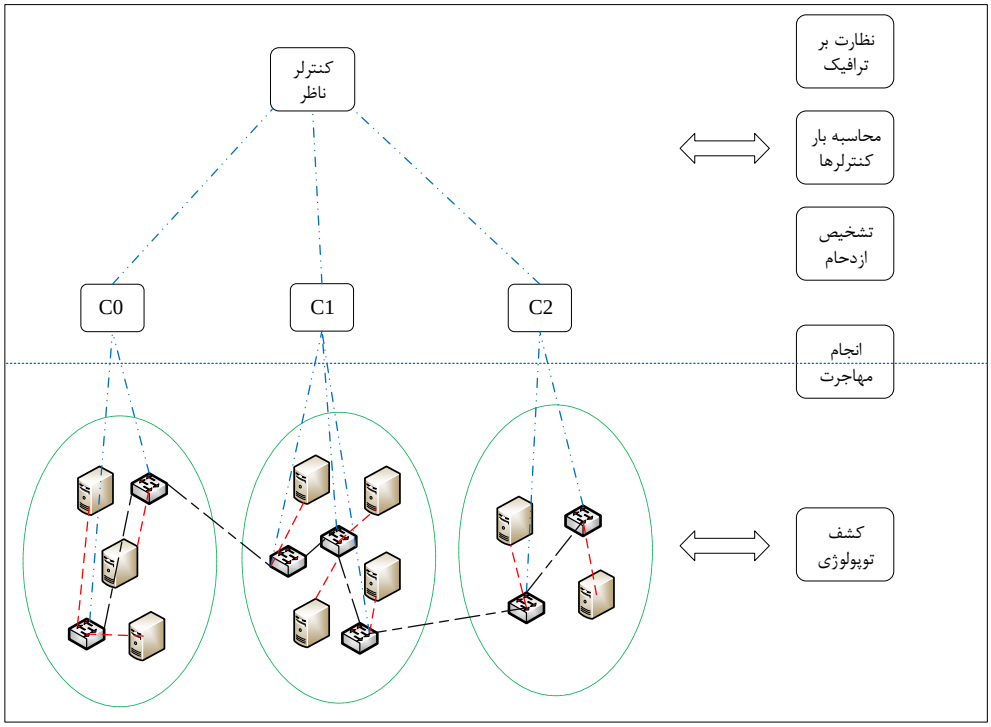
$$0 \leq \widehat{LC_j} \leq \gamma_{max} \qquad (٧)$$

$$x_i^{jj'} \in \{0,1\} \; \forall j, j' \in U, i \in V \qquad (٨)$$

رابطه (٦) بیانگر این است که هر سوئیچ فقط می‌تواند به یک کنترل‌کننده مهاجرت کند و رابطه (٧) تضمین‌کننده این مسئله است

که بار کنترل‌کننده بعد از مهاجرت از آستانه حداکثر بار مجاز کنترل‌کننده تجاوز نکند. رابطه (۸) نیز پارامتر تعیین‌کننده انجام مهاجرت یا عدم آن است.

برای شبکه‌های در مقیاس کوچک می‌توان این معادلات را حل نمود ولی برای شبکه‌های با مقیاس بزرگ‌تر یک روش فرا ابتکاری نیاز است. شکل ۱ چارچوب پیشنهادی را برای کنترل ازدحام نمایش می‌دهد.



شکل (۱): چارچوب روش پیشنهادی

۳-۳ مراحل پیاده‌سازی روش پیشنهادی

در این بخش، مراحل لازم برای پیاده‌سازی و اجرای طرح پیشنهادی ارائه می‌گردد. این طرح در سه مرحله: محاسبه بار سوئیچ‌ها، محاسبه بار کنترل‌کننده‌ها جهت تشخیص ازدحام و درنهایت، طرح مهاجرت سوئیچ ارائه خواهد شد.

مرحله اول: محاسبه بار سوئیچ‌ها

در اینجا یک SDN بنام $G(U,V)$ را با N کنترل‌کننده و M سوئیچ در نظر گرفته‌شده است به‌طوری‌که U مجموعه کنترل‌کننده‌ها و V مجموعه سوئیچ‌های موجود در شبکه می‌باشند:

$$U=\{C_1, C_2, \ldots, C_N\}, V=\{S_1, S_2, \ldots, S_M\} \quad (۹)$$

و هر سوئیچ تنها به یک کنترل‌کننده اصلی متصل است.

$$V_j = \{ \forall S_i (V | S_i (C_j, U_{j=1}^{N} V_j = V , \cap_{j=1}^{N} V_j = \emptyset , i=1,2,\ldots,M \ , j=1,2,\ldots,N\} \quad (۱۰)$$

همان‌طور که توضیح داده شد در صورت عدم تطابق جریان ورودی، سوئیچ این جریان را به‌عنوان packet-in به کنترل‌کننده Master ارسال می‌نماید و بنابراین بار ایجادشده توسط سوئیچ را در این حالت طبق رابطه (۱) معادل تعداد packet-in ارسالی به کنترل‌کننده در نظر می‌گیریم.

به‌این‌ترتیب می‌توان بار تحمیلی هر سوئیچ در دامنه هر کنترل‌کننده را محاسبه و سوئیچ یا سوئیچ‌های نامزد مهاجرت را انتخاب نمود.

مرحله دوم: محاسبه بار کنترل‌کننده‌ها، تشخیص ازدحام و تعیین نیاز به مهاجرت

در این مرحله دو حالت مختلف برای پیشگیری از وقوع ازدحام در نظر گرفته‌شده است:

حالت اول: تعریف حد آستانه برای حداکثر و حداقل بار مجاز هر کنترل‌کننده

در این حالت چنانچه بار یک کنترل‌کننده از حد آستانه γ_{max} بیشتر شد، این کنترل‌کننده به مجموعه کنترل‌کننده‌های خروجی مهاجرت اضافه می‌شود و سپس از مجموعه سوئیچ‌های تحت مدیریت این کنترل‌کننده، سوئیچ یا سوئیچ‌هایی برای مهاجرت انتخاب می‌شوند. از طرف دیگر چنانچه بار کنترل‌کننده از حد آستانه γ_{min} کمتر باشد به مجموعه کنترل‌کننده‌های مقصد مهاجرت اضافه‌شده و سوئیچ‌های کاندید مهاجرت، به این کنترل‌کننده مهاجرت داده می‌شوند و درصورتی‌که چنین کنترل‌کننده‌ای در شبکه موجود نبود، کنترل‌کننده با کمترین بار به‌عنوان مقصد مهاجرت در نظر گرفته خواهد شد.

حالت دوم: محاسبه تناسب بار کنترل‌کننده‌ها

درصورتی‌که در حالت اول کنترل‌کننده‌ای برای مبدأ مهاجرت پیدا نشد، جهت پیشگیری از وقوع ازدحام از حالت دوم استفاده می‌شود. در اینجا باید نسبت بار کنترل‌کننده‌ها(δ) را طبق رابطه (۱۱) به دست آورده و چنانچه این نسبت از حد آستانه$\frac{\gamma_{max}}{\gamma_{min}}$ بزرگ‌تر بود کنترل‌کننده‌های صورت و مخرج این کسر به‌عنوان کنترل‌کننده‌های مبدأ و مقصد مهاجرت انتخاب می‌شوند.

$$\delta_{C_iC_j} = \frac{LC_i}{LC_j} \qquad i,j = 1,2,\dots,N \qquad (۱۱)$$

چنانچه در هیچ‌کدام از این دو حالت شرط مهاجرت وجود نداشت به این معنی است که سیستم در شرایط متعادل قرار داشته و نیازی به مهاجرت نیست.

حالت سوم: فن مهاجرت سوئیچ و الگوریتم پیشنهادی

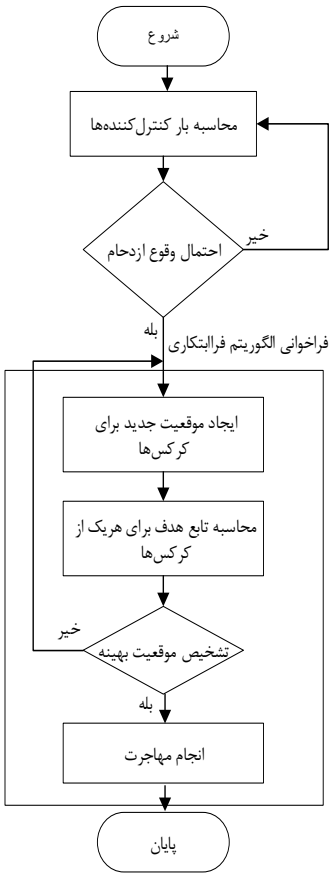
همان‌طور که بیان گردید مسئله مهاجرت سوئیچ یک مسئله NP-hard است. مراحل اول و دوم تا مرحله تشخیص ازدحام، به‌وسیله روابطی که توضیح داده‌شده، قابل انجام است لیکن انجام مرحله سوم و بهینه‌سازی مهاجرت سوئیچ به یک روش فرا ابتکاری نیاز دارد. به همین دلیل با بهره‌گیری از الگوریتم فرا ابتکاری کرکس آفریقایی و ایجاد تغییراتی در آن به حل مسئله پرداختیم که در ادامه توضیح داده خواهد شد.

۳-۴ کنترل ازدحام به کمک الگوریتم فرا ابتکاری AVOA بهبودیافته

۳-۴-۱ مدل کردن مسئله در AVOA

برای حل مسئله ازدحام در چنین شبکه‌ای ابتدا لازم است آن را در الگوریتم فرا ابتکاری موردنظر مدل کنیم. در الگوریتم AVOA تعدادی کرکس و یک طعمه داریم که هدف، بهینه‌سازی موقعیت کرکس‌ها نسبت به طعمه موردنظر است. طعمه که همان تابع هدف در الگوریتم است را مجموع اختلاف بار کنترل‌کننده‌ها با حد آستانه‌بالا و پایین در نظر می‌گیریم. طبق رابطه زیر:

$$Fitness = \sum_{j=1}^{m} max(0, LC_j - \gamma_{\max}) + max(0, \gamma_{\min} - LC_j) \qquad (۱۲)$$



شکل (۲): فلوچارت مهاجرت سوئیچ‌ها با استفاده از الگوریتم کرکس

هر موقعیت کرکس در زمان t که با $X(t)$ نمایش داده می‌شود نیز یک حالت از تخصیص سوئیچ‌ها به کنترل‌کننده‌ها است. به‌عنوان‌مثال $X_i(t) = [C_1, C_2, C_1, C_3, C_1, C_2]$ به این معنی است که در اختصاص i ام سوئیچ‌ها به کنترل‌کننده‌ها، S_1 به C_1 و S_2 به C_2 و S_3 به C_1 و S_4 به C_3 و S_5 به C_1 و S_6 به C_2 متصل است.

هدف در اینجا به دست آوردن بهترین $X_i(t)$ است که کمترین مقدار را در تابع هدف (رابطه ۱۲) دارا باشد. بنابراین می‌توان چارچوب فوق را برای حل مسئله کنترل ازدحام در نظر گرفت:

۳-۴-۲ بهبود الگوریتم AVOA جهت حل مسئله ازدحام

الگوریتم AVOA در حالت ساده در ۵ مرحله اجرا می‌گردد [۱۹]، ولی در حل مسئله موردنظر و با توجه به پیاده‌سازی مسئله در محیط شبکه و نه در نرم‌افزارهای شبیه‌سازی مانند متلب، باید تغییراتی در الگوریتم ایجاد کرد تا اولاً قابل‌استفاده در محیط شبکه باشد و دوما نیازها و شروط موردنظر را برای حل مسئله برآورده سازد. بدین ترتیب الگوریتم AVOA بهینه‌شده طبق مراحل زیر عمل خواهد کرد.

- ایجاد جمعیت اولیه

در ابتدای الگوریتم، جمعیتی از کرکس‌ها به‌طور تصادفی در فضای جستجو مقداردهی اولیه می‌شوند. هر کرکس نشان‌دهنده یک راه‌حل بالقوه برای مسئله بهینه‌سازی است. در این مرحله اختصاص اولیه سوئیچ‌ها به کنترل‌کننده‌ها انجام می‌شود و درواقع یک کرکس تعریف می‌شود. تخصیص اولیه در Mininet صورت می‌پذیرد.

- مرحله اکتشاف

مرحله اکتشاف مسئول تنوع بخشیدن به جستجو با اجازه دادن به کرکس‌ها برای کشف مناطق جدید فضای جستجو است. در طول این مرحله، موقعیت هر کرکس با استفاده از فرمول (۱۳) به‌روز می‌شود.

$$X_i(t+1) = X_i(t) + rand_1 * \left(X_{best}(t) - X_i(t)\right) + rand_2 * \left(X_{mean}(t) - X_i(t)\right) \quad (۱۳)$$

که در آن:

$X_i(t)$ موقعیت فعلی کرکس i در تکرار t است.

$X_{best}(t)$ موقعیت بهترین کرکس (یعنی کرکسی با بالاترین تناسب) در تکرار t است.

$X_{mean}(t)$ میانگین موقعیت همه کرکس‌ها در تکرار t است.

$rand_1$ و $rand_2$ اعداد تصادفی هستند که به‌طور یکنواخت بین ۰ و ۱ توزیع‌شده‌اند.

- فاز بهره‌برداری

مرحله بهره‌برداری بر اصلاح جستجو در اطراف شناخته‌شده‌ترین راه‌حل‌ها متمرکز است. در طول این مرحله، موقعیت هر کرکس بر اساس بهترین راه‌حلی که تاکنون پیداشده است به‌وسیله رابطه (۱۴) به‌روز می‌شود.

$$X_i(t+1) = X_{best}(t) + rand * \left(X_i(t) - X_{best}(t)\right) \quad (۱۴)$$

در این فاز باید قیودی را برای جدا کردن سوئیچ از یک کنترل‌کننده و اضافه کردن آن به کنترل‌کننده دیگر در نظر گرفت. روابط زیر به‌عنوان شروط اولیه برای حذف سوئیچ از و یا اضافه کردن آن به کنترل‌کننده در نظر گرفته‌شده‌اند:

برای انتخاب بهترین سوئیچ جهت مهاجرت از کنترل‌کننده مبدأ:

$$Select\ S_k = \arg\min_{S_i}\left(\gamma_{max} - R(S_i, C_j)\right) \quad (۱۵)$$

که در آن:

- S_k سوئیچ انتخابی برای مهاجرت است
- $R(S_i, C_j)$ بار کنترل‌کننده بعد از کاهش بار سوئیچ موردنظر است

و برای انتخاب کنترل‌کننده مقصد:

- ابتدا امکان اضافه شدن سوئیچ به کنترل‌کننده بررسی می‌شود:

$$A(S_i, C_j) = \begin{cases} 1 & if \quad LC_j + LS_i \le \gamma_{max} \\ 0 & otherwise \end{cases} \quad (۱۶)$$

- سپس، سوئیچ‌های حذف‌شده به کنترل‌کننده‌ای که بیشترین کاهش هزینه را ایجاد می‌کند، تخصیص داده می‌شوند:

$$Assign\ S_k\ to\ C_m = \arg\max_{C_j}\left(\frac{R(S_k, C_j)}{P(S_k, C_j)} * A(S_k, C_j)\right) \quad (۱۷)$$

که در آن:

$$P(S_k, C_j) = \begin{cases} 1 & if\ \ in\ X(t): S_k \in C_j\ and\ in\ X(t+1): S_k \in C_j \\ \alpha & else \end{cases} \quad (۱۸)$$

و $\alpha > 1$ یک ضریب مجازات است که مهاجرت از کنترل‌کننده اصلی را ناپسند می‌کند.

- تابع هدف

تابع هدف برای ارزیابی کیفیت موقعیت هر کرکس استفاده می‌شود. در رابطه (۱۲) یک تابع هدف برای حل مسئله پیشنهاد شد، اما با توجه به نیاز به حداقل رساندن مهاجرت سوئیچ‌ها بین کنترل‌کننده‌ها یک پارامتر پاداش و مجازات به آن اضافه‌شده تا درصورتی‌که تخصیص سوئیچ به کنترل‌کننده همان وضعیت قبلی آن باشد، مقداری به تابع هدف اضافه نگردد ولی اگر سوئیچ به یک کنترل‌کننده دیگر اختصاص داده‌شده بود مقداری را به‌عنوان مجازات به تابع هدف آن اضافه کند. طبق رابطه زیر:

$$Fitness = \sum_{j=1}^{m} max(0, LC_j - \gamma_{\max}) + max(0, \gamma_{\min} - LC_j) + \sum_{j=1}^{m} \sum_{S_k \in C_j} P(S_k, C_j) \quad (۱۹)$$

- به‌روزرسانی موقعیت بر اساس تابع هدف

در این مرحله، تخصیص جدید را به‌عنوان یک جواب در نظر می‌گیرد. تابع هدف آن را جهت مقایسه با تخصیص‌های بعدی ذخیره کرده و مجدداً به فاز بهره‌برداری برای تکرار مراحل به تعداد مشخص‌شده بازمی‌گردد. در هر تکرار، تابع هدف جدید را با قبلی مقایسه کرده و تابع هدف بهتر را به‌عنوان جواب و برای مقایسه با تابع هدف بعدی در نظر می‌گیرد. درنهایت و بعد از اتمام تکرار، بهترین تخصیص که به بهترین تابع هدف تعلق دارد، جواب نهائی خواهد بود.

بنابراین بهبودهای اعمال‌شده بر الگوریتم AVOA عبارت‌اند از:

- اولویت‌بندی بازگشت سوئیچ‌ها به کنترل‌کننده اصلی
- بهینه‌سازی تابع برازندگی
- انتخاب هوشمندانه سوئیچ‌ها برای حذف
- بررسی مجدد سوئیچ‌های حذف‌شده برای تخصیص به کنترل‌کننده‌های دیگر

تمام موارد فوق با تعریف توابعی به زبان پایتون و اضافه‌کردن آن‌ها به برنامه کنترل‌کننده ناظر پیاده‌سازی شده است.

الگوریتم زیر چگونگی انجام فرآیند را بیان می‌کند:

الگوریتم (۱): الگوریتم روش پیشنهادی

ورودی: G(U,V)

خروجی: G(U,V) بهینه‌شده

۱. برای همه کنترل‌کننده‌ها
۲. محاسبه بار سوئیچ‌ها و کنترل‌کننده‌ها طبق روابط (۱) و (۲)
۳. اگر بار کنترل‌کننده‌ای بیشتر از حد آستانه بالا بود به مجموعه کنترل‌کننده‌های مبدأ مهاجرت C_s اضافه شود
٤. برای همه کنترل‌کننده‌های عضو C_s
٥. انتخاب بهترین سوئیچ برای مهاجرت طبق رابطه (۱۵) و اضافه شدن به مجموعه سوئیچ‌های نامزد مهاجرت
٦. برای همه سوئیچ‌های نامزد مهاجرت
۷. پیدا کردن کنترل‌کننده مقصد برای انجام مهاجرت طبق روابط (۱٦) تا (۱۸)
۸. محاسبه تابع هدف برای هر وضعیت جدید طبق رابطه (۱۹)
۹. مقایسه توابع هدف و به دست آوردن بهترین موقعیت
۱۰. انجام مهاجرت

٤- پیاده‌سازی و ارزیابی

٤-۱ پیاده‌سازی

در این تحقیق برای پیاده‌سازی الگوریتم پیشنهادی از سیستم‌عامل لینوکس Ubuntu 20.0 که بر روی ماشین مجازی VMware با ۱٦ گیگابایت رم و Intel(R) Core(TM) i7-4750HQ CPU @ 2.00GHz نصب‌شده است استفاده کرده‌ایم. برای شبیه‌سازی محیط شبکه از شبیه‌ساز شبکه Mininet استفاده‌شده است.

شبکه موردنظر از ۳ دامنه که توسط ۳ کنترل‌کننده مدیریت می‌شوند تشکیل‌شده و نظارت بر عملکرد کل شبکه نیز بر عهده یک کنترل‌کننده ناظر است. ۱۰ سوئیچ و ۱۲ میزبان نیز در شبکه موجود می‌باشند که بین کنترل‌کننده‌ها توزیع‌شده‌اند.

کنترل‌کننده‌های مورداستفاده در این مدل، کنترل‌کننده RYU بوده و ارتباط بین کنترل‌کننده‌ها توسط پیام‌رسان مقیم حافظه Redis صورت می‌گیرد. درروش پیشنهادی تمام توابع با استفاده از کد پایتون پیاده‌سازی شده‌اند. کنترل‌کننده RYU یک کنترل‌کننده متن‌باز است که الگوریتم بهبودیافته کرکس آفریقائی به‌صورت یک کلاس با توابع موردنیاز به آن اضافه‌شده است. قبل از راه‌اندازی شبکه لازم است پارامترهایی که در برنامه مورداستفاده قرار می‌گیرند

مقداردهی گردند. این پارامترها عبارت‌اند از حد آستانه بالا، حد آستانه پایین، α و تعداد دفعات تکرار الگوریتم. حد آستانه بالا و پایین معمولاً به ترتیب ۸۰٪ و ۵۰٪ درصد حداکثر ظرفیت پردازش کنترل‌کننده و برای پیش‌گیری از وقوع ازدحام تعیین می‌گردند. به این منظور γ_{max} برابر ۱۳۰، γ_{min} برابر با ۸۰ منظور شده است. α نیز که به‌عنوان جریمه در صورت تخصیص سوئیچ به کنترل‌کننده‌ای غیر کنترل‌کننده فعلی و برای کاهش مهاجرت‌های اضافی است برابر با ۵ و تعداد تکرار نیز ۱۰ در نظر گرفته‌شده است.

ابتدا توسط مقلد شبکه Mininet شبکه موردنظر راه‌اندازی شده و سپس در سه ترمینال جداگانه کنترل‌کننده‌های سه‌گانه و در یک ترمینال دیگر نیز کنترل‌کننده ناظر که حاوی الگوریتم روش پیشنهادی است راه‌اندازی می‌گردند (شکل‌های ۳ (الف تا ه)).

٤-٢ ارزیابی

در ابتدا برای بررسی عملکرد صحیح تابع برازش[1] مقادیر تولیدشده توسط برنامه در یک فایل ذخیره و نمودار تغییرات آن در یک دوره تناوب موردبررسی قرار گرفت.

```
$ sudo python topo1.py
[sudo] password for test:
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Starting controllers
*** Starting switches
*** Starting CLI:
mininet> exit
*** Stopping 3 controllers
c0 c1 c2
*** Stopping 21 links
.....................
*** Stopping 10 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
*** Stopping 12 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Done
```

(الف)

```
$ ryu-manager --ofp-tcp-listen-port 6653 controller1.py
loading app controller1.py
loading app ryu.controller.ofp_handler
instantiating app controller1.py of SwitchMigration
instantiating app ryu.controller.ofp_handler of OFPHandler
s=== 2
s=== 1
s=== 3
s=== 4
key ---> value tcp:127.0.0.1:6653 -----> {'packet_in': {2: 0, 1: 0, 3: 0, 4:
 0}}
switches =  {2: 0, 1: 0, 3: 0, 4: 0}
```

(ب)

```
$ ryu-manager --ofp-tcp-listen-port 6654 controller2.py
loading app controller2.py
loading app ryu.controller.ofp_handler
instantiating app controller2.py of SwitchMigration
instantiating app ryu.controller.ofp_handler of OFPHandler
s=== 5
s=== 6
s=== 7
key ---> value tcp:127.0.0.1:6654 -----> {'packet_in': {5: 0, 6: 0, 7: 0}}
switches =  {5: 0, 6: 0, 7: 0}
```

(ج)

```
$ ryu-manager --ofp-tcp-listen-port 6655 controller3.py
loading app controller3.py
loading app ryu.controller.ofp_handler
instantiating app controller3.py of SwitchMigration
instantiating app ryu.controller.ofp_handler of OFPHandler
s=== 8
s=== 9
s=== 10
key ---> value tcp:127.0.0.1:6655 -----> {'packet_in': {8: 0, 9: 0, 10: 0}}
switches =  {8: 0, 9: 0, 10: 0}
```

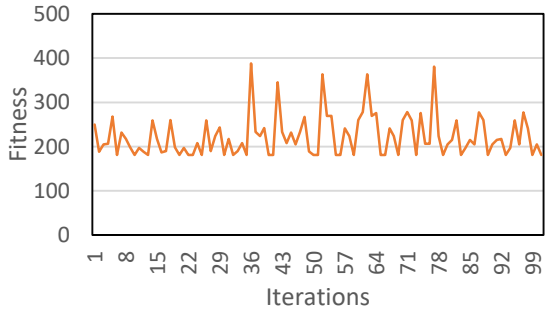
(د)

```
$ ryu-manager  master_controller_Avoa.py
loading app master_controller_Avoa.py
loading app ryu.controller.ofp_handler
instantiating app master_controller_Avoa.py of MasterController
 class master
instantiating app ryu.controller.ofp_handler of OFPHandler
 _monitor
 get_all_data
status=  {b'tcp:127.0.0.1:6654': {'packet_in': {5: 0, 6: 0, 7: 0}}, b'tcp:12
7.0.0.1:6653': {'packet_in': {2: 0, 1: 0, 3: 0, 4: 0}}, b'tcp:127.0.0.1:6655
': {'packet_in': {8: 0, 9: 0, 10: 0}}}
```

(ه)

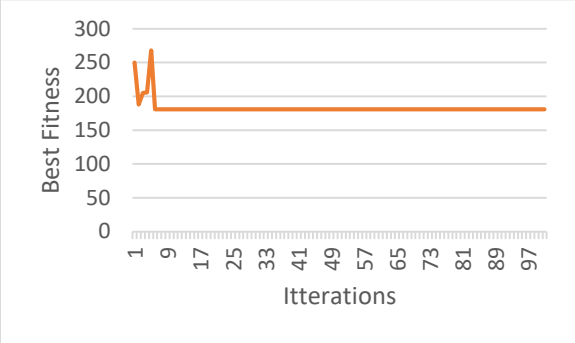
شکل (۳): مراحل راه‌اندازی شبکه و اجرای روش پیشنهادی

شکل (٤-الف) تغییرات تابع برازش در یک دوره تناوب تخصیص سوئیچ‌ها به کنترل‌کننده‌ها را نشان می‌دهد و همان‌گونه که در شکل (٤-ب) ملاحظه می‌گردد در هر تلاش، مقدار تابع برازش در تخصیص سوئیچ‌ها به کنترل‌کننده‌ها اندازه‌گیری شده و با مقدار قبلی مقایسه می‌گردد و درجایی که کمترین مقدار تابع برازش به‌دست‌آمده (تلاش شماره ٦ با مقدار ۱۸۱) مقادیر بعدی نادیده درنظر گرفته‌شده و این کم‌ترین مقدار تابع به‌عنوان بهترین تخصیص در نظر گرفته می‌شود.



(الف)

[1] Fitness

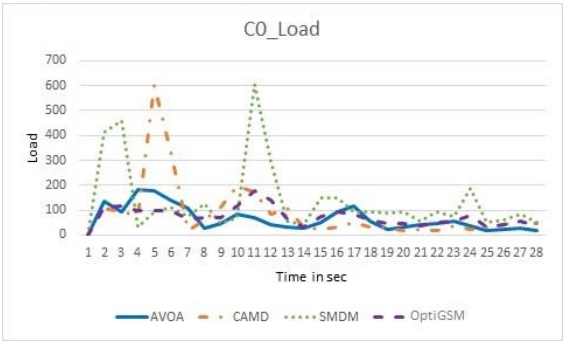


(ب)

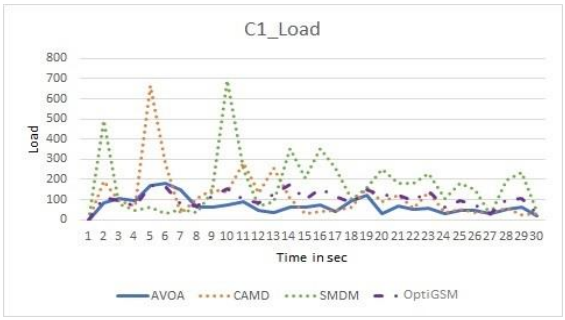
شکل (٤): تغییرات تابع برازش در یک دوره تناوب تخصیص سوئیچ‌ها به کنترل‌کننده‌ها و به دست آوردن بهترین تابع

نتایج به‌دست‌آمده از روش پیشنهادی با روش‌های به‌کاررفته در [۷] (روش OptiGSM) که براساس انحراف معیار بار کنترل‌کننده‌ها و با استفاده از الگوریتم حریصانه مهاجرت را انجام می‌دهد، [۲۰] (روش CAMD) که سوئیچ و کنترل‌کننده با کمترین بار را به‌عنوان سوئیچ مهاجر و مقصد مهاجرت در نظر می‌گیرد و [۱۸] (روش SMDM) که پس از تشخیص عدم توازن بار، از الگوریتم حریصانه برای مهاجرت سوئیچ‌ها استفاده می‌کند، مقایسه شده است. برای ارزیابی از پارامترهای میزان بار کنترل‌کننده‌ها، توان عملیاتی[1]، تأخیر[2]، لرزش[3] و RTT استفاده‌شده است. توان عملیاتی نرخ بسته‌های موفق تحویل داده‌شده به مقصد در واحد زمان است. تأخیر، کل زمان موردنیاز برای ارسال یک پیام PACKET IN بین یک سوئیچ و یک کنترل‌کننده است. از لرزش به‌عنوان تغییر در تأخیر بسته تعریف می‌شود و RTT مدت‌زمان لازم برای ارسال یک بسته داده از مبدأ به مقصد و دریافت پاسخ از مقصد است. با استفاده از ابزارهای D-ITG و IPerf در شبکه ایجاد ترافیک کرده [۲۱ و ۲۲] و روش پیشنهادی مورد ارزیابی قرار گرفت.

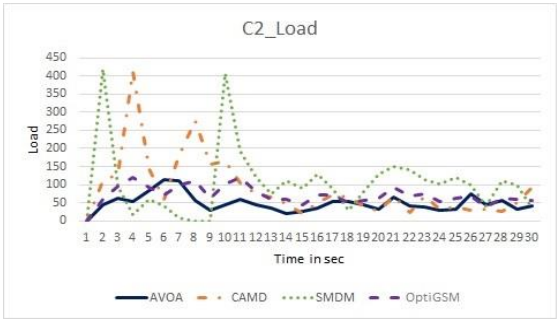
همان‌گونه که در شکل (۵) مشاهده می‌گردد دو روش CAMD و SMDM در مقابل جریان‌های فیل که بار زیادی به کنترل‌کننده تحمیل می‌کنند مقاوم نبوده و میزان نوسانات بالا است، روش OptiGSM مانند روش پیشنهادی این تحقیق با چنین مشکلی مواجه نیست. بعد از کنترل ازدحام تقریباً هر چهار روش مشابه یکدیگر عمل می‌نمایند



(الف)



(ب)

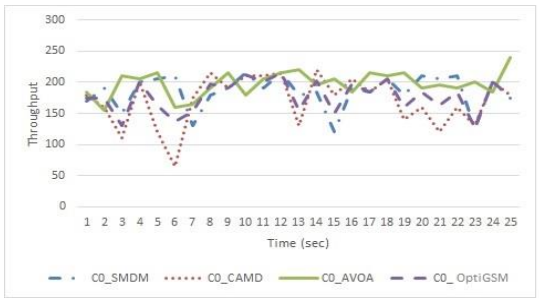


(ج)

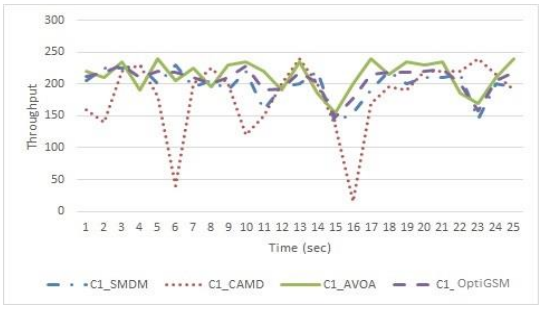
شکل (۵): مقایسه بار کنترل‌کننده‌های C0 و C1 و C2 بین چهار روش موردنظر

شکل (٦) توان عملیاتی بین کنترل‌کننده‌ها را در چهار روش بیان‌شده مقایسه می‌کند و همان‌طور که ملاحظه می‌گردد، روش پیشنهادی نسبت به دو روش پایه ثبات بیشتری در نگهداری توان عملیاتی در بالاترین حد خود را دارد و به‌طور میانگین حدود ۱۵٪ نسبت به سه روش دیگر بهبودیافته است.

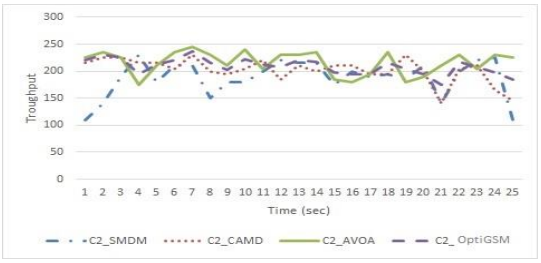
[1] Throughput
[2] Delay
[3] Jitter



(الف)



(ب)

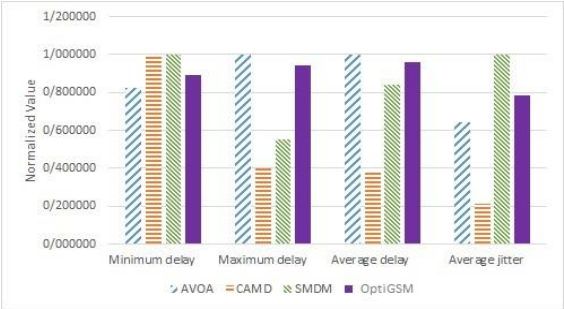


(ج)

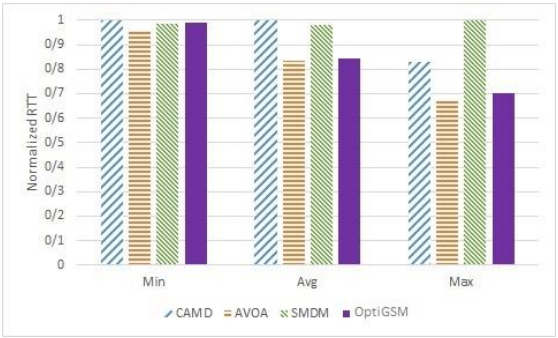
شکل (٦): مقایسه توان عملیاتی کنترل‌کننده‌های C0 و C1 و C2 بین چهار روش موردنظر

با تحلیل پیچیدگی زمانی چهار روش مذکور مشخص گردید روش SMDM دارای پیچیدگی زمانی O(m+n)، روش CAMD دارای O(nlogn+m)، روش OptiGSM دارای O(m*n) و روش پیشنهادی نیز در بدترین حالت دارای پیچیدگی زمانی O(m*n) است که m و n به ترتیب تعداد کنترل‌کننده‌ها و تعداد سوئیچ‌ها می‌باشند. نمودار Jitter و Delay شکل (۷) نیز بیانگر همین مسئله است.

همچنین مقایسه RTT بین روش‌های مذکور در شکل (۸) نشان‌دهنده عملکرد بهینه روش پیشنهادی نسبت به سه روش پایه به علت انتخاب بهینه مهاجرت سوئیچ‌ها است.



شکل (۷): مقایسه delay و Jitter بین چهار روش موردنظر



شکل (۸): مقایسه RTT بین چهار روش موردنظر

۵- نتیجه‌گیری

در شبکه‌های نرم‌افزارمحور با جداسازی لایه کنترل و لایه داده و اطلاع کنترل‌کننده از وضعیت شبکه تحت نظارت خود باعث شده تا مدیریت شبکه و پیشگیری و حل مشکلاتی نظیر ازدحام، عدم توازن بار و تأخیر به کمک توسعه کنترل‌کننده و افزودن الگوریتم‌های موردنیاز بهینه گردد. در این مقاله با استفاده از الگوریتم فرا ابتکاری کرکس آفریقائی و ایجاد شرایطی برای بهبود آن، روشی جهت کنترل ازدحام مبتنی بر مهاجرت سوئیچ ارائه شد. نتایج به‌دست‌آمده از ارزیابی روش پیشنهادی و مقایسه آن باکارهای مشابه نشان‌دهنده کارآیی آن در کنترل ازدحام و بهبود پارامترهای مؤثر در کنترل ازدحام نظیر RTT و توان عملیاتی دارد. دو روش CAMD و SMDM نسبت به جریان‌های فیل عملکرد مناسبی نداشته و روش OptiGSM توان عملیاتی کمتری نسبت به روش پیشنهادی دارد که درنتیجه در الگوریتم پیشنهادی توان عملیاتی حدود ۱۵٪ افزایش و میزان RTT حدود ۲۰٪ کاهش داشته است. البته با توجه به تخصیص‌های متعدد سوئیچ‌ها به کنترل‌کننده‌ها جهت ایجاد کرکس‌های جدید، تعداد مهاجرت‌ها نسبت به روش‌های ساده‌تر مانند CAMD بیشتر است. برای ادامه این

تحقیق، نویسندگان در نظر دارند با ترکیب روش خود و روش مجازی‌سازی شبکه، روش پیشنهادی خود را توسعه داده و به ارزیابی کارآیی الگوریتم پیشنهادی خود در آن بپردازند.

References

[1] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," Jun. 2014, [Online]. Available: http://arxiv.org/abs/1406.0440

[2] Hodaei and S. Babaie, "A Survey on Traffic Management in Software-Defined Networks: Challenges, Effective Approaches, and Potential Measures", *Springer*, May 01, 2021,. doi: 10.1007/s11277-021-08100-3.

[3] T. Semong *et al.*, "Intelligent load balancing techniques in software defined networks: A survey," *Electronics (Switzerland)*, vol. 9, no. 7, pp. 1–24, Jul. 2020, doi: 10.3390/electronics9071091

[4] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A Survey of Networking Applications Applying the Software Defined Networking Concept Based on Machine Learning," *IEEE Access*, vol. 7, pp. 95397–95417, 2019, doi: 10.1109/ACCESS.2019.2928564

[5] O. Adekoya, A. Aneiba, and M. Patwary, "An Improved Switch Migration Decision Algorithm for SDN Load Balancing," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 1602–1613, 2020, doi: 10.1109/OJCOMS.2020.3028971.

[6] V. G. Ourimi and S. Bakhtiari, "A novel approach based on gray wolf evolutionary algorithm for controller load balancing in software defined networks using dynamic switch migration," *Journal of Soft Computing and Information Technology*, Vol 11, pp. 32-48, 2022.

[7] U. Prajapati, C. Chatterjee, and A. Banerjee, "OptiGSM: Greedy-Based Load Balancing with Minimum Switch Migrations in Software-Defined Networks," *IEEE Transactions on Network and Service Management*,vol 21, pp. 2200-2210, 2023, doi: 10.1109/TNSM.2023.

[8] U. Prajapati, B. Chand Chatterjee, and A. Banerjee, "FractionalLB: Controller Load Balancing Using Fractional Switch Migration in Software-Defined Networks," *IEEE Networking Letters*, vol. 6, no. 2, pp. 129–133, Jan. 2024, doi: 10.1109/lnet.2024.3357089.

[9] W. K. Lai, Y. C. Wang, Y. C. Chen, and Z. T. Tsai, "TSSM: Time-Sharing Switch Migration to Balance Loads of Distributed SDN Controllers," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1585–1597, Jun. 2022, doi: 10.1109/TNSM.2022.3146834.

[10] S. Yeo, Y. Naing, T. Kim, and S. Oh, "Achieving balanced load distribution with reinforcement learning-based switch migration in distributed SDN controllers," *Electronics (Switzerland)*, vol. 10, no. 2, pp. 1–16, Jan. 2021, doi: 10.3390/electronics10020162.

[11] H. Babbar, S. Rani, A. K. Bashir, and R. Nawaz, "LBSMT: Load Balancing Switch Migration Algorithm for Cooperative Communication Intelligent Transportation Systems," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1386–1395, Sep. 2022, doi: 10.1109/TGCN.2022.3162237.

[12] Y. Xu *et al.*, "Dynamic Switch Migration in Distributed Software-Defined Networks to Achieve Controller Load Balance," in *IEEE Journal on Selected Areas in Communications*, Institute of Electrical and Electronics Engineers Inc., pp. 515–529, Mar. 2019. doi: 10.1109/JSAC.2019.2894237.

[13] M. A. Jiru, K. Adere, T. G. Krishna, and J. R. Perumalla, "An Improved Switch Migration Method-Based Efficient Load Balancing for Multiple Controllers in Software-Defined Networks," *Journal of Cases on Information Technology*, vol. 25, no. 1, 2023, doi: 10.4018/JCIT.326136.

[14] L. P. A. Sanchez, Y. Shen and M. Guo, "MDQ: A QoS-Congestion Aware Deep Reinforcement Learning Approach for Multi-Path Routing in SDN. *Journal of Network and Computer Applications*, Mar. 2025, 235, 104082

[15] C. Y. Chu, K. Xi, M. Luo, and H. J. Chao, "Congestion-aware single link failure recovery in hybrid SDN networks" in *IEEE Conference on Computer Communications (INFOCOM)*. April. 2015, pp. 1086-1094,

[16] F. Al-Tam and N. Correia, "On load balancing via switch migration in software-defined networking," *IEEE Access*, vol. 7, pp. 95998–

96010, 2019, doi: 10.1109/ACCESS.2019.2929651.
[17] OpenFlow Switch Specification, ON Foundation, 2011. [Online]. Available:https://www.opennetworking.org/
[18] C. Wang, B. Hu, S. Chen, D. Li, and B. Liu, "A Switch Migration-Based Decision-Making Scheme for Balancing Load in SDN," *IEEE Access*, vol. 5, pp. 4537–4544, 2017, doi: 10.1109/ACCESS.2017.2684188.
[19] B. Abdollahzadeh, F. S. Gharehchopogh, and S. Mirjalili, "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems," *Comput Ind Eng*, vol. 158, Aug. 2021, doi: 10.1016/j.cie.2021.107408.
[20] K. S. Sahoo and B. Sahoo, "CAMD: A switch migration based load balancing framework for software defined networks," *IET Networks*, vol. 8, no. 4, pp. 264–271, Jul. 2019, doi: 10.1049/iet-net.2018.5166.
[21] M. T. Islam, N. Islam, and M. Al Refat, "Node to Node Performance Evaluation through RYU SDN Controller," *Wirel Pers Commun*, vol. 112, no. 1, pp. 555–570, May 2020, doi: 10.1007/s11277-020-07060-4.
[22] S. Bhardwaj and S. N. Panda, "Performance Evaluation Using RYU SDN Controller in Software-Defined Networking Environment," *Wirel Pers Commun*, vol. 122, no. 1, pp. 701–723, Jan. 2022, doi: 10.1007/s11277-021-08920-3.

Journal of Applied And Basic Machine Intelligence Research (ABMIR)

Applied and Basic Machine Intelligence Research

Vol. 2, No. 2, Autumn & Winter 2024, pp.95 - 110

Proposing a method for congestion control based on switch migration in software-defined networks using an improved African vulture metaheuristic algorithm

MohammadReza Jenabzadeh[1], Vahid Ayatollahitafti [2*], MohammadReza Mollakhalili Meybodi[3], Mohammadreza Mollahoseini Ardakani[3]

[1] PhD Student, Department of Computer Engineering, Maybod Branch, Islamic Azad University, Maybod , Iran

[2] Assistant Professor, Department of Computer Engineering, Taft Branch, Islamic Azad University, Taft, Iran

[3]Assistant Professor, Department of Computer Engineering, Maybod Branch, Islamic Azad University, Maybod , Iran

Article Information

Original Research Paper

Received:
2025 January 18

Accepted:
2025 April 6

Keywords:
Software-defined networks, Congestion control, Switch migration, African vulture algorithm, Metaheuristic optimization

Corresponding Author*:
vahid.ayat@iau.ac.ir

Abstract

Software-defined networks have emerged as an innovative solution for managing large and complex networks due to their high programmability and flexibility. However, in multi-domain networks that utilize multiple controllers to enhance performance and scalability, serious challenges arise, including congestion management and load balancing among controllers. This paper presents a method based on dynamic and targeted migration of switches to distribute traffic and reduce pressure on controllers, with the aim of addressing the congestion issue. The proposed method employs an improved African Vulture Optimization Algorithm for managing controller load. In this approach, the load levels of switches and controllers are first measured, and if necessary, heavily loaded switches are transferred to controllers with greater capacity. The performance of the proposed method is compared with two similar methods. Simulation results indicate that the proposed method increases network throughput by 15% and improves delay and jitter about 20%. These results indicate the higher efficiency of the proposed method in improving network performance.

: 10.22034/ABMIR.2025.22659.1093

E-ISSN: 2821-2037