فصلنامهٔ علمی- پژوهشی
فناوری اطلاعات و ارتباطات ایران

سال یازدهم، شماره‌های ۳۹و۴۰، بهار و تابستان ۱۳۹۸
صص: ۴۹-۷۲

بهبود مسیریابی جهت کنترل ازدحام در شبکه‌هاب مبتنی بر نرم‌افزار با استفاده از کنترلرهای توزیع‌شده

سعید بختیاری* اردشیر آذرنژاد**
* استادیار دانشگاه علوم انتظامی امین، تهران
**کارشناسی‌ارشد فناوری اطلاعات دانشگاه آزاد اسلامی واحد تهران مرکزی
تاریخ دریافت: ۱۳۹۸ /۰۴/۱۳ تاریخ پذیرش: ۱۳۹۹/۰۳/۲۴

چکیده

شبکه‌های مبتنی بر نرم‌افزار (SDN) برای استفاده در تعیین مسیریابی ترافیک شبکه قابل انعطاف هستند، زیرا سطح داده‌ای و سطح کنترلی را از یکدیگر تفکیک می‌کنند. یکی از چالش‌های بزرگی که پیش روی شبکه‌های مبتنی بر نرم‌افزار قرار گرفته است، انتخاب مکان‌هایی مناسب برای قراردادن و توزیع کنترلرها (کنترل‌کننده‌ها) است؛ به گونه‌ای که بتوان تأخیر بین کنترلرها و سوئیچ‌ها را در شبکه‌های گسترده کاهش داد. در همین راستا اغلب روش‌های ارائه شده بر روی کاهش تأخیر متمرکز بوده‌اند. ولی تأخیر تنها یکی از عواملی است که در کارائی شبکه و کاهش هزینه‌ی کلی بین کنترلرها و سوئیچ‌های مرتبط با آن‌ها نقش دارد. این مقاله به بررسی عوامل بیشتری برای کاهش هزینه بین کنترلرها و سوئیچ‌ها نظیر ترافیک لینک‌های ارتباطی می‌پردازد. به همین منظور یک الگوریتم مبتنی برخوشه‌بندی برای بخش‌بندی شبکه ارائه می‌شود. با بهره گیری از این الگوریتم می‌توان تضمین کرد که هر بخش از شبکه می‌تواند حداکثر هزینه (شامل تأخیر و ترافیک موجود روی لینک‌ها) را در بین کنترلر و سوئیچ‌های مربوط به آن کاهش دهد. در این مقاله، با بکارگیری از Topology Zoo، شبیه‌سازی‌های گسترده‌ای تحت توپولوژی‌های واقعی شبکه انجام شده است. نتایج شبیه‌سازی‌ها نشان می‌دهد در شرایطی که احتمال ازدحام در شبکه بالا می‌رود، الگوریتم پیشنهادی با شناسایی لینک‌های گلوگاه در مسیرهای ارتباطی هر گره با سایر گره‌ها، توانسته به خوبی ازدحام را در شبکه کنترل نماید. لذا، با در نظر گرفتن دو معیار تأخیر و میزان مشغول بودن لینک‌ها، فرآیند قرارگیری و توزیع کنترلرها را در عمل خوشه‌بندی با دقت بالاتری انجام می‌دهد. با این کار، میانگین حداکثر هزینه‌ی انتها به انتها بین هر کنترلر و سوئیچ‌های مربوط به آن به ترتیب در توپولوژی‌های Chinanet کشور چین، Uunet کشور آمریکا، DFN کشور آلمان، و Rediris کشور اسپانیا به اندازه‌ی ۴۱/۴۶۹۴، ۲۹/۲۸۵۳، ۲۱/۳۸۰۵ و ۴۶/۴۸۲۹ درصد کاهش یافته است.

واژه‌های کلیدی: شبکه‌های مبتنی بر نرم‌افزار، کنترلرهای توزیع‌شده، قراردادن کنترلرها، خوشه‌بندی، کنترل ازدحام.

نویسنده عهده‌دار مکاتبات : سعید بختیاری saeid.bakhtiarii@chmail.ir

۱- مقدمه

افزایش فناوری‌های رایانشی جدید اعم از کلان داده‌ها[1]، اینترنت اشیاء[2]، و رایانش ابری[3] باعث شده تا تغییرات قابل ملاحظه‌ای در روش ذخیره‌سازی، گردآوری و انتقال اطلاعات و داده‌ها صورت گیرد. در کنار این روندهای رایانشی جدید، چالش‌های جدیدی اعم از مدیریت و ارتقای شبکه، استفاده‌ی بهینه از منابع، انتقال سریع داده‌ها نیز پدید آمده است. به منظور غلبه بر این چالش ها، از شبکه-های مبتنی بر نرم‌افزار (SDN)[4] به عنوان یک الگوی مطلوب برای شبکه‌های نسل آینده استفاده می‌شود. شبکه-های نرم‌افزار محور در مقایسه با شبکه‌های متعارف از اصل تفکیک سطح کنترلی[5] و سطح داده‌ای[6] بهره می‌برد. این بدین معنا است که سطح کنترلی در این شبکه‌ها به وسیله‌ی یک مجموعه از کنترلرهای[7] اختصاصی شکل گرفته و هر کدام از آن‌ها می‌توانند یک یا چند سوئیچ که وظیفه‌ی هدایت بسته‌های داده‌ای به سمت جلو را بر عهده دارند را مدیریت نمایند. در نتیجه توابع کنترلی و مدیریتی در این شبکه‌ها به گونه‌ای طراحی می‌شوند که خدمات شبکه و برنامه‌های کاربردی، از زیرساختار زیرین آن مجزا باشند.

در شبکه‌های نرم‌افزار محور، کنترلرِ مبتنی بر نرم‌افزار نقش یک واسط هوشمند شبکه را بازی می‌کند و سوئیچ‌های شبکه نیز نقش دستگاه‌های ساده‌ای برای هدایت و فوروارد بسته‌های داده‌ای به سمت جلو را بر عهده دارند که البته می‌توان این فرآیند فوروارد را از طریق واسط‌های بازی مانند OpenFlow برنامه نویسی کرد [۱]. در صورتی که یک بسته وارد یک سوئیچ شده ولی جدول جریان در این سوئیچ هیچ تطابقی با الگوی این بسته نداشته باشد، سوئیچ اقدام به ایجاد یک بسته‌ی درخواست نموده و آن را برای کنترلر مربوطه ارسال می‌کند. کنترلر پس از دریافت این درخواست اقدام به ارسال یک سیاست فوروارد جدید نموده و بر همین اساس سوئیچ اقدام به بروز رسانی جدول جریان خود می‌کند. پس از آن، بسته‌ی داده‌ای بر مبنای جدول جریان که بروز شده است تحویل داده می‌شود.

همان‌طور که در این مدل تشریح شده است، همه‌ی توابع موجود در SDN از طریق مبادله‌ی مکرر پیام ها در بین کنترلر ها و سوئیچ‌ها صورت می‌گیرد. بنابراین، موقعیت کنترلرها نقش قابل ملاحظه‌ای در مبادله‌ی پیام داشته و از این رو بر روی کارائی SDN تأثیر می‌گذارد. مسأله‌ی تعیین موقعیت‌هایی مناسب برای قرار دادن و توزیع کنترلرها را مسأله‌ی گمارش کنترلرها[8] می‌گویند. در این مسأله به دنبال آن هستیم که چطور می‌توان کنترلرها را در یک شبکه‌ی مبتنی بر نرم‌افزار قرار داده و سوئیچ‌های مربوطه را به گونه‌ای به این کنترلرها تخصیص داد تا به هدف مد نظر برسیم. گمارش کنترلرها اهمیت بسیاری در شبکه‌های گسترده[9] دارد که دلیل آن را می‌توان ناشی از توپولوژی نامتعارف و تأخیر بالا در انتشار بسته‌های داده‌ای دانست.

به منظور حل مسأله‌ی گمارش کنترلرها در شبکه‌های نرم‌افزار محور، معیارهای مختلفی ارائه شده است. در بین این معیارها، تأخیر در بین کنترلر و شبکه نقش بسیار مهمی را بازی می‌کند چرا که تأثیر قابل ملاحظه‌ای بر روی کارائی کلی SDN دارد [۲] [۳]. برای مثال در صورتی که یک بسته‌ی داده‌ای هیچ تطابقی با الگوهای تعریف شده در جدول جریان یک سوئیچ نداشته باشد، نیاز است تا این سوئیچ برای یک بازه‌ی زمانی منتظر مانده تا جدول‌های جریان را قبل از پردازش این بسته‌ی داده‌ای دریافت نماید. بدیهی است که یک زمان انتظار طولانی می‌تواند افت کلی بازدهی SDN را به همراه داشته باشد، یعنی برنامه‌های کاربردی[10] که نسبت به زمان حساس هستند با کندی روبرو شده و وظایفی که باید به صورت بلادرنگ صورت گیرند، به هیچ وجه انجام نخواهند شد. به طور معمول، لینک‌های ارتباطی که پهنای باند محدودی دارند و ترافیک بالای شبکه باعث شده تا شبکه دچار ازدحام شده و تأخیر بالایی پیش روی شبکه قرار گیرد. با توجه به ویژگی انحصاری که در SDN وجود دارد، تأخیر ناشی از ازدحام بین کنترلرها و سوئیچ‌ها در شبکه‌های SDN بسیار محدود خواهد بود. چرا که سطح کنترلی از سطح داده‌ای جدا بوده و از این رو پیام‌های کنترلی که بین کنترلرها و سوئیچ‌ها مبادله می‌شود از طریق یک کانال اختصاصی (مد برون باند) منتقل می‌شود [۴] [۵]. علاوه بر این، پیام‌های کنترلی، شامل یک سری

[1] Big Data
[2] Internet of Things (IoT)
[3] Cloud Computing
[4] Software Defined Networking (SDN)
[5] Control Plane
[6] Data Plane
[7] Controller
[8] Controller Placement Problem (CPP)
[9] Wide Area Networks (WAN)
[10] Application

جریان داده‌ای سبک وزن در مقایسه با بارهای داده‌ای موجود در سطح داده‌ای می‌باشند [۶]. در کنار معیار تأخیر، باید بار موجود و درصد مشغول بودن لینک‌های ارتباطی را هم در نظر بگیریم تا از به وجود آمدن نقاط گلوگاه در شبکه و در نتیجه به وجود آمدن ازدحام جلوگیری به عمل آوریم.

در این مقاله به بررسی هزینه انتها به انتها (شامل تأخیر انتها به انتها و درصد مشغول بودن لینک‌ها) در بین کنترلرها و سوئیچ‌ها می‌پردازیم و به دنبال راهکاری برای کاهش هزینه و کنترل ازدحام در شبکه‌های گسترده خواهیم بود. هر کدام از این ها را به صورت مجزا مورد بحث قرار می‌دهیم. در ابتدا باید یک شبکه را به چندین زیر شبکه تقسیم بندی کرده تا بتوان هزینه انتها به انتها در بین کنترلرها و سوئیچ‌ها را کاهش داد. با در نظر گرفتن اهمیت تأخیر و هم‌چنین بار موجود روی لینک‌های ارتباطی، روش پیشنهادی در این مقاله ارائه شده است. باید حتماً در نظر داشته باشیم که تأخیر انتها به انتها تنها یکی از مؤلفه‌های تأثیر گذار بر روی کارائی شبکه می‌باشد. علاوه بر تأخیر عواملی هم‌چون بار موجود روی لینک‌های ارتباطی بین کنترلرها و سوئیچ‌های مربوط به آن‌ها نیز در کارائی کلی شبکه تاثیرگذار می‌باشد.

برای این منظور، یک الگوریتم مبتنی بر خوشه‌بندی برای تقسیم‌بندی شبکه ارائه شده است. در این الگوریتم این اطمینان داده می‌شود که هر بخش از شبکه می‌تواند هزینه حداکثری در بین کنترلر و سوئیچ‌های مربوطه را کاهش دهد. در ادامه، در بخش دوم به بررسی مطالعات مربوطه و تعاریف پایه خواهیم پرداخت. در بخش سوم به بیان ریاضی مسأله‌ی گمارش و توزیع کنترلرها می‌پردازیم. در بخش چهارم، راهکار جدید پیشنهادی را به صورت کامل تشریح خواهیم نمود و به تحلیل کارائی نتایج خواهیم پرداخت.

۲- شبکه‌های مبتنی بر نرم‌افزار

SDN یک معماری جدید برای شبکه‌های کامپیوتری است که طی آن کنترل اطلاعات از خود اطلاعات و انتقال اطلاعات، مجزا می‌شود. روترها و سوئیچ‌های کنونی شبکه‌ها هرچقدر که پیشرفته و قدرتمند باشند عملیات انتقال و کنترل اطلاعات را با هم انجام می‌دهند. در معماری SDN کنترل اطلاعات از سخت‌افزار سوئیچ و روتر مجزا شده و به یک لایه بالاتر رفته و توسط نرم‌افزار انجام می‌شود.

با مجزا شدن سطح کنترلی از سطح داده‌ای در سخت‌افزارها، شرکت‌ها می‌توانند نرم‌افزارها و ابزارهای زیادی برای کنترل اطلاعات نوشته و در نتیجه سرعت، انعطاف‌پذیری[11]، مقیاس‌پذیری[12]، دسترس‌پذیری[13]، و قابلیت اعتماد[14] شبکه را بیشتر کنند. شرکت‌های مختلف می‌توانند برای سخت افزارهایی با برندهای مختلف رابط‌های برنامه‌نویسی کاربردی[15] (API) بنویسند که قابلیت ها و امکانات بیشتری برای شبکه به همراه دارند و مدیریت شبکه را متمرکز و یکپارچه می‌کنند و همچنین امنیت شبکه بالاتر می رود زیرا کاربران با نوشتن نرم‌افزارهایی می‌توانند مدیریت و مانیتورینگ بهتر و بیشتری روی اطلاعات داشته باشند و بر اساس نیازهای شبکه و تهدیداتی که متوجه شبکه آنها است، فایروال ها و سیستم های کشف فیلترینگ را برنامه ریزی و سیاستگذاری کنند. مزیت دیگر، پیکربندی مجدد شبکه و سخت‌افزار بدون نیاز به شرکت سازنده آن سخت‌افزار است. در شبکه‌های کنونی کاربران محدود به استفاده از فناوری و معماری ارائه شده توسط شرکت‌های سازنده سخت افزار هستند و نمی‌توانند خودشان دست به توسعه شبکه بزنند. برنامه‌نویسی رابط شبکه در SDN توسط خود کاربر صورت می‌گیرد و مطابق با نیازهای او می‌تواند بومی‌سازی شود. استاندارد SDN به گونه ای طراحی شده است که فرآیند ارسال اطلاعات در شبکه ها را آسان تر و انعطاف پذیری شبکه‌ها را تحت فضای برنامه‌ریزی شده هوشمند بیشتر می‌کند.

به منظور بهبود مسیریابی در شبکه‌های مبتنی بر نرم‌افزار با استفاده از کنترلرهای توزیع شده، ابتدا برخی از مفاهیم مربوط به آن را بیان می‌نماییم و سپس به بررسی پژوهش‌های اخیر صورت پذیرفته در این حوزه می‌پردازیم.

مفهوم کنترلر در شبکه‌های مبتنی بر نرم‌افزار: شبکه‌های مبتنی بر نرم‌افزار سعی دارد هوشمندی شبکه‌ها را بیشتر کرده و با انتقال بخش کنترل داده‌ها از سوئیچ و روتر سخت‌افزاری به لایه‌های نرم‌افزاریِ مجازی شبکه و بهره‌گیری از یک واحد نرم‌افزاری متمرکز، قابلیت‌هایی مانند برنامه‌ریزی، مقیاس‌پذیری، انعطاف‌پذیری، خودکارسازی، هوشمندی و توسعه نرم‌افزاری شبکه توسط سازمان‌ها را

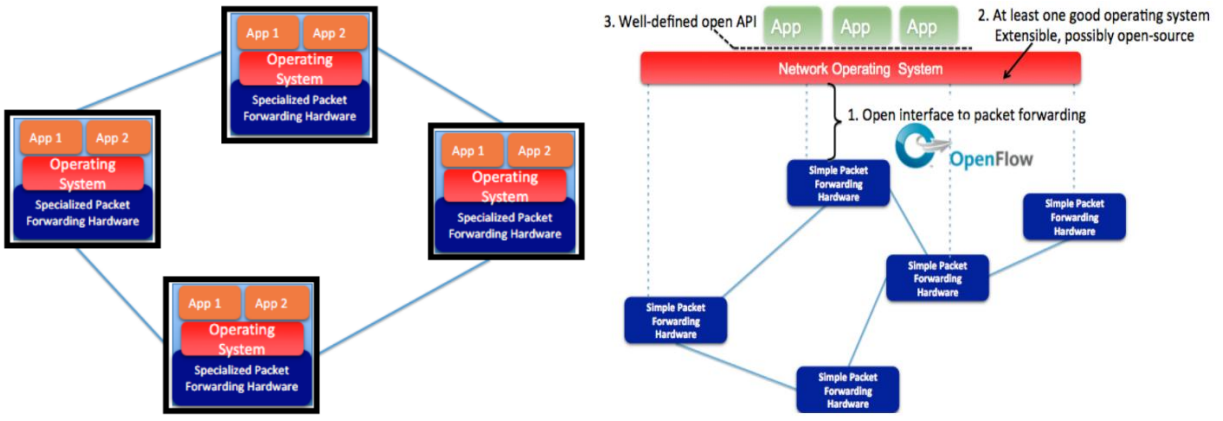
[11] Flexibility
[12] Scalability
[13] Availability
[14] Reliability
[15] Application Programming Interface

فراهم کند [۷]. این واحد متمرکز، کنترلر یا همان کنترل-کننده نام دارد و باید حتماً تأکید کنیم که منظور از متمرکز بودن به معنی منطقاً متمرکز می‌باشد، به این معنی که کنترلرها را می‌توانیم به صورت فیزیکی و آن هم توزیع‌شده، در سطح شبکه قرار دهیم.

معماری شبکه‌های مبتنی بر نرم‌افزار: مدیریت و کنترل شبکه‌های بزرگ همیشه دردسرهای مخصوص به خود را دارد. یکی از آسان‌ترین روش‌های پیشگیری از بروز مشکلات و پیچیدگی‌های مدیریت شبکه‌های بزرگ استفاده از محصولات یک تولیدکننده در تمامی قسمت‌های شبکه مورد نظر است. اتکا به یک تولیدکننده، علاوه بر تحمیل هزینه‌های بیشتر (به خاطر محدودیت‌های مربوط به لایسنس، حق نام، و...) می‌تواند خلاقیت را از سازمان‌ها و شرکت‌ها دور کند.

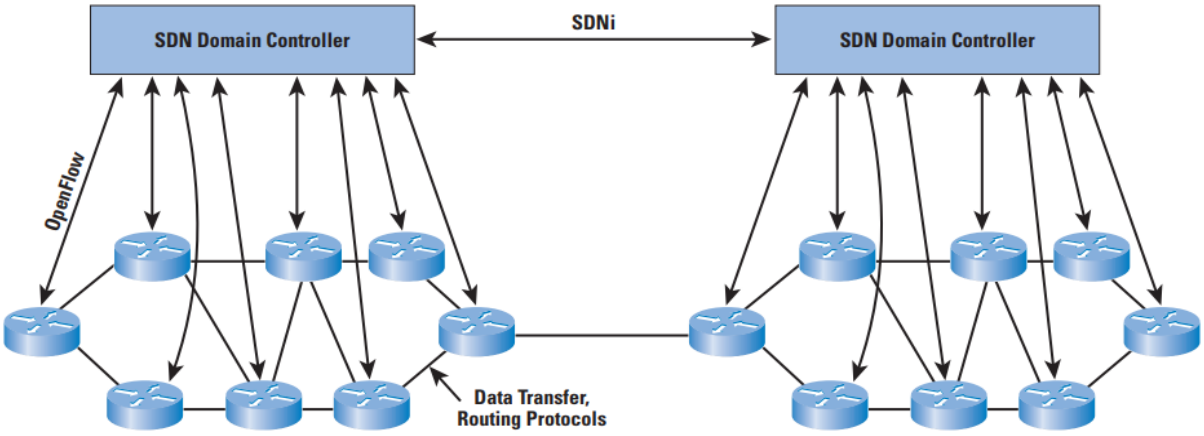
شبکه‌های امروزی شامل کاربرانی است که بوسیله سوئیچ ها و روترها با یکدیگر ارتباط یافته‌اند. این تجهیزات به صورت دستگاه‌هایی عرضه می‌شود که سخت‌افزار، سیستم‌عامل و نرم‌افزار توسط تولیدکننده به صورت یکپارچه در آنها تعبیه شده و تغییر در سیستم‌عامل تقریبا امکان‌پذیر نیست. از این‌رو منطق معماری این تجهیزات را عمودی می‌نامند. در واقع در ساختارهای فعلی، در شبکه‌های بزرگ سوئیچ ها، روترها و سایر تجهیزات شبکه، هم داده و هم اطلاعات کنترلی را در بر دارند که کار بهینه‌سازی ساختار شبکه را بسیار مشکل می‌سازد.

اما تجهیزاتی که برای SDN و استفاده از OpenFlow تولید می‌شوند، از منطق معماری افقی پیروی می‌کنند. در این معماری، دیگر از دستگاه‌هایی یکپارچه خبری نیست و تولیدکننده امکان استفاده از سیستم‌عامل و نرم‌افزار دلخواه مشتری را روی سخت‌افزار تولید شده فراهم می‌کند تا بتوان به‌طور سفارشی از سخت‌افزار بهره جست. در واقع از دیدگاه شبکه می‌توان گفت، قابلیت مدیریت دلخواه چند Control Plane مختلف و استفاده از نرم‌افزارهای کاربردی مجزا روی این تجهیزات فراهم می‌شود. در SDN، داده‌ها و اطلاعات کنترلی تجهیزات شبکه مانند سوئیچ ها و روترها، توسط یک API جدا می‌شوند. در شکل ۱ مقایسه‌ای از معماری عمودی تجهیزات فعلی شبکه در مقابل معماری افقی تجهیزات شبکه SDN نمایش داده شده است.



شکل ۱- معماری عمودی تجهیزات فعلی شبکه در مقابل معماری افقی تجهیزات شبکه SDN [۸]

دامنه‌های موجود در شبکه‌های مبتنی بر نرم‌افزار: در یک شبکه بزرگ، استقرار یک کنترلر واحد برای مدیریت تمام دستگاه‌های شبکه کار مناسبی نیست. یک سناریو محتمل‌تر این است که مدیر شبکه، شبکه را همانطور که در شکل ۲ نشان داده شده است، به تعدادی دامنه SDN که همپوشانی ندارند تقسیم کند.



شکل ۲-تقسیم‌بندی شبکه SDN به دامنه‌های مختلف [۹]

وجود چندین دامنه نیازمند ایجاد کنترلرهای تکی برای برقراری ارتباط با یکدیگر از طریق یک پروتکل استاندارد برای تبادل اطلاعات مسیریابی می‌باشد. برای این منظور IETF اخیراً بر روی توسعه و گسترش پروتکلی به نام SDNi[۱۶] کار می‌کند.

۳- کارهای مرتبط

در این بخش به بررسی تحقیقاتی که بر روی مسأله‌ی گمارش کنترلرها صورت گرفته است می‌پردازیم. تحقیقات موجود را می‌توان بر حسب اهداف زیر به چهار بخش تقسیم کرد [۱۰]: کاهش تأخیر شبکه در بین کنترلرها و سوئیچ‌ها، افزایش قابلیت اطمینان و اعتماد، کاهش هزینه‌ی توسعه و مصرف انرژی و روش چند هدفی.

کاهش تأخیر شبکه در بین کنترلرها و سوئیچ‌ها: آقای هِلر و همکارانش [۱۱] به مطالعه‌ی مسأله‌ی گمارش کنترلرها در SDN پرداختند و خاطر نشان کردند که تأخیر در انتشار (تأخیر میانگین و تأخیر در بدترین حالت)، ملاحظه‌ی اصلی آن‌ها در این مطالعه می‌باشد. این مسأله به صورت یک مسأله‌ی سهولت در مکان‌یابی شبیه‌سازی شده است و از الگوریتم K-center نیز برای حل این مسأله استفاده شده است. آقای یائو و همکارانش [۱۲] نیز هر دوی تأخیر در انتظار و ظرفیت کنترلر را در نظر گرفتند. بنابراین فرآیند گمارش را می‌توان به عنوان یک مسأله‌ی K-center در نظر گرفت [۱۳]. نتایج شبیه‌سازی‌های آن‌ها نشان می‌دهد که راهکار پیشنهادی آن‌ها می‌تواند تعداد کنترلرها و بار کنترلرهای درگیر را کاهش دهد. در [۱۴] به ایجاد یک توازن در بین تأخیر در انتشار و بار ترافیکی پرداخته شده است. نتایج شبیه‌سازی نشان می‌دهد که بار موجود در سطح کنترلی را می‌توان با افزایش جزئی تأخیر متوازن‌سازی کرد. در [۱۵] به بررسی مسأله‌ی گمارش کنترلرها و آن هم از یک دید متفاوت پرداخته شده است. بر مبنای این دیدگاه، بجای کمینه‌سازی مستقیم تأخیر بین کنترلرها و سوئیچ ها، هدف این است که تعداد سوئیچ‌های کنترلی را در یک بازه‌ی زمانی که با تأخیر روبرو هستیم افزایش دهیم.

افزایش قابلیت اطمینان و تاب‌آوری: آقای ژانگ و همکارانش [۱۶] به ارائه‌ی یک الگوریتم مبتنی بر برش کمینه[۱۷] پرداخته‌اند تا بتوانند احتمال قطعی ارتباط در بین کنترلر و سوئیچ را به حداقل سطح ممکن برسانند. آقای هیو و همکارانش [۱۷] نیز به مطالعه‌ی مسأله‌ی گمارش کنترلر برای تضمین قابلیت اطمینان پرداخته‌اند و یک الگوریتم حریصانه را برای گمارش کارآمد و رسیدن به قابلیت اطمینان ارائه نموده‌اند. در [۱۸] هدفی که دنبال شده است این بوده که ارتباط بین دستگاه‌های هدایت‌کننده‌ی بسته‌های داده‌ای و کنترلرها به بالاترین سطح ممکن برسد. در این مطالعه خاطر نشان شده است که تنوع در مسیرها می‌تواند بقای شبکه را در شرایط بروز اشکال و خرابی افزایش دهد. تحقیقات مشابهی نیز صورت گرفته که می‌توان به [۱۹] اشاره کرد. شبیه‌سازی‌هایی نیز در [۲۰] و با هدف افزایش قابلیت اطمینان SDN ارائه شده است. در این مطالعه به گمارش مناسب کنترلرها پرداخته شده است. جدای از تحقیقاتی که به بهبود تاب‌آوری اتصالات شبکه

[16] Interfacing SDN Domain Controllers

[17] Min-cut algorithm

پرداخته‌اند، در [۲۱] روشی ارائه شده است که می‌تواند اشکالات کنترلر در شبکه‌های گسترده‌ی SDN را تحمل نماید.

کاهش هزینه‌ی توسعه و مصرف انرژی: پژوهشگران تحقیق [۲۲] به توسعه‌ی مدلی بهینه پرداخته‌اند و مصرف کلی انرژی را بر اساس شبیه‌سازی‌هایشان کاهش داده‌اند. چالش اصلی در این مدل، پیچیدگی بالای آن می‌باشد چرا که بعضی از محاسبات را نمی‌توان در کمتر از ۳۰ ساعت انجام داد. آن‌ها در این کار شرایط قرارگیری را برای بسط دادن یک شبکه مبتنی بر نرم‌افزار با کمترین هزینه مورد بررسی قرار دادند. بدین صورت که با داشتن طراحی یک شبکه موجود و تعدادی سوئیچ جدید که باید به شبکه قبلی اضافه شوند، مدل بررسی می‌کند که شبکه چگونه مجدداً سازماندهی شود تا هزینه بروزرسانی کمینه شود. از آنجایی که مسأله بسط دادن، کلی شده مسأله برنامه‌ریزی است، این مدل می‌تواند برای برنامه‌ریزی یک شبکه از ابتدا نیز مورد استفاده قرار بگیرد. آن‌ها برای حل این مسأله از یک راه‌حل برنامه‌ریزی خطی استفاده کردند. در [۲۳] راهکاری ارائه شده است که می‌تواند به صورت پویا به اضافه و یا حذف کنترلرها و آن هم بر اساس تغییر بارها بپردازد. یکی از روش‌ها برای به حداقل رساندن مصرف انرژی، روش GreCo می‌باشد که در [۲۴] ارائه شده است. در این روش، لینک‌های غیر ضروری خاموش شده و در عین حال حفظ ارتباط در بین سوئیچ‌ها و کنترلرها تضمین می‌شود. با توجه به شبیه‌سازی‌ها، با روش GreCo می‌توان تا ۵۵ درصد مصرف انرژی را در طی ساعات شلوغی و اوج کاهش داد. روش دیگری در [۲۵] ارائه شده که در آن در ابتدا تعداد کنترلرها مشخص شده و سپس کنترلرهای ضروری فعال می‌شوند تا به این شکل در مصرف انرژی صرفه‌جویی شود. یک مدل دیگر در [۲۶] برای به حداقل رساندن هزینه‌ی بروز رسانی توسعه یافته است و آن هم در زمانی که سوئیچ‌های جدید به شبکه اضافه می‌شوند. با توجه به اینکه این روش در شبکه‌های مبتنی بر نرم‌افزار محدود نمی‌باشد، می‌توان از آن برای ایجاد یک شبکه‌ی جدید و یا بروز رسانی شبکه‌های موجود استفاده نمود.

روش چند هدفی: در [۲۷]، مباحثی در خصوص مسأله‌ی گمارش کنترلرها و آن هم با تمرکز بر تاب‌آوری و اشکال در شبکه‌های هسته‌ای مبتنی بر نرم‌افزار ارائه شده است. یک مجموعه‌ی ابزاری با نام POCO ارائه شده که این امکان را به اپراتورهای شبکه داده تا گمارش را به صورت شبه بهینه انجام دهند. یافته‌های آن‌ها نشان می‌دهد که در اغلب توپولوژی‌ها، بیش از ۲۰ درصد از همه‌ی گره‌ها باید از نوع کنترلر بوده تا ارتباط پیوسته‌ی همه‌ی گره‌ها با یکی از کنترلرها و آن هم در سناریوهای خرابی تضمین شود. این مجموعه ابزار در [۲۸] به صورت توسعه‌یافته تر ارائه شده است به گونه‌ای که در آن از یک روش ابتکاری با دقت کمتر استفاده شده ولی زمان محاسباتی کمتری برای غلبه بر ماهیت پویای شبکه‌های بزرگ مقیاس دارد. ایجاد توازن در بین زمان و میزان صحت نیز به صورت کامل از طریق توپولوژی‌های مختلفی بررسی شده است. پژوهشگران در [۲۹] به ارائه‌ی یک چارچوب کنترل و مدیریت مبتنی بر نرم‌افزار و آن هم برای شبکه‌هایی با ستون فقرات ثابت پرداخته‌اند. الگوریتم‌هایی برای تخصیص مدیران و کنترلرها در یک لایه‌ی کنترلی ارائه شده که هدف آن رسیدن به یک توازن بار و مصرف انرژی می‌باشد. در [۳۰]، یک روش گمارش کنترلر و مبتنی بر چگالی ارائه شده است که مسأله‌ی گمارش کنترلر بر حسب تأخیر، تحمل‌پذیری در برابر اشکال و تعداد کنترلرها، ارائه شده است. ارزیابی‌های کارائی نشان می‌دهد که این روش می‌تواند به کارائی مطلوبی دست پیدا کرده و سرعت اجرا را نیز در عین حال کاهش می‌دهد. آن‌ها برای حل مسأله قرارگیری کنترلرها به چگالی گره‌ها در توپولوژی شبکه توجه کردند. بدین صورت که در ابتدا بر اساس چگالی گره‌ها، شبکه را به تعدادی خوشه‌ی مجزا تقسیم می‌کنند و به دلیل آنکه گره‌های درون هر خوشه اتصال قوی با دیگر گره‌های درون خوشه خود دارند و اتصال آن‌ها با گره‌های دیگر خوشه‌ها ضعیف است، در هر خوشه یک کنترل‌کننده قرار می‌گیرد. ژانگ و همکاران در [۳۱] به تدوین مسأله‌ی گمارش کنترلرها بر روی یک مدل چند هدفی پرداخته‌اند که هدف آن افزایش قابلیت اطمینان در بین کنترلرها و سوئیچ‌ها می‌باشد. نتایج شبیه‌سازی‌ها نشان می‌دهد که روش پیشنهادی می‌تواند کارائی شبکه را بهبود داده و به توان مطلوبی در بین این اهداف دست پیدا کند.

با توجه به بررسی‌ها، بسیاری از روش‌هایی که در بالا مطرح شد تنها تأخیر در انتشار بسته در بین کنترلرها و سوئیچ‌ها را در نظر می‌گیرد تا بتواند تأخیر را به حداقل سطح ممکن

برساند. البته تأخیر در انتشار تنها یکی از عوامل تأثیرگذار بر روی تأخیر سراسری می‌باشد. از جمله عوامل دیگر می‌توان به تأخیر در صف‌بندی کنترلرها اشاره کرد. بنابراین نیاز به تحقیقات گسترده‌ای برای لحاظ کردن همه‌ی مؤلفه‌ها می‌باشد. در مقاله‌ی [۳۲] که به عنوان مقاله‌ی پایه در این پژوهش مورد استفاده قرار گرفته است، همه‌ی تأخیرهای ممکن در بین کنترلرها و سوئیچ‌ها را در نظر گرفته و راهکاری را برای کاهش تأخیرها ارائه نموده است. علاوه بر مقالات و مطالعاتی که در بالا به آن‌ها اشاره شد، می‌توان به موارد زیر نیز در حوزه موضوع پژوهش اشاره کرد:

یک کار دیگر در خصوص قرارگیری کنترلرها توسط بَری و همکارانش انجام شده است [۳۳]. آنها دو الگوریتم مکاشفه-ای برای تأمین کنترلر به صورت پویا پیشنهاد دادند. اهداف آن‌ها شامل کمینه کردن زمان تنظیم جریان، ترافیک کنترلی و تخصیص مجدد سوئیچ به کنترلر می‌باشد.

هم‌چنین به تازگی تحقیقی توسط هیو و همکارانش انجام شده است که به مسأله قرارگیری کنترلرها از دیدگاه ذخیره انرژی نگاه می‌کند [۳۴]. آن‌ها برای کار خود، مسأله قرارگیری ذخیره‌کننده انرژی را توسط یک مسأله خطی عدد صحیح دودویی[18] مدل کردند. در این مدل، مصرف انرژی شبکه‌هایی که برای ترافیک کنترلی استفاده می‌شوند، تحت محدودیت‌های تأخیر مسیرهای کنترلی و بار کنترلرها کمینه می‌شوند. هم‌چنین آن‌ها با توجه به پیچیدگی مدل خطی عدد صحیح، یک الگوریتم مکاشفه‌ای ژنتیک نیز برای پیدا کردن
راه‌حل‌های زیربهینه ارائه دادند.

یک کار متفاوت دیگر نیز توسط ایشیگاکی و همکارانش انجام شده است [۳۵]. آن‌ها در کار خود، راه‌حلی برای مسأله‌ی قرارگیری کنترلرها ارائه داده‌اند که قرارگیری را با توجه به بار روی گره‌های ارتباطی انجام می‌دهد. باید در نظر گرفت که پیام‌های کنترلی بین یک کنترلر و یک سوئیچ که در بخش داده ردوبدل می‌شوند، توسط دیگر سوئیچ‌ها به کنترلر
می‌رسند. معمولاً الگوریتم‌های مسیریابی برای چنین انتقال‌هایی بر اساس کوتاه‌ترین مسیر عمل می‌کنند که باعث می‌شود دائماً برخی گره‌های خاص به عنوان نقاط بازپخش انتخاب شوند. به عبارت دیگر بسیاری از کوتاه‌ترین مسیرها بین سوئیچ‌ها و کنترلر ممکن است شامل گره‌های مشابه باشند. بدین صورت بار زیادی بر روی این گره‌ها به وجود می‌آید که ممکن است باعث خرابی در این نقاط شود. به همین دلیل در این مقاله قرارگیری با توجه به حداکثر بار بر روی این گره‌ها انجام شده است.

تمام موارد فوق تنها بخشی از مطالعات و پژوهش‌هایی بود که در خصوص گمارش و قرارگیری کنترلرها و توزیع آن‌ها در شبکه‌های مبتنی بر نرم‌افزار صورت گرفته است.

۴-روش پیشنهادی

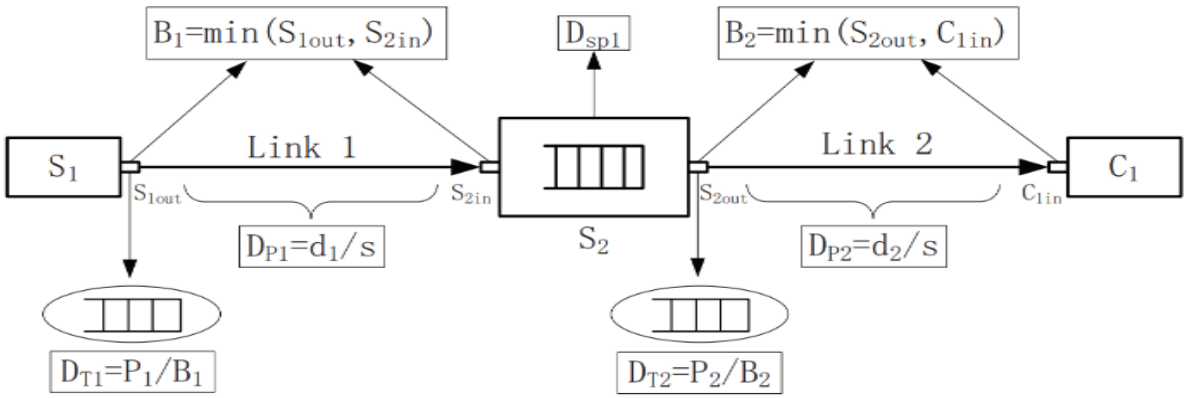
در این بخش با توجه به بررسی‌های صورت گرفته در حوزه‌ی موضوع پژوهش و هم‌چنین با درنظر گرفتن معایب مقاله پایه [۳۲]، روش پیشنهادی خود را که ECNPA[19] نامگذاری کرده‌ایم، ارائه خواهیم کرد و به تشریح آن خواهیم پرداخت. عیب عمده الگوریتم به کار رفته در مقاله پایه در این است که در عمل خوشه‌بندی کنترلرها، تنها معیار تأخیر را در نظر می‌گیرد و توجهی به بار موجود روی لینک‌ها ندارد که این موضوع باعث به وجود آمدن ازدحام در لینک‌ها و ایجاد لینک‌های گلوگاه[20] می‌شود. ولی در روش پیشنهادی، علاوه بر تأخیر بین کنترلرها و سوئیچ‌ها، بار موجود روی لینک‌ها نیز در نظر گرفته شده و به تبع آن با دقت بالاتری عمل خوشه‌بندی را ادامه می‌دهیم. تأخیر و هم‌چنین بار موجود روی لینک‌ها در بین کنترلرها و سوئیچ‌های مربوط به آن‌ها را می‌توان یک ضرورت برای SDN در نظر گرفت چرا که همه‌ی توابع در یک شبکه‌ی مبتنی بر SDN می‌توانند به وسیله‌ی مبادله‌ی مکرر پیام ها در بین کنترلرها و سوئیچ‌ها صورت گیرد. در این بخش به بررسی مؤلفه‌های ممکن مربوط به تأخیر در بین کنترلرها و سوئیچ‌ها می‌پردازیم و در بخش بعد به تدوین مسأله‌ی گمارش کنترلرها خواهیم پرداخت.

در یک شبکه‌ی مبتنی بر نرم‌افزار با تعداد مشخصی از گره‌ها و لینک‌ها، فرض کنید که $S = \{s_1, s_2, \dots, s_m\}$ بیانگر سوئیچ‌های موجود در شبکه باشد و $C = \{c_1, c_2, \dots, c_n\}$ نیز بیانگر کنترلرها باشد. فرض بر آن است که تعداد گام‌های[21] حرکتی مربوط به سوئیچ s_m تا کنترلر c_n به صورت $h(s_m, c_n)$ نمایش داده می‌شود. در شکل ۳، تأخیر انتها به انتها را در شرایطی مشاهده می‌کنید

[18] Binary Integer Program (BIP)

[19] Enhanced Cluster-based Network Partitioning Algorithm
[20] Bottleneck
[21] Hop

که انتقال یک بسته از سوئیچ s_1 به سمت کنترلر c_1 صورت می‌گیرد.



شکل ۳: تأخیر انتها به انتها در انتقال بسته در SDN [۳۲]

تأخیر انتها به انتها مربوط به انتقال یک بسته‌ی داده‌ای از سوئیچ s_1 به کنترلر c_1، متشکل از سه مؤلفه می‌باشد: تأخیر در انتقال بسته (D_{Ti})، تأخیر در انتشار بسته (D_{Pi}) و تأخیر در پردازش سوئیچ (D_{SP_i}). تأخیر در انتقال به معنای زمان مورد نیاز برای انتقال بیت‌های یک بسته‌ی داده‌ای به سمت یک لینک می‌باشد که آن را به صورت $D_{Ti} = P_i/B_i$ نمایش می‌دهند و در آن، P_i بیانگر تعداد بیت‌های داده‌ای یک بسته در لینک i بوده و B_i نیز بیانگر پهنای باند لینک می‌باشد. در نظر داشته باشید که پهنای باند یک لینک خاص را می‌توان به وسیله‌ی حداقل نسبت انتقال واسطهای شبکه مشخص کرد، یعنی $B_1 = \min(S_{1out}, S_{2in})$ که در شکل ۳ نمایش داده شده است. تأخیر در انتشار بسته‌ی داده‌ای به معنای زمان مورد نیاز برای رسیدن یک بسته به مقصد می‌باشد. تأخیر در انتشار به صورت $D_{pi} = \frac{d_i}{s}$ مشخص شده که در آن، d_i بیانگر فاصله‌ی لینک i بوده و s بیانگر سرعت سیگنال رسانه‌ای است که برای انتقال داده‌ها بکار گرفته می‌شود. تأخیر در پردازش سوئیچ نیز به صورت D_{spi} نمایش داده می‌شود که تحت تأثیر بار سوئیچ i قرار می‌گیرد. بنابراین تأخیر انتها به انتها برای بسته‌ای که از سوئیچ S_m به سمت کنترلر c_n حرکت می‌کند در معادله ۱ نمایش داده شده است:

$$D_{e2e}(s_m, c_n) = \sum_{i=1}^{h(s_m,c_n)} \left(\frac{P_i}{B_i} + \frac{d_i}{S} + D_{SP}(i-1)\right) \quad (۱)$$

در نظر داشته باشید که سه مؤلفه از تأخیر انتها به انتها، وزن یکسانی در شبکه‌های مختلف ندارند. برای مثال در یک شبکه محلی[۲۲] (LAN) که پهنای باند کم سرعتی داشته و محدوده‌ی انتقال نیز محدود می‌باشد، تأخیر در انتقال بسته‌های داده‌ای (D_T)، غالب بوده در حالی که تأخیر در انتشار (D_P) را می‌توان نادیده گرفت. در مقابل، در یک شبکه‌ای که مجهز به سوئیچ‌های ۱۰ یا ۱۰۰ گیگابیت بر ثانیه می‌باشد [۳۶] [۳۷] و فاصله‌ی صدها مایلی دارد، تأخیر در انتقال بسته را می‌توان نادیده گرفت در حالی که تأخیر در انتشار بسته‌های داده‌ای یک تأخیر مهم می‌باشد. تأخیر در پردازش سوئیچ (D_{SP}) نیز به وسیله‌ی کارائی سوئیچ‌ها مشخص می‌شود. سوئیچ‌هایی با کارائی بالا و سوئیچ‌هایی با سرعت سیمی را همراه با تأخیر پایین و نوسان در شبکه‌های ستون فقرات مشاهده می‌کنیم. اخیراً دستاوردهای مهمی بر روی سوئیچ‌های SDN صورت گرفته است. برای مثال همان‌طور که در [۳۸] مطرح شده است، Corsa DP2100 [۳۷] می‌تواند به توان عملیاتی ۱۰۰ گیگابیت بر ثانیه دست پیدا کند و این بدین معنا است که تأخیر در پردازش سوئیچ (D_{sp}) را می‌توان در صورتی که از این نوع سوئیچ‌ها در یک

[22] Local Area Network (LAN)

شبکه‌ی ستون فقرات مبتنی بر نرم‌افزار استفاده شود، چشم‌پوشی کرد.

در ادامه به تدوین مسأله‌ی تقسیم‌بندی شبکه در SDN می‌پردازیم. برای یک شبکه‌ی مبتنی بر نرم‌افزار با تعداد مشخصی از گره‌ها و لینک ها، توپولوژی فیزیکی شبکه را به صورت یک گراف $G = (V, E)$ نمایش می‌دهیم که در آن، V بیانگر یک مجموعه از گره‌ها بوده و هر دوی سوئیچ‌ها و کنترلرها را در بر دارد. این بر مبنای این فرضیه بوده که کنترلرها در مکان‌هایی قرار دارند که سوئیچ‌ها نیز در آن مکان قرار دارند به گونه‌ای که کنترلرها و سوئیچ‌ها را می‌توان به هم متصل نمود [۱۱] [۱۲] [۲۸].

E بیانگر مجموعه‌ای از لینک‌های فیزیکی در بین این گره‌ها می‌باشد. فرض بر آن است که تعداد زیرگراف‌ها به صورت K مشخص می‌شود و بیانگر تعداد زیرشبکه‌های موجود در شبکه باشد. در زمانی که SDN در نظر گرفته می‌شود، پارتیشن شبکه را می‌توان به صورت $SDN_i(V_i, E_i)$ تعریف کرد:

$$\bigcup_{i=1}^{k} V_i = V; \bigcup_{i=1}^{K} E_i = E \quad (۲)$$

$$SDN_i \cap SDN_j = \emptyset, \quad \forall i \neq j, i, j \in k \quad (۳)$$

$$similarity(SDN_i) = TRUE, \quad \forall i \in K \quad (۴)$$

$$Similarity\left(SDN_i \cap SDN_j\right) = FALSE \quad \forall i \neq j, i, j \in k \quad (۵)$$

$$SDN_i \text{ is a connected region} \quad \forall i \in k \quad (۶)$$

معادله‌ی ۲ بکار گرفته شده تا نمایش داده شود که کل زیرشبکه‌ها باید همه‌ی اِلمان‌ها را پوشش دهند: گره‌ها و لینک‌ها. در معادله‌ی ۳، $\emptyset$ به معنای یک مجموعه‌ی خالی بوده به این معنا که یک گره یا لینک تنها می‌تواند به یک زیرشبکه تخصیص داده شود و معادله‌ی ۴ نیز مبین این بوده که المان‌های موجود در یک زیرشبکه داری تشابه یکسانی می‌باشند. در معادله‌ی ۵ نشان داده شده است که اِلمان‌هایی که به زیرشبکه‌های مختلف تخصیص داده می‌شوند، دارای کمترین تشابه با یکدیگر هستند. منظور ما از تشابه، همان هزینه (تأخیر و بار) می‌باشد. به طور خاص، انتظار می‌رود که شبکه به گونه‌ای تقسیم شود که گره‌های موجود در یک خوشه دارای هزینه کمتری باشند. معادله‌ی ۶ نشان می‌دهد که همه‌ی رئوس موجود در زیرشبکه‌ها به وسیله‌ی لینک به هم متصل می‌باشند. معادلاتی که در بالا به آن‌ها اشاره شد، همگی در مقاله پایه مورد استفاده قرار گرفته‌اند. حال هدفی که در این مقاله دنبال می‌کنیم این است که کنترلرها را در یک شبکه‌ی گسترده‌ی مبتنی بر نرم‌افزار به گونه‌ای قرار دهیم تا بتوان حداکثر هزینه بین کنترلرها و سوئیچ‌ها را به حداقل سطح ممکن رساند. در روش پیشنهادی اگر $Load_i$ را میزان مشغول بودن لینک i ام در نظر بگیریم، میزان مشغول بودن مسیر بین سوئیچ s_m و کنترلر c_n از رابطه‌ی ۷ محاسبه می‌شود:

$$Load(s_m, c_n) = \max\{Load_i\} \quad (\forall i \in h(s_m, c_n)) \quad (۷)$$

هزینه را به صورت معادله‌ی ۸ نمایش می‌دهیم که شامل تأخیر انتها به انتها و درصد بار موجود یا همان میزان مشغول بودن لینک گلوگاه در مسیر بین کنترلر تا سوئیچ می‌باشد:

$$Cost = D_{e2e}(s_m, c_n) \times Load(s_m, c_n) \quad (۸)$$

بنابراین تابع هدف[23] را به صورت رابطه ۹ بیان می‌کنیم:

$$min\{max\{Cost\}\} \quad , \quad \forall s_m, c_n \in SDN_i \ , (i \in k) \quad (۹)$$

۴-۱- تشریح نحوه تقسیم‌بندی شبکه به k زیرشبکه با استفاده از الگوریتم پیشنهادی

در این بخش به ارائه‌ی یک الگوریتم بخش‌بندی شبکه بر مبنای خوشه‌بندی می‌پردازیم و چگونگی بخش‌بندی یک شبکه به زیرشبکه‌ها را با هدف کاهش ماکزیمم هزینه‌ی انتها به انتها ارائه می‌دهیم. مسأله‌ی بخش‌بندی شبکه مشابه با مسأله‌ی خوشه‌بندی می‌باشد و راهکارهای آن را می‌توان از الگوریتم‌های خوشه‌بندی الهام گرفت. البته الگوریتم‌های استاندارد خوشه‌بندی، مانند K-means و K-center را نمی‌توان به صورت مستقیم برای بخش‌بندی یک شبکه به زیرشبکه‌ها بکار گرفت که دلیل آن را می‌توان ناشی از موارد زیر دانست: اول اینکه انتخاب تصادفی مراکز اولیه نمی‌تواند این تضمین را بدهد که هر پارتیشن می‌تواند هزینه حداکثری در بین مرکز و گره‌های مربوط به آن را در زیرشبکه‌ها کاهش دهد. دوم اینکه فاصله‌ی اقلیدسی را

[23] Objective Function

نمی‌توان برای محاسبه‌ی فاصله‌ی بین دو گره جغرافیایی بکار گرفت. به منظور غلبه بر این معایب، از الگوریتم پیشنهادی برای حل مسأله‌ی بخش‌بندی شبکه استفاده می‌کنیم. در الگوریتم پیشنهادی، به محاسبه‌ی هزینه بین هر دو گره پرداخته می‌شود. مسیری با کوتاه‌ترین فاصله در الگوریتم پیشنهادی بکار گرفته شده تا تأخیر انتها به انتها محاسبه شود. به طور خاص، در یک توپولوژی شبکه با لینک‌ها و گره‌های مختلف، ماتریس مجاورت این توپولوژی در ابتدا محاسبه می‌شود. مختصات این گره‌ها را از روی نقشه‌ی گوگل به دست آوردیم و فاصله‌ی این لینک‌ها را نیز با استفاده از فرمول "haversine" محاسبه کردیم [۳۹] [۴۰]. هم‌چنین کوتاه‌ترین مسیر بین هر دو گره را با استفاده از الگوریتم دایجسترا محاسبه خواهیم کرد [۴۱]. مراحل الگوریتم پیشنهادی در زیر مشاهده می‌شود:

الگوریتم پیشنهادی برای خوشه‌بندی و توزیع کنترلرها

ورودی‌ها:

۱- توپولوژی شبکه که به صورت $G = (V, E)$ نمایش داده می‌شود و از داخل یک فایل XML وارد برنامه می‌شود که در آن V نشان‌دهنده‌ی گره‌ها و E نشان‌دهنده‌ی یال‌ها یا همان لینک‌های ارتباطی می‌باشد.

۲- تعداد زیر شبکه‌ها. (K)

شروع

گام اول: محاسبه‌ی تأخیر انتها به انتها بین هر دو گره دلخواه. $\forall a, b \in V \; l_{e2e}(a, b)$.

گام دوم: محاسبه‌ی میزان مشغول بودن لینک گلوگاه در مسیر بین هر دو گره دلخواه. $\forall a, b \in V \; load(a, b)$.

گام سوم: محاسبه‌ی اولین مرکز شبکه. گره‌ای که دارای کمترین هزینه (تأخیر و بار) نسبت به سایر گره‌ها باشد به عنوان اولین مرکز انتخاب می‌شود.

گام چهارم: پیدا کردن مرکز بعدی شبکه. گره‌ای که دارای بیشترین هزینه (تأخیر و بار) نسبت به مراکز قبلی می‌باشد، به عنوان مرکز بعدی انتخاب می‌شود.

گام پنجم: توزیع بردار $v(v \in V)$ بر روی یکی از خوشه‌ها و آن هم با استفاده از رابطه‌ی زیر:

$$v \in Cluster_i \; , \quad if \; Cost(v, c_i) < Cost(v, c_j) \; , \forall i, j \in \{1,2, \dots, k\}$$

گام ششم: بروز رسانی مراکز $C' = \{c'_1, c'_2, \dots, c'_k\}$ به گونه‌ای که جمع هزینه‌ی مربوط به همه‌ی گره‌ها در خوشه i نسبت به مرکز c'_i به حداقل سطح ممکن برسد.

$$c'_i = v_m \; , \quad if \; Cost(v_m, v) = minimum$$
$$\forall \; m \in size(cluster_i), v \in Cluster_i \; , \; i = \{1,2, \dots, k\}$$

گام هفتم: مراحل ۴، ۵ و ۶ را تکرار می‌کنیم تا اینکه شبکه به k زیر شبکه تقسیم شود.

پایان

خروجی‌ها:

۱- مراکز K خوشه.

۲- تخصیص سوئیچ‌ها به هر یک از خوشه‌ها.

شبه‌کد مربوط به محاسبه فاصله بین هر دو گره از دو سر یک لینک در شبکه WAN با استفاده از فرمول Haversine که در بالا به آن اشاره شد، به صورت زیر می‌باشد:

شبه‌کد محاسبه فاصله بین هر دو گره با فرمول Haversine

```
1)Begin
2)Function dist = CalcDistance(node1,node2)
3)         lat1 = node1(1);
4)         lng1 = node1(2);
5)         lat2 = node2(1);
6)         lng2 = node2(2);
7)         earthRadius = 3958.75;   % In
Miles(6370.99056 In Kilometers)
8)         dLat = toRadians(lat2 - lat1);
9)         dLng = toRadians(lng2 - lng1);
10)         a = sin(dLat / 2) * sin(dLat / 2) +
11)         cos(toRadians(lat1)) *
cos(toRadians(lat2)) *
12)         sin(dLng / 2) * sin(dLng / 2);
13)         c = 2 * atan2(sqrt(a), sqrt(1 - a));
14)         dist = earthRadius * c;
15)         meterConversion = 1/0.000621371192;
16)         dist=dist * meterConversion;
17)         Function r=toRadians(d)
18)             r=d/180*pi;
19)         End function
20)End function
21)End
```

شبه‌کد مربوط به محاسبه تأخیر انتها به انتها بین هر دو گره از دو سر یک لینک در شبکه WAN با توجه به معادله‌ی ۱ که در بالا به آن اشاره شد، به صورت زیر می‌باشد:

شبه‌کد محاسبه تأخیر انتها به انتها با معادله‌ی ۱

```
1)Begin
2)Function  [Delay]  =  CalcLatency(Distance)     %
Distance is in Meters
3)     SpeedOfLight=299792458;  % Meters Per Second
4)         WANFiberSpeed=(SpeedOfLight*65)/100;     %
Meters Per Second
5)     Delay=(Distance/WANFiberSpeed)*1000;  % Delay
in Millisecond
6)End function
7)End
```

شبه‌کد مربوط به محاسبه میزان مشغول بودن لینک گلوگاه در مسیر بین هر دو گره دلخواه در شبکه با توجه به معادله‌ی ۷ که در بالا به آن اشاره شد، به صورت زیر می‌باشد:

شبه‌کد محاسبه میزان مشغول بودن لینک گلوگاه با معادله‌ی ۷

```
1)Begin
% Find the bottleneck link load
2)For each node with indices i,j in Topology
3)tmp=[];
4)  For k=FindLinks(path,links)
5)    tmp=[tmp;loads(loads(:,1)==k,:)];
6)  End for
7)tmp=tmp(tmp(:,2)==max(tmp(:,2)),2);
8)AllNodesBottleNeckLinkLoad(i,j)=tmp(1);
9)AllNodesBottleNeckLinkLoad(j,i)=tmp(1);
10)End for
11)End
```

بطور خلاصه، اولین مرحله در الگوریتم پیشنهادی به این صورت است که می‌تواند مرکز واقعی شبکه (محور) را به راحتی پیدا کند. به طور خاص، الگوریتم پیشنهادی اقدام به محاسبه‌ی هزینه مربوط به هر گره نسبت به سایر گره‌ها نموده و گره ای را که دارای کمترین هزینه تا سایر گره‌ها می‌باشد را به عنوان محور انتخاب می‌کند. در گام بعد، الگوریتم پیشنهادی اقدام به پیدا کردن مرکز دوم شبکه می‌کند. این مرکز دوم، به عنوان گره ای انتخاب می‌شود که بیشترین هزینه را دارد. در گام بعد، الگوریتم پیشنهادی نقش محور (c_1) و مرکز دوم (c_2) را به عنوان دو مرکز اولیه انتخاب کرده و گره‌های مربوطه را به این مراکز تخصیص می‌دهد. به طور خاص به ازای هر گره‌ی n_i، هزینه آن نسبت به مراکز محاسبه شده تا دو مقدار $Cost_1 = Cost(n_i, c_1)\,, Cost_2 = Cost(n_i, c_2)$ به دست آید. این دو مقدار با هم مقایسه شده و گره به مرکزی تخصیص داده می‌شود که به آن هزینه‌ی کمتری دارد. برای مثال در صورتی که $Cost_1 < Cost_2$ باشد، گره ی n_i به c_1 تخصیص داده می‌شود. زمانی که گره به یک مرکز دیگر تخصیص داده شد، محور این خوشه بر مبنای حداقل هزینه‌ای که در گام سوم مشخص شده است محاسبه می‌شود. این فرآیند تا زمانی ادامه پیدا می‌کند که شبکه به k زیرشبکه تقسیم شود. قابل ذکر است که بر خلاف الگوریتم‌های متعارف خوشه‌بندی، که به صورت تصادفی اقدام به انتخاب یک باره‌ی همه k مرکز نموده و سپس همه‌ی آن‌ها را در طی تکرارهایی بهینه می‌سازند، الگوریتم پیشنهادی در ابتدا اقدام به تقسیم کل شبکه به تعدادی زیرشبکه نموده تا به k زیرشبکه دست پیدا کند. در طول هر بخش، می‌توان هزینه را در بین مراکز و گره‌های مربوط به آن‌ها کاهش داده و از این رو هزینه حداکثری در سطح قابل ملاحظه‌ای در مقایسه با الگوریتم‌های متعارف خوشه‌بندی مانندK-center و K-means کاهش پیدا می‌کند.

۵- ارزیابی روش پیشنهادی

در این بخش به ارزیابی روش پیشنهادی می‌پردازیم و آن را با راهکارهای موجود و آن هم تحت توپولوژی‌های واقعی که از پروژه‌ی Topology Zoo[۲۴] به دست آمده است [۴۲] مقایسه می‌نماییم. الگوریتم پیشنهادی در نرم‌افزار متلب به عنوان یک محیط رایانش عددی قوی پیاده سازی شده است. این نرم‌افزار به صورت گسترده در حوزه‌ی تحقیقات گمارش کنترلر SDN بکار گرفته می‌شود. شبیه‌سازی‌ها شامل گام‌های زیر می‌باشد:

گام اول: در اولین مرحله با استفاده از تابعی با نام ImportGraphML دیتاست‌های مربوطه را که در قالب فایل‌های XML می‌باشند و شامل اطلاعاتی مانند موقعیت جغرافیایی[۲۵] گره‌های شبکه، مشخصات لینک‌ها، توپولوژی شبکه، و ... می‌باشند، وارد برنامه می‌کنیم. موقعیت جغرافیایی گره‌های شبکه شامل مشخصات طول جغرافیایی[۲۶] و عرض جغرافیایی[۲۷] می‌باشد که در جدولی به نام latlong ذخیره می‌شوند. اطلاعات مربوط به لینک‌ها در جدول links و اطلاعات مربوط به توپولوژی شبکه در جدول topology ذخیره می‌شود. تمامی عملیات مربوط به گام اول با اجرای تابع main انجام می‌شود.

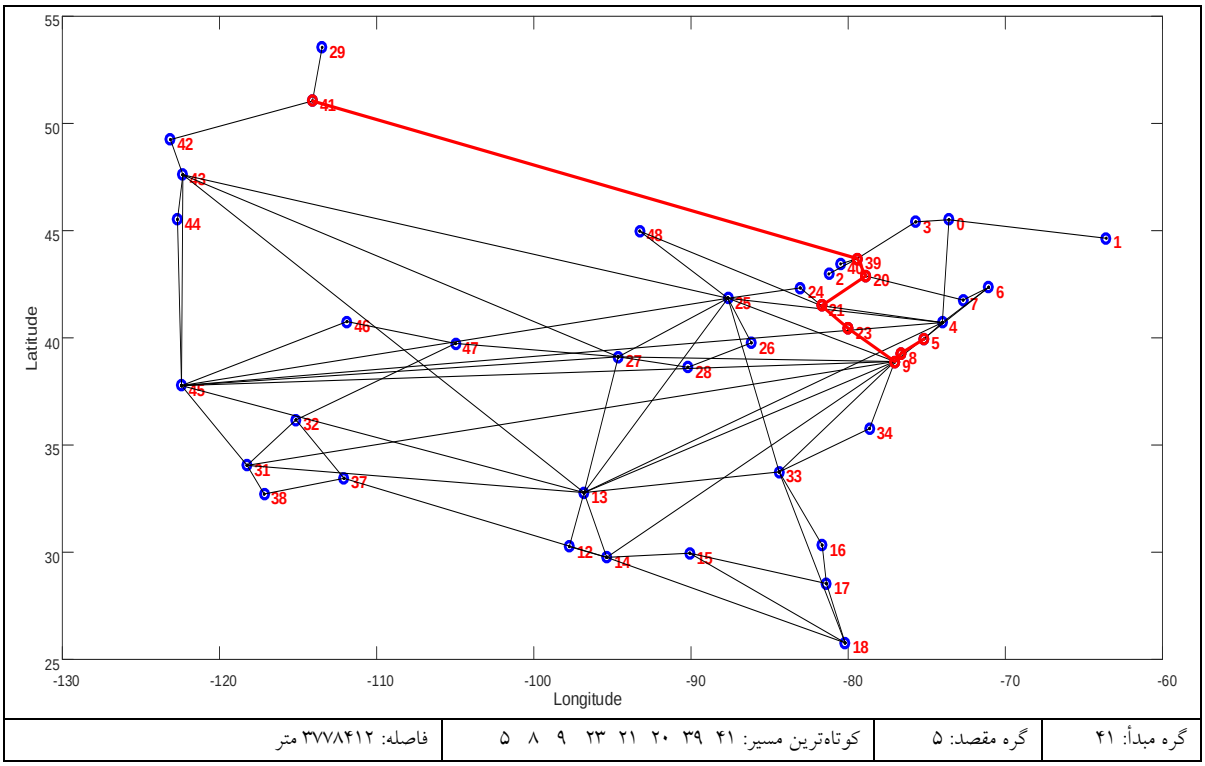
گام دوم: فاصله‌ی بین هر دو گره از دو سر یک لینک با استفاده از فرمول Haversine که در تابع CalcDistance پیاده‌سازی شده است، محاسبه می‌شود.

گام سوم: کوتاه‌ترین مسیر بین هر دو گره دلخواه در شبکه با استفاده از الگوریتم Dijkstra که یکی از معروف‌ترین و محبوب‌ترین الگوریتم‌های مسیریابی می‌باشد، محاسبه می‌شود. این گام توسط تابع Dijkstra انجام شده و نتایج در جدول AllNodesDistances ذخیره می‌شود. در شکل ۴ نمونه‌ای از مسیریابی بین هر دو گره دلخواه از شبکه با استفاده از الگوریتم Dijkstra نمایش داده شده است. لازم به ذکر است که هم در روش پیشنهادی و هم در روش

[۲۴] پروژه Topology Zoo شامل صدها توپولوژی شبکه بوده که توسط سرویس‌دهندگان شبکه تدارک دیده شده است.

[25] Geographical Location
[26] Longitude
[27] Latitude

به کار رفته در مقاله پایه، از همین الگوریتم مسیریابی استفاده شده است.



گره مبدأ: ۴۱	گره مقصد: ۵	کوتاه‌ترین مسیر: ۴۱ ۳۹ ۲۰ ۲۱ ۲۳ ۹ ۸ ۵	فاصله: ۳۷۷۸۴۱۲ متر

شکل ۴- نمونه‌ای از مسیریابی بین گره‌های شبکه‌ی Uunet با استفاده از الگوریتم Dijkstra

گام چهارم: در این مرحله با توجه به نتایج به دست‌آمده از مرحله قبل و با استفاده از تابع CalcLatency تاخیر انتها به انتها بین هر دو گره دلخواه محاسبه می‌شود. توجه به این نکته حائز اهمیت است که با توجه به این که گره‌های ما در سطح شبکه WAN توزیع شده‌اند و لینک‌های مورد استفاده فیبر نوری می‌باشد و با در نظر گرفتن این که سرعت انتقال سیگنال در فناوری فیبر نوری ۶۵ درصد سرعت نور است، محاسبات لازم در تابع CalcLatency انجام شده است. هم‌چنین نتایج به دست آمده از این مرحله در جدول AllNodesLatencies ذخیره شده است.

گام پنجم: اطلاعات مربوط به بار موجود روی هر لینک و درصد مشغول بودن آن توسط یک کنترلر مرکزی در جدولی به نام loads ذخیره می‌شود. در این مرحله با استفاده از تابعی به نام FindLinks ، لینک گلوگاه بین هر دو گره دلخواه به دست آمده و میزان مشغول بودن آن در جدولی به نام AllNodesBottleNeckLinkLoad ذخیره می‌شود. با شناسایی لینک‌های گلوگاه توانسته‌ایم ازدحام را در شبکه کنترل کنیم و با دقت بالاتری کنترلرها را توزیع کنیم و عمل خوشه‌بندی را انجام دهیم. لازم به توضیح است که برای پیاده‌سازی این مرحله، درصد مشغول بودن هر لینک را به صورت یک عدد تصادفی بین ۰ تا ۱۰۰ در نظر گرفته‌ایم که نشان‌دهنده وضعیت شبکه در آن لحظه مشخص می‌باشد.

گام ششم: با توجه به تابع هدفی که در بخش قبل معرفی کردیم، هزینه مربوط بین هر دو گره دلخواه محاسبه شده و در جدولی به نام AllNodesCosts ذخیره می‌شود.

گام هفتم: نتایج حاصل از اجرای الگوریتم پیشنهادی با الگوریتم K-means و CNPA مقایسه شده و نشان می‌دهیم که الگوریتم پیشنهادی بهتر عمل کرده است.

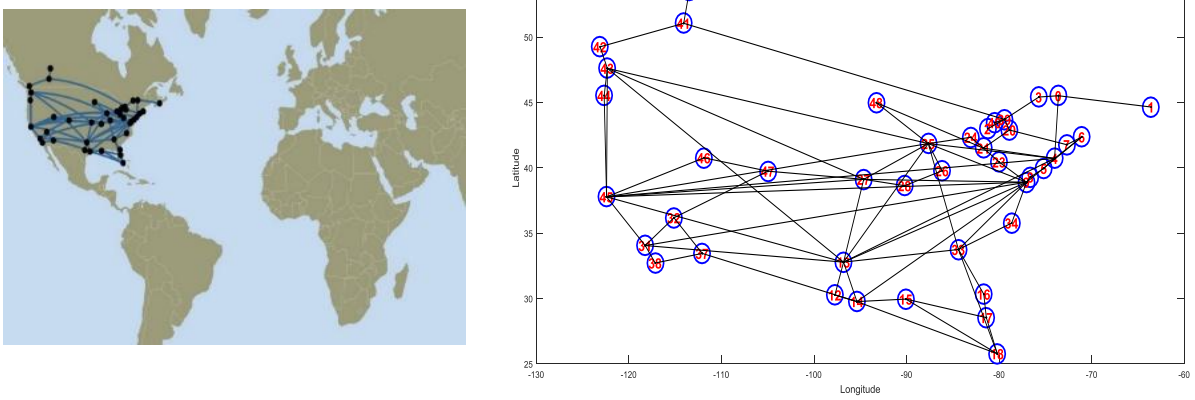
۵-۱- نتایج پیاده‌سازی

در این قسمت نتایج حاصل از پیاده‌سازی را با اجرای ۱۰۰ مرتبه از هر سه الگوریتم، در چند نمونه از توپولوژی‌های مربوط به پروژه Topology Zoo با هم مقایسه می‌کنیم. ضمناً با توجه به این که در مقاله پایه، توپولوژی Chinanet مربوط به کشور چین مورد بررسی قرار گرفته است و هم‌چنین جهت نشان دادن هر چه بهتر عملکرد الگوریتم پیشنهادی در توپولوژی‌های پیچیده‌تر و متفاوت‌تر در سایر

کشورهای پیشرفته جهان مانند آمریکا، آلمان، و... به انتخاب دیتاست‌های مربوطه پرداخته و نتایج به دست آمده را ارزیابی می‌کنیم.

۵-۱-۱- ارزیابی نتایج به دست آمده در توپولوژی Uunet مربوط به کشور آمریکا

در شکل ۵ نمایی از توپولوژی واقعی شبکه‌ی Uunet را در مقابل توپولوژی پیاده‌سازی شده آن در نرم-افزار متلب مشاهده می‌کنیم:

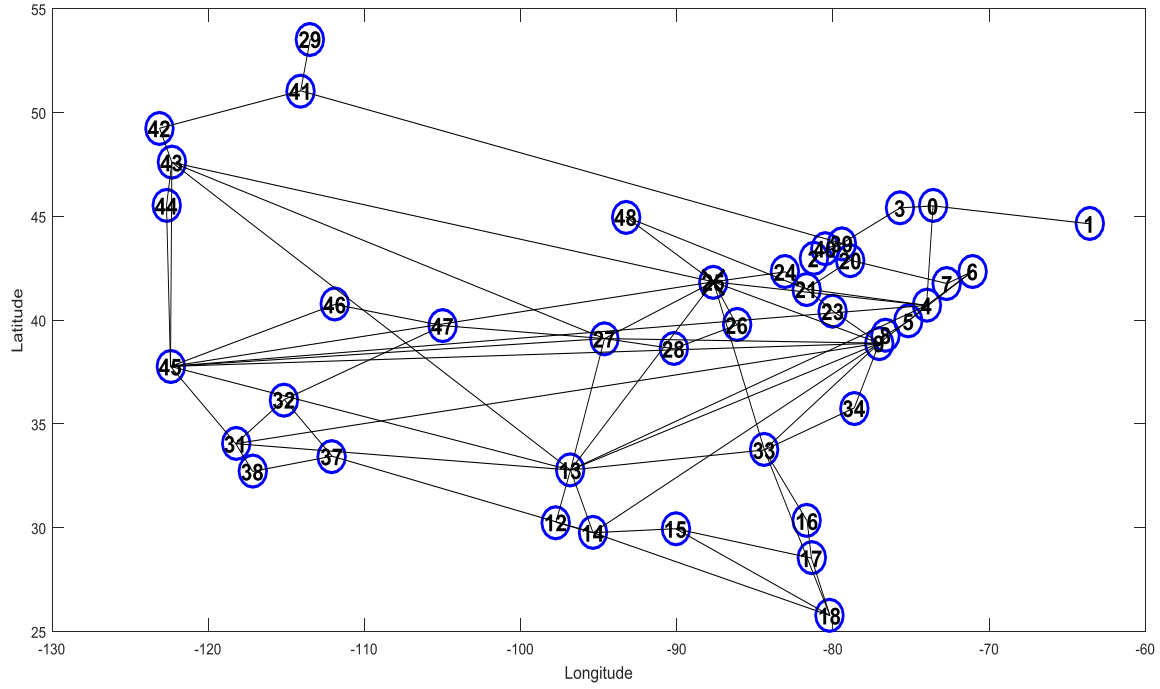


توپولوژی پیاده‌سازی شده

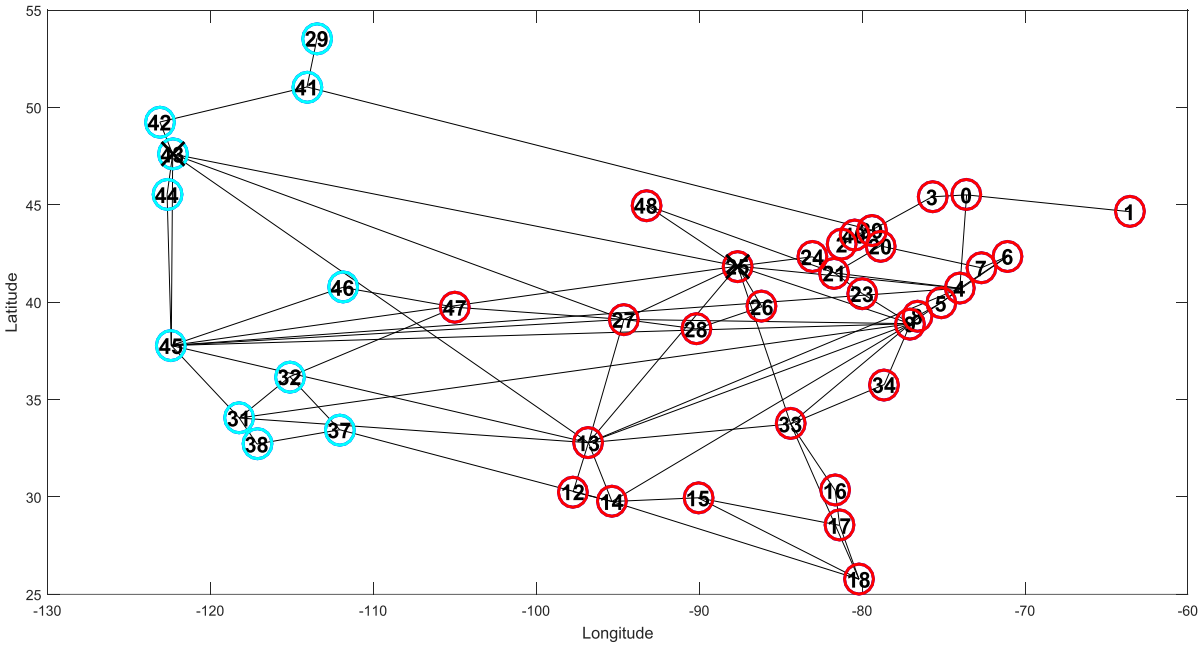
توپولوژی واقعی [۴۳]

شکل ۵- توپولوژی واقعی شبکه‌ی Uunet در مقابل توپولوژی پیاده‌سازی شده آن

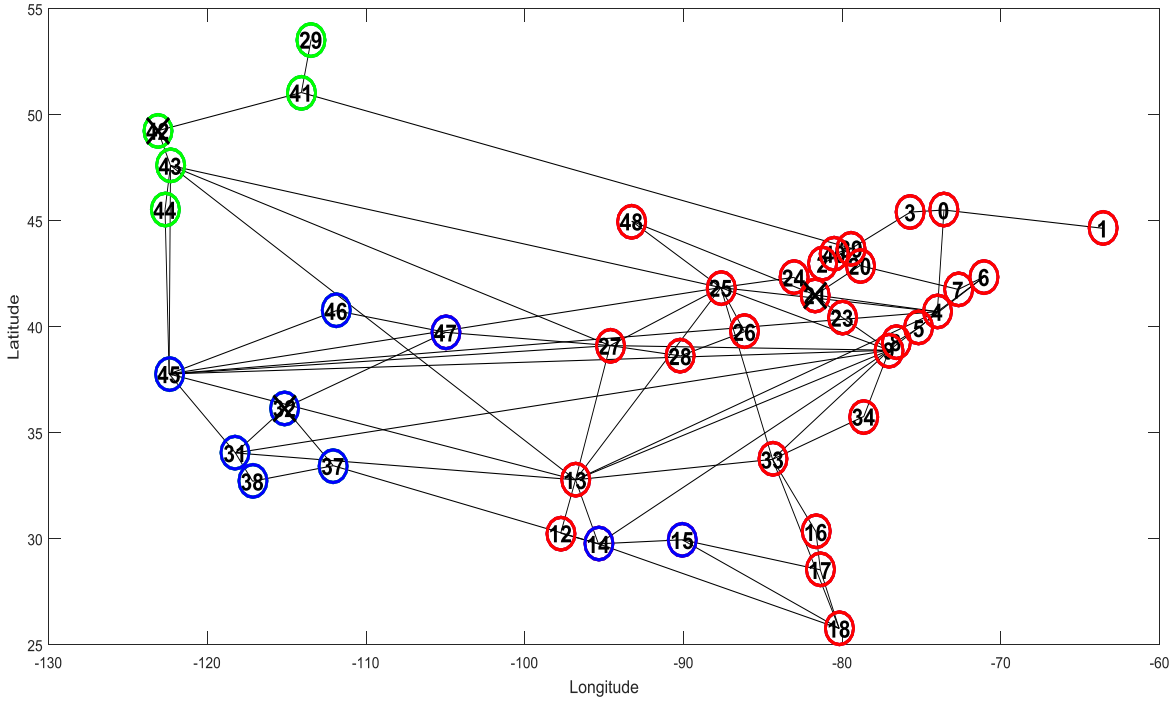
در ادامه، نتایج مربوط به اجرای هر یک از الگوریتم های CNPA و ECNPA را نمایش می‌دهیم. مراحل انتخاب سرخوشه‌ها در الگوریتم CNPA تا رسیدن به تعداد زیر-شبکه دلخواه در شکل ۶ نمایش داده شده است:



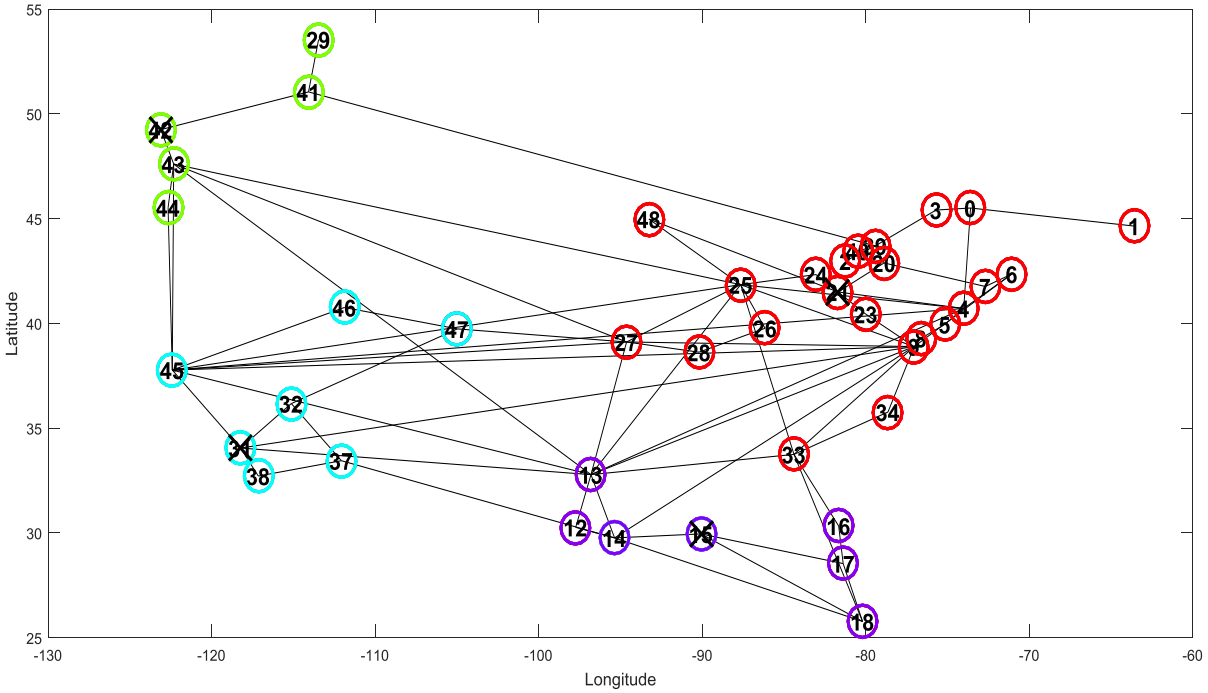
مرحله اول: ۲۵



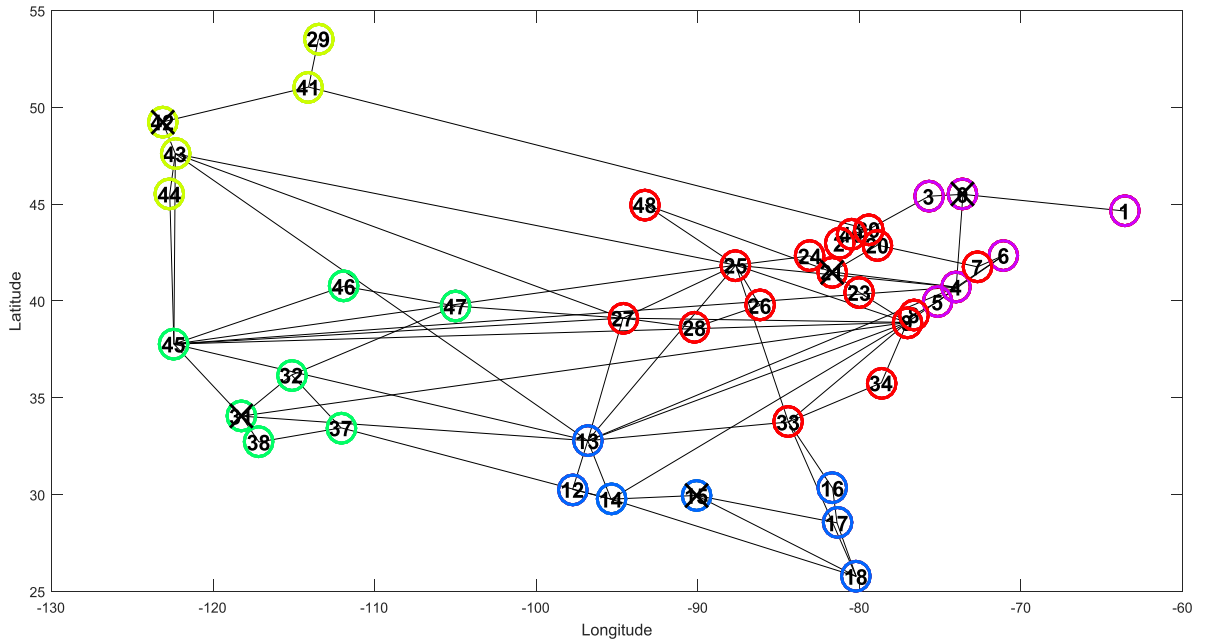
مرحله دوم: ۲۵ ۴۳



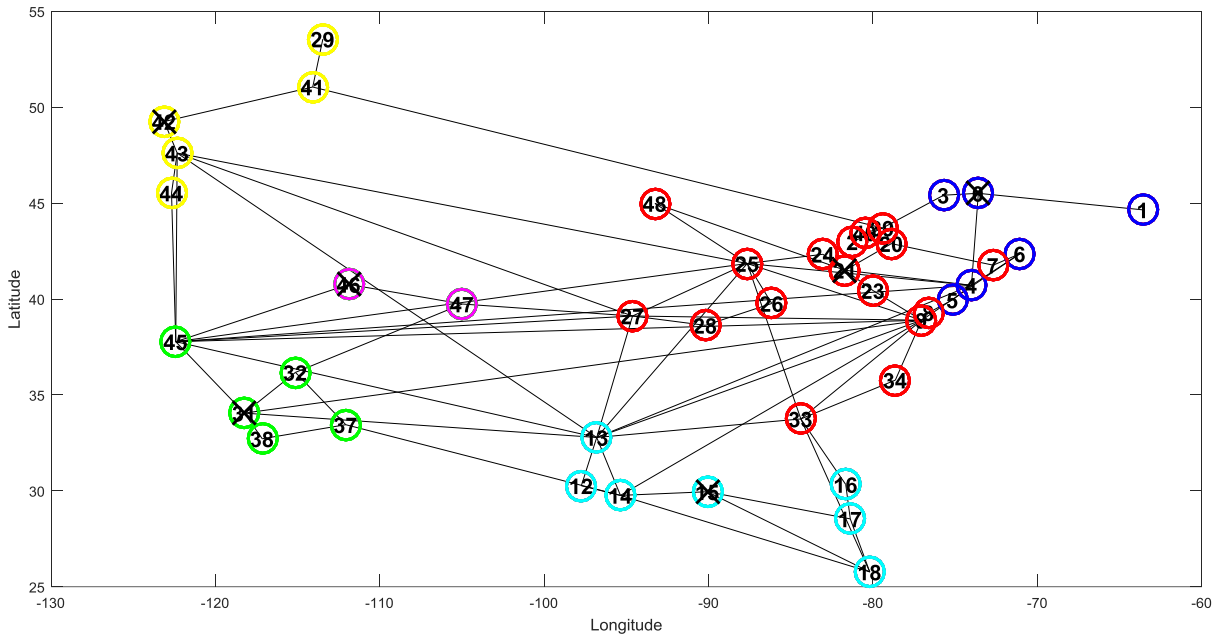
مرحله سوم: ۲۱ ۴۲ ۳۲



مرحله چهارم: ۲۱ ۴۲ ۳۱ ۱۵



مرحله پنجم: ۲۱ ۴۲ ۳۱ ۱۵ ۰

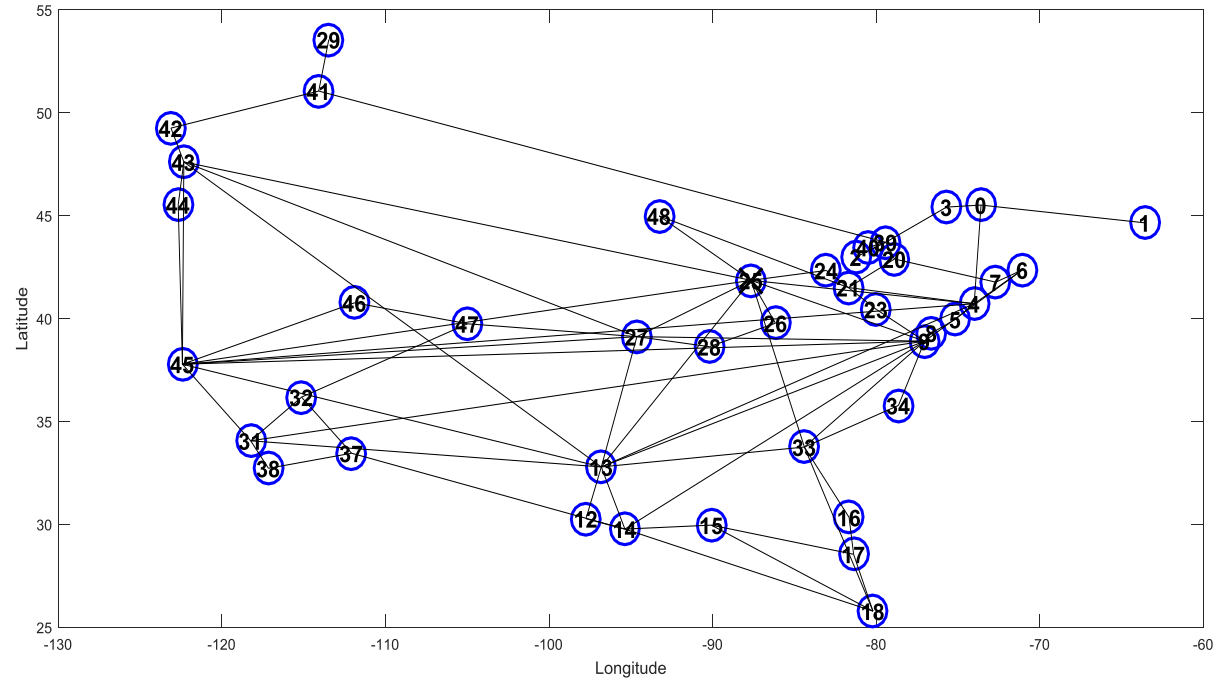


مرحله ششم: ۲۱ ۴۲ ۳۱ ۱۵ ۰ ۴۶

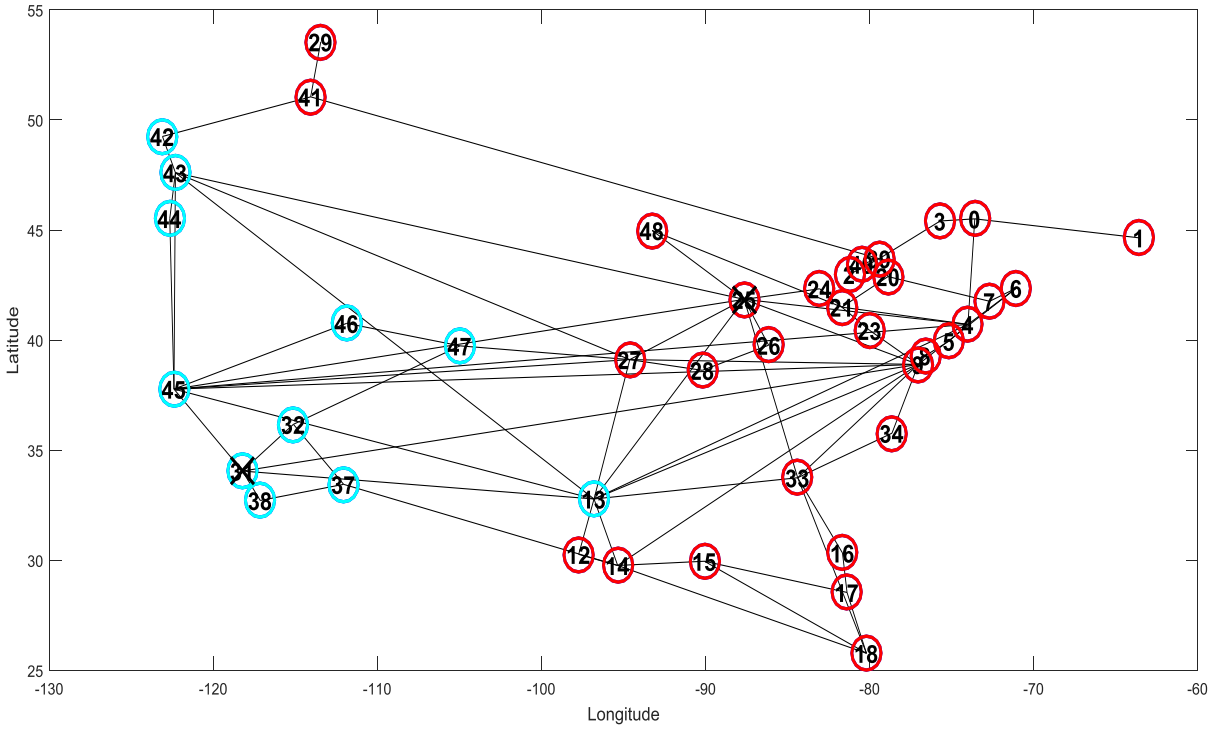
شکل ۶- اجرای الگوریتم CNPA روی شبکه Uunet به ازای k=6

حداکثر هزینه به دست آمده در تمام خوشه‌ها، بین سرخوشه ۲۱ (که معرف کنترلر می‌باشد) تا گره ۳۳ متعلق به آن (که معرف سوئیچ می‌باشد) با اجرای الگوریتم CNPA در توپولوژی Uunet برابر با ۶/۹۹۴۵ می‌باشد.

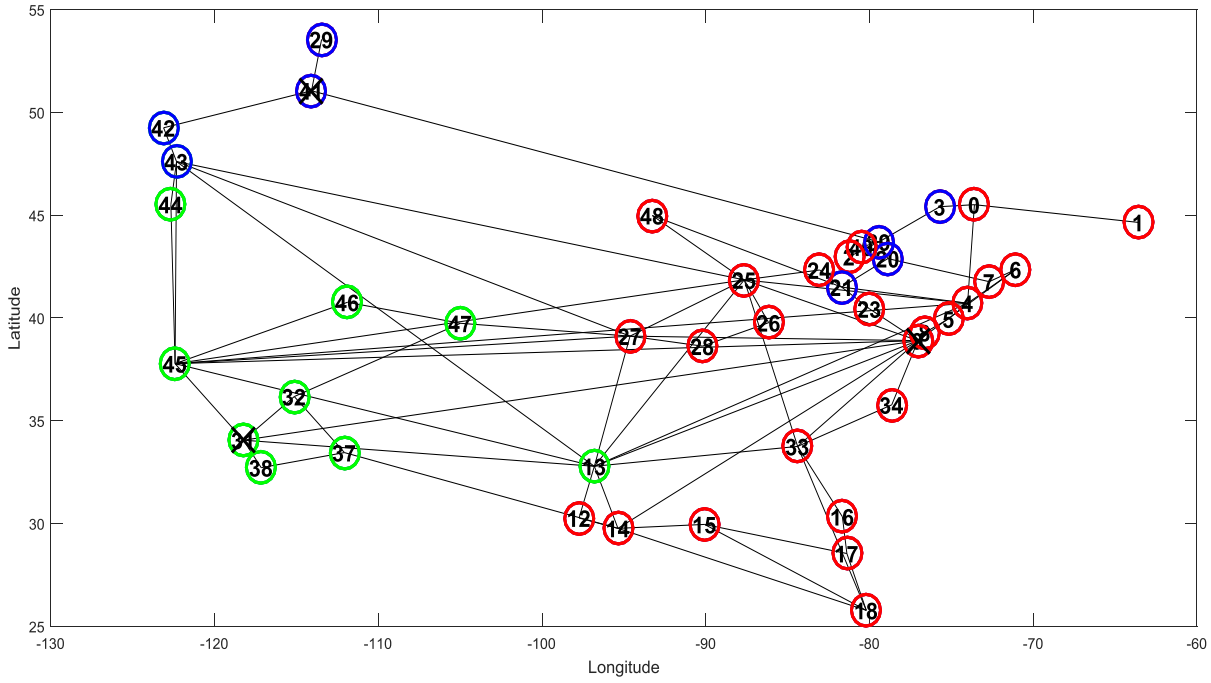
مراحل انتخاب سرخوشه‌ها در الگوریتم پیشنهادی (ECNPA) تا رسیدن به تعداد زیرشبکه دلخواه در شکل ۷ نمایش داده شده است:



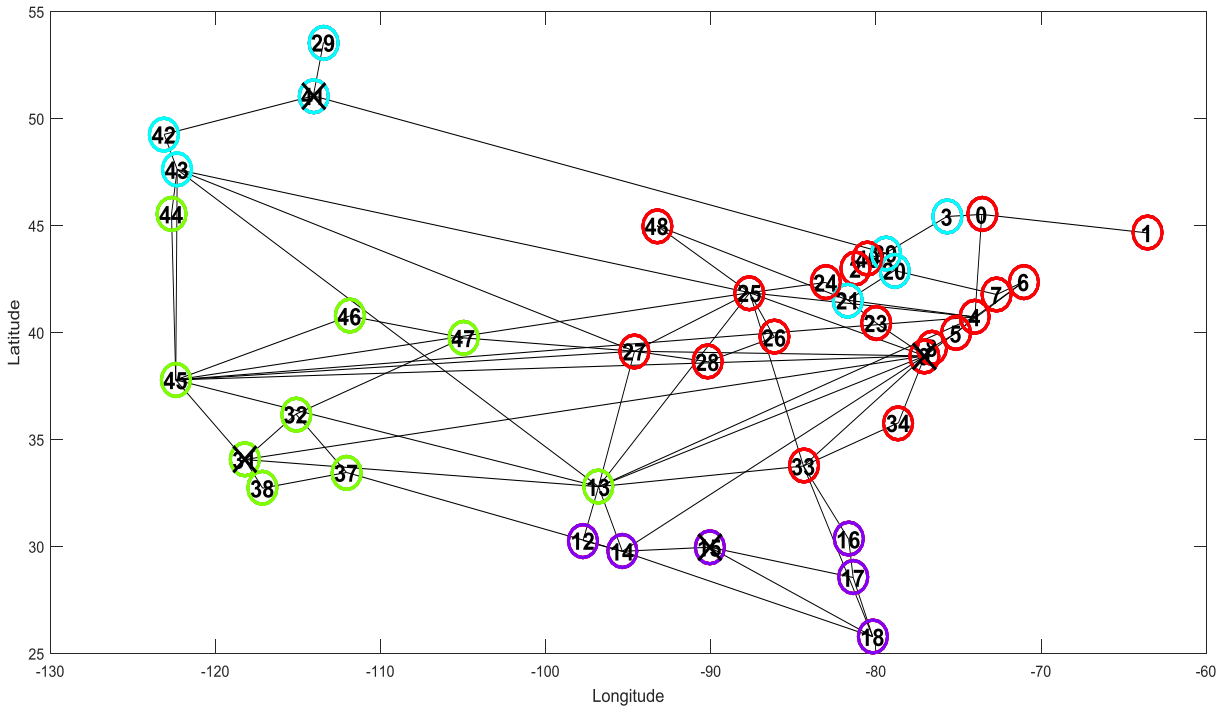
مرحله اول: ۲۵



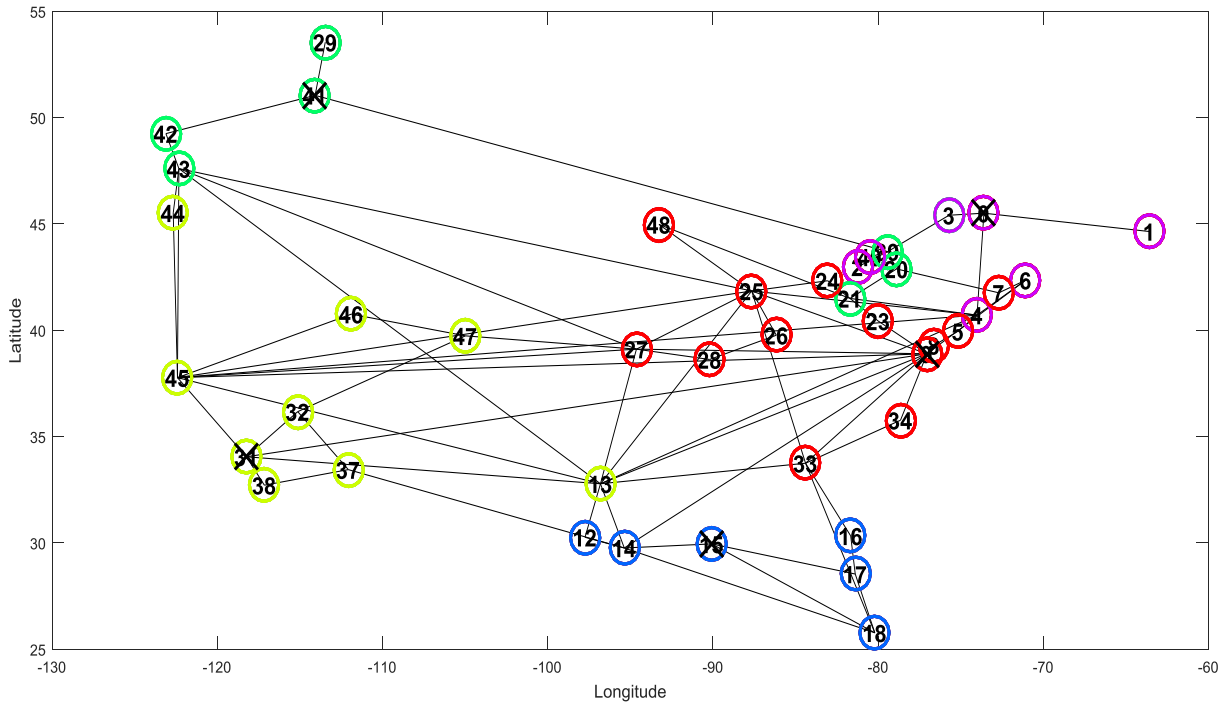
مرحله دوم: ۲۵ ۳۱



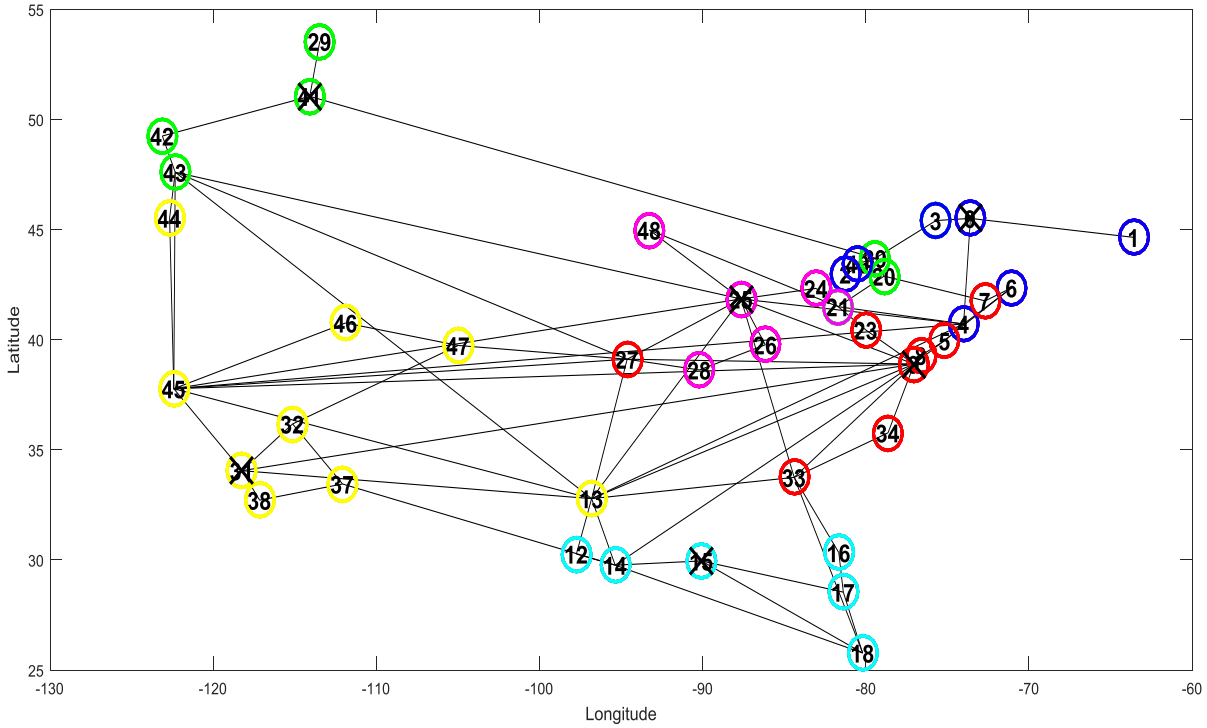
مرحله سوم: ۹ ۳۱ ۴۱



مرحله چهارم: ۹ ۳۱ ۴۱ ۱۵



مرحله پنجم: ۹ ۳۱ ۴۱ ۱۵ ۰



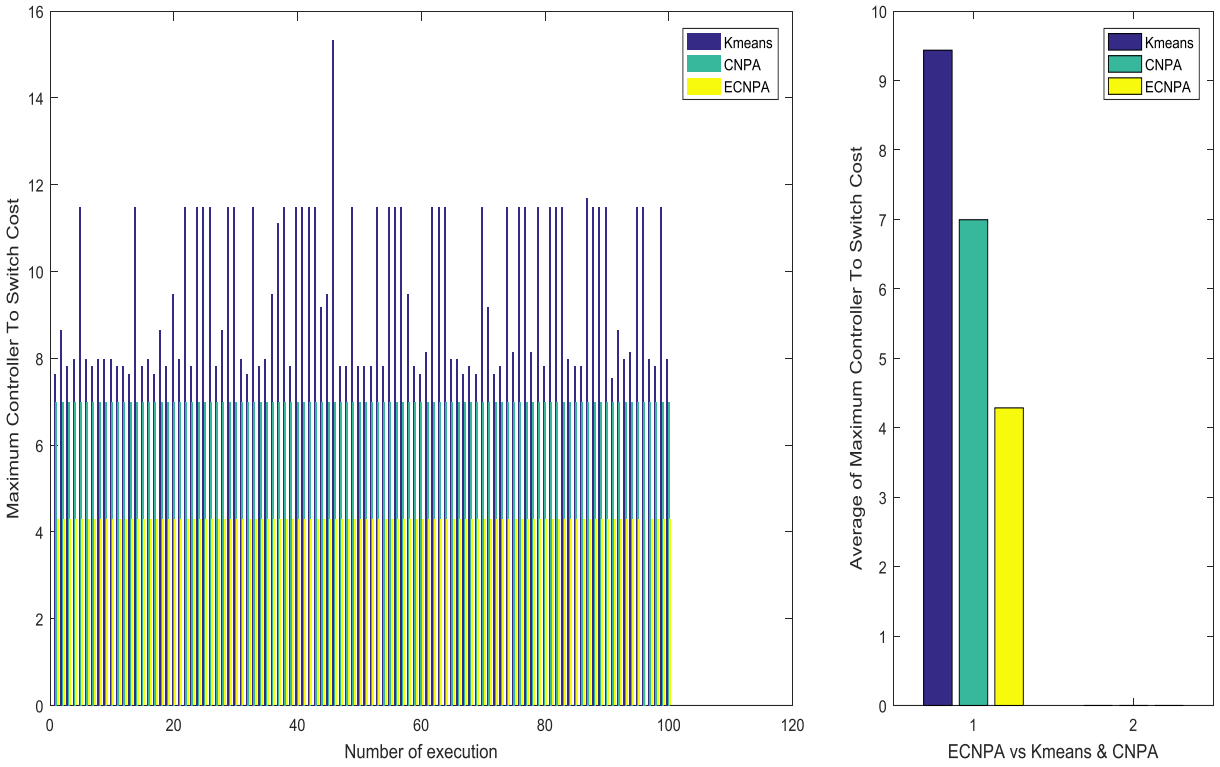
مرحله ششم: ۹ ۳۱ ۴۱ ۱۵ ۰ ۲۵

شکل ۷: اجرای الگوریتم پیشنهادی روی شبکه Uunet به ازای k=6

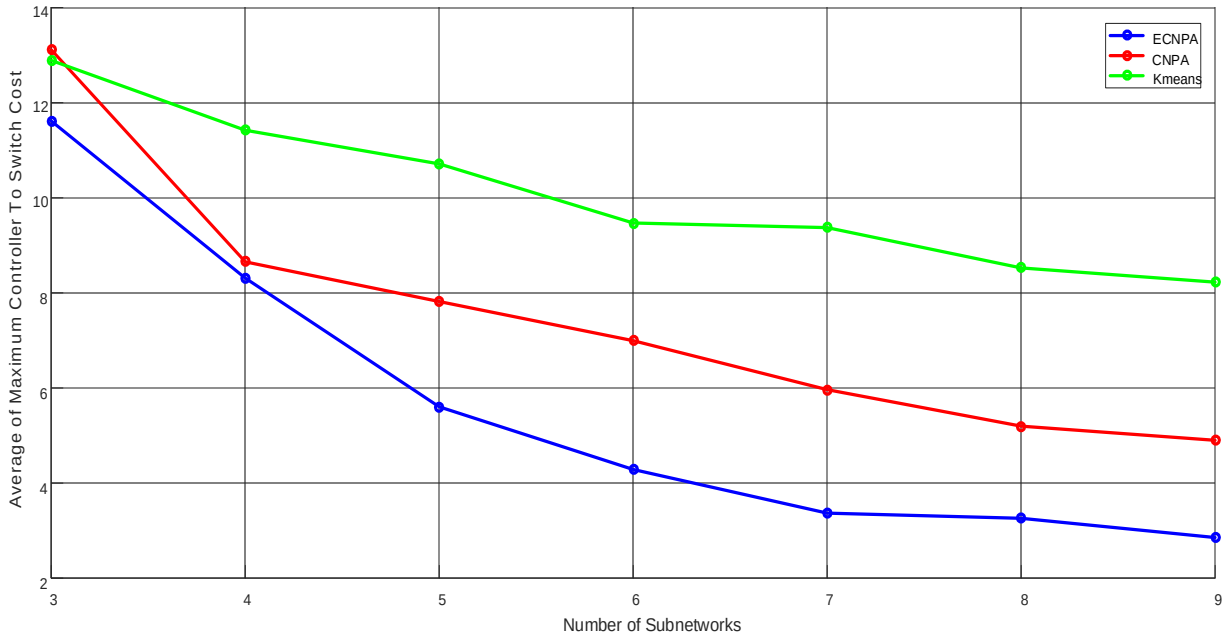
حداکثر هزینه به دست آمده در تمام خوشه‌ها، بین سرخوشه ۳۱ (که معرف کنترلر می‌باشد) تا گره ۴۶ متعلق به آن (که معرف سوئیچ می‌باشد) با اجرای الگوریتم پیشنهادی در توپولوژی Uunet برابر با ۴/۲۸۵۲ می‌باشد.

در نمودار ۱، نتایج به دست آمده در ۱۰۰ بار اجرای هر سه الگوریتم، نشان می‌دهد که در توپولوژی Uunet، میانگین حداکثر هزینه بین هر کنترلر و سوئیچ‌های مربوط به آن، در الگوریتم پیشنهادی مقدار کمتری می‌باشد. هم‌چنین نتایج شبیه‌سازی‌ها در توپولوژی Uunet نیز به وضوح نشان می‌دهد که اگرچه میزان تأخیر انتها به انتها در الگوریتم پیشنهادی ممکن است در بعضی از مواقع در مقایسه با الگوریتم CNPA کمی بیشتر باشد ولی در شرایطی که احتمال ازدحام در شبکه بالا می‌رود، الگوریتم پیشنهادی با شناسایی لینک‌های گلوگاه در مسیرهای ارتباطی هر گره با سایر گره‌ها، توانسته به خوبی ازدحام را در شبکه کنترل نماید و با در نظر گرفتن دو معیار تأخیر و میزان مشغول بودن لینک‌ها، فرایند قرارگیری و توزیع کنترل‌ها را در عمل خوشه‌بندی با دقت بالاتری انجام دهد و کارایی شبکه را افزایش دهد.

در نمودار ۲ نیز یک مقایسه‌ی کلی از هر سه الگوریتم زمانی که شبکه را به زیرشبکه‌های مختلفی تقسیم‌بندی می‌کنیم، ارائه شده است. نتایج به دست آمده نشان می‌دهد که با افزایش تعداد زیرشبکه‌ها در توپولوژی Uunet، میانگین هزینه‌ی انتها به انتها کاهش می‌یابد و الگوریتم پیشنهادی در کاهش میانگین هزینه‌ی انتها به انتها، نسبت به دو الگوریتم دیگر، بهتر عمل کرده است.



نمودار ۱- مقایسه نتایج الگوریتم پیشنهادی با الگوریتم K-means و CNPA در شبکه Uunet به ازای k=6



نمودار ۲- مقایسه‌ی کلی هر سه الگوریتم به ازای تعداد زیرشبکه‌های مختلف در توپولوژی Uunet

۶- نتیجه‌گیری

در این پژوهش به بررسی روش‌های جدیدی برای کاهش هزینه‌ی انتها به انتها (شامل تأخیر انتها به انتها و درصد مشغول بودن لینک‌ها) بین کنترلرها و سوئیچ‌های مرتبط با آن‌ها پرداختیم. در ابتدا تأخیر انتها به انتها در بین کنترلرها و سوئیچ‌ها را بررسی و به صورت کیفی تدوین نمودیم. در مرحله بعد به شناسایی لینک‌های گلوگاه که عامل ایجاد ازدحام هستند، پرداختیم و یک الگوریتم مبتنی بر خوشه-بندی را برای بخش‌بندی شبکه ارائه دادیم تا یک شبکه را به چندین زیرشبکه تقسیم کنیم تا بتوان هزینه انتها به انتها را

کاهش داد. به منظور ارزیابی کارائی الگوریتم پیشنهادی، شبیه‌سازی‌های زیادی را تحت توپولوژی‌های واقعی در پروژه‌ی Topology Zoo شامل توپولوژی Chinanet کشور چین، توپولوژی Uunet کشور آمریکا، توپولوژی DFN کشور آلمان، و توپولوژی Rediris کشور اسپانیا انجام دادیم. نتایج شبیه‌سازی‌ها نشان می‌دهد که الگوریتم پیشنهادی می‌تواند میانگین هزینه انتها به انتها در بین کنترلرها و سوئیچ‌های مربوطه را نسبت به الگوریتم‌هایی مانند K-means کاهش دهد. مراکز اولیه‌ی الگوریتم K-means به

صورت تصادفی انتخاب می‌شوند و از این رو نتیجه‌ی پارتیشن شبکه به ازای هر بار اجرا متغیر است. بنابراین بدیهی است که ماکزیمم هزینه انتها به انتها در بین کنترلر و سوئیچ‌های مربوطه در زیرشبکه‌ها افزایش پیدا می‌کند. نتایج پیاده‌سازی به ازای صد بار اجرای هر یک از الگوریتم‌ها مورد بررسی قرار گرفته است و نتایج به دست آمده نشان می‌دهد که الگوریتم پیشنهادی می‌تواند همان نتایج را به ازای هر بار اجرا به دست آورد. دلیل آن این بوده که مراکز اولیه‌ی مناسبی محاسبه شده است و تعداد ثابتی از مراکز اولیه منجر به تقسیم بندی ثابت شبکه می‌شود. هم‌چنین الگوریتم پیشنهادی می‌تواند با شناسایی لینک‌های گلوگاه، از به وجود آمدن ازدحام در شبکه جلوگیری به عمل آورد.

در انتها لازم به ذکر است که در راستای استفاده از کنترلر-های توزیع‌شده و قرار دادن آن‌ها در شبکه‌های مبتنی بر نرم‌افزار می‌توان از الگوریتم‌های ابتکاری و فرا ابتکاری مانند الگوریتم ازدحام ذرات، الگوریتم ژنتیک، الگوریتم کلونی زنبور عسل، الگوریتم تکامل تفاضلی، و ... نیز جهت انجام دقیق‌تر و بهتر عمل خوشه‌بندی و توزیع کنترلرها استفاده کرد.

منابع

1.N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 69-74, 2008.

2.W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu, "Performance modelling and analysis of software-defined networking under bursty multimedia traffic," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, p. 77, 2016.

3.H. Huang, H. Yin, G. Min, H. Jiang, J. Zhang, and Y. Wu, "Data-driven information plane in software-defined networking," *IEEE Communications Magazine*, vol. 55, pp. 218-224, 2017.

4.D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the Benefits of Incremental {SDN} Deployment in Enterprise Networks," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 333-345.

5.N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 87-98, 2014.

6.*OpenFlow Switch Specification Version 1.5.1*. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

7.B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 1617-1634, 2014.

8.A. Ominike Akpovi, A. Adebayo, and F. Osisanwo, "Introduction to Software Defined Networks (SDN)," 2016.

9.W. Stallings, "Software-defined networks and openflow," *The internet protocol Journal*, vol. 16, pp. 2-14, 2013.

10.G. Wang, Y. Zhao, J. Huang, and W. Wang, "The controller placement problem in software defined networking: A survey," *IEEE Network*, vol. 31, pp. 21-27, 2017.

11.B. Heller, R. Sherwood, and N. McKeown, "The controller placement

networks," *China Communications*, vol. 11, pp. 38-54, 2014.
21.M. Tanha, D. Sajjadi, and J. Pan, "Enduring node failures through resilient controller placement for software defined networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1-7.
22.A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE communications letters*, vol. 19, pp. 30-33, 2015.
23.H. K. Rath, V. Revoori, S. M. Nadaf, and A. Simha, "Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1-6.
24.A. Ruiz-Rivera, K.-W. Chin, and S. Soh, "GreCo: An energy aware controller association algorithm for software defined networks," *IEEE communications letters*, vol. 19, pp. 541-544, 2015.
25.M. T. I. ul Huque, G. Jourjon, and V. Gramoli, "Revisiting the controller placement problem," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, 2015, pp. 450-453.
26.A. Sallahi and M. St-Hilaire, "Expansion model for the controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 21, pp. 274-277, 2017.
27.D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," in *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, 2013, pp. 1-9.
28.S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Transactions on Network and Service Management*, vol. 12, pp. 4-17, 2015.
29.D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive resource management and control in software defined problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 7-12.
12.G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 18, pp. 1339-1342, 2014.
13.S. Khuller and Y. J. Sussmann, "The capacitated k-center problem," *SIAM Journal on Discrete Mathematics*, vol. 13, pp. 403-418, 2000.
14.Y. Hu, T. Luo, W. Wang, and C. Deng, "On the load balanced controller placement problem in Software defined networks," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 2430-2434.
15.L. Han, Z. Li, W. Liu, K. Dai, and W. Qu, "Minimum control latency of SDN controller placement," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 2175-2180.
16.Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, 2011, pp. 1-6.
17.Y.-N. Hu, W.-D. Wang, X.-Y. Gong, X.-R. Que, and S.-D. Cheng, "On the placement of controllers in software-defined networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 92-171, 2012.
18.L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: An enhanced controller placement strategy for improving SDN survivability," in *2014 IEEE Global Communications Conference*, 2014, pp. 1909-1915.
19.Y. Jimenez, C. Cervelló-Pastor, and A. J. García, "On the controller placement for designing a distributed SDN control layer," in *2014 IFIP Networking Conference*, 2014, pp. 1-9.
20.Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined

pdf-page.html?/content/dam/common/documents/content-types/solution-brief/brocade-mlx-service-provider-sb.pdf
37.*Corsa's DP2100 SDN switching and routing platform*. Available: http://www.corsa.com/products/dp2100/
38.R. Daniels and D. Whittaker. (2015). *Benchmarking the SDN Switch*. Available: https://www.opennetworking.org/images/stories/sdn-solution-showcase/germany2015/Spirent%20-%20Benchmarking%20the%20SDN%20Switch.pdf
39.C. Veness, "Calculate distance and bearing between two Latitude/Longitude points using Haversine formula in JavaScript," *Movable Type Scripts*, 2011.
40.G. Wang, Y. Zhao, J. Huang, and R. M. Winter, "On the data aggregation point placement in smart meter networks," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1-6.
41.S. Skiena, "Dijkstra's algorithm," in *Implementing discrete mathematics: combinatorics and graph theory with mathematica*, ed: Addison-Wesley Reading, MA, 1990, pp. 225-227.
42.S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, pp. 1765-1775, 2011.
43.(2019, 02/05/2019). *The Internet Topology Zoo*. Available: http://www.topology-zoo.org/
30.J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, and T. Li, "Density cluster based approach for controller placement problem in large-scale software defined networkings," *Computer Networks*, vol. 112, pp. 24-35, 2017.
31.B. Zhang, X. Wang, L. Ma, and M. Huang, "Optimal controller placement problem in Internet-oriented software defined network," in *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2016, pp. 481-488.
32.G. Wang, Y. Zhao, J. Huang, and Y. Wu, "An effective approach to controller placement in software defined wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, pp. 344-355, 2017.
33.M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, *et al.*, "Dynamic Controller Provisioning in Software Defined Networks," in *CNSM*, 2013, pp. 18-25.
34.Y. Hu, T. Luo, N. C. Beaulieu, and C. Deng, "The energy-aware controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 21, pp. 741-744, 2017.
35.G. Ishigaki and N. Shinomiya, "Controller placement algorithm to alleviate burdens on communication nodes," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, 2016, pp. 1-5.
36.*SDN-Enabled Programmatic Control of the Network*. Available: http://www.brocade.com/en/backend-content/