مسیریابی قابل اطمینان در شبکه‌های روی تراشه‌ی آگاه از ازدحام

مریم رضایی راوری[۱]، وحید ستاری نائینی[۲]
[۱] کارشناسی ارشد معماری کامپیوتر، دانشگاه تحصیلات تکمیلی صنعتی و فناوری پیشرفته کرمان، m.rezaei@student.kgut.ac.ir
[۲]دانشیار بخش مهندسی کامپیوتر، دانشگاه شهید باهنر کرمان، vsnaeini @uk.ac.ir

چکیده

کارایی شبکه‌های روی تراشه تحت تأثیر الگوریتم‌های مسیریابی می‌باشد. ازدحام در شبکه با توجه به افزایش زمان تاخیر بسته، تأثیر منفی در کارایی شبکه‌ی روی تراشه دارد. قابلیت اطمینان دربرابر خرابی هم یکی از اهداف کلیدی در طراحی شبکه‌های روی تراشه است. برای دستیابی به عملکرد بهتر همراه با تحمل‌پذیری خطا در شبکه‌ی روی تراشه دو تابع کلیدی مورد نیاز است: الف) توانایی جلوگیری از مسیرهای متراکم و تعادل حجم ترافیک و ب) توانایی تحمل خطاها و ارائه یک سیستم کارا حتی در صورت وجود مشکل فیزیکی. بدین منظور در این مقاله یک مدل هزینه برای انتخاب مسیری با قابلیت اطمینان بیشتر و تراکم کمتر پیشنهاد شده است. در این مدل، ابتدا از الگوریتم مسیریابی آگاه از ازدحام مبتنی‌بر روش Q-Learning برای بررسی ازدحام در شبکه استفاده می‌شود؛ سپس برای درنظرگرفتن قابلیت اطمینان، وضعیت لینک‌های مجاور بررسی می‌شود. در نهایت با توجه به اهمیت قابلیت اطمینان به این پارامتر وزن بیشتری اختصاص داده می‌شود و مسیری با کمترین هزینه برای ارسال بسته‌ها انتخاب می‌شود. نتایج حاصل از شبیه‌سازی تحت دو الگوی ترافیکی نشان می‌دهد که عملکرد روش پیشنهادی در حضور لینک‌های خطا نسبت به الگوریتمی که فقط ازدحام را بررسی می‌کند بهبود پیدا می‌کند.

کلید واژه

شبکه‌ی روی تراشه، قابلیت اطمینان، تحمل‌پذیری خطا، الگوریتم مسیریابی آگاه از ازدحام، Q-learning

مقدمه

با پیشرفت تکنولوژی و کوچکتر شدن اندازه ترانزیستورها، پیچیدگی سیستم‌های چندپردازنده‌ای بر روی تراشه‌ها در حال افزایش است. شبکه‌ی روی تراشه[۱]معماری نسبتاً جدیدی است که به علت ناکارآمدی معماری گذرگاه مشترک در سیستم روی تراشه[۲]اخیراً بسیار مورد توجه محققین قرار گرفته است. این معماری مزایایی از قبیل کاهش توان مصرفی، مقاومت بیشتر در برابر نویز محیطی، کوتاه‌تر شدن زمان طراحی و کوچکتر شدن حجم مدار و ... را در بر دارد [۱].

تا به امروز، بسیاری از NoCها همبندی مش را به دلیل ساختار ساده، سهولت اجرا و پشتیبانی برای استفاده مجدد به کار گرفته‌اند. در NoC، مبتنی‌بر مش، هر هسته توسط یک رابط شبکه محلی[۳]به یک سوئیچ متصل است. هر سوئیچ از طریق لینک دو طرفه به همسایگان خود متصل می‌شود. برای هر بسته، ممکن است چندین مسیر از هر مبدا به هر مقصد وجود داشته باشد. مسیر بسته‌ها در شبکه به وسیله‌ی الگوریتم مسیریابی مشخص می‌شود. که این امر باعث شده الگوریتم‌های مسیریابی تأثیر مستقیم روی کارایی و توان مصرفی داشته باشند و به عنوان یکی از مهمترین موضوعات در این معماری مطرح شوند [۲, ۳].

الگوریتم مسیریابی از لحاظ تنوع مسیر و انطباق، می‌تواند به سه گروه مسیریابی قطعی[۴] مسیریابی تا حدی تطبیقی[۵]و مسیریابی تطبیقی[۶]طبقه‌بندی شود. از آن‌جایی که اندازه شبکه افزایش می‌یابد، ازدحام یکی از مهم‌ترین پارامترهای محدود کننده کارایی NoC است. الگوریتم‌های مسیریابی تطبیقی با در نظر گرفتن مسیرهای با تاخیر کم تا حدی مشکل ازدحام در شبکه را حل می‌کنند. در سال‌های اخیر، تحقیقات زیادی جهت بهبود بهره‌وری مسیریابی NoC صورت گرفته است. هدف بسیاری از تکنیک‌های مسیریابی تطبیقی موجود، توزیع ترافیک روی کل شبکه است. با این حال، افزایش ترافیک ممکن است منجر به ازدحام شود که در نتیجه کارایی کاهش می‌یابد و همچنین تاخیر در شبکه افزایش می‌یابد [۲].

الگوریتم‌های مسیریابی تطبیقی را می‌توان به توابع مسیریابی و انتخاب تقسیم کرد. تابع مسیریابی مجموعه‌ای از کانال‌های خروجی را براساس سوئیچ‌های فعلی و مقصد تعیین می‌کند. تابع انتخاب، یک کانال خروجی از مجموعه کانال‌های ارائه شده توسط تابع مسیریابی را انتخاب می‌کند. تابع انتخاب را می‌توان به دو دسته‌ی بدون توجه به ازدحام یا آگاه از ازدحام طبقه‌بندی کرد [۴]. در الگوریتم بدون توجه به ازدحام،

[۱] Network on Chip (NoC)
[۲] System on chip (SoC)
[۳] Network Interface (NI)
[۴] Deterministic
[۵] Partially adaptive
[۶] fully adaptive

تاریخ دریافت: ۱۳۹۶/۰۲/۱۲ تاریخ پذیرش: ۱۳۹۸/۰۴/۰۲

تصمیمات مسیریابی مستقل از شرایط ازدحام در شبکه است. به عنوان مثال، در الگوریتم مسیریابی XY، بسته ابتدا در جهت X و سپس در جهت Y ارسال می‌شود. در مقابل، الگوریتم‌های مسیریابی آگاه از ازدحام وضعیت ازدحام در شبکه را در مسیریابی در نظر می‌گیرند.

DyXY با استفاده از اطلاعات محلی مسیر را انتخاب می‌کند، که ممکن است منجر به عبور بسته از طریق مناطق متراکم شود بنابراین استفاده از اطلاعات محلی کافی نیست [۵]. معمول‌ترین روش‌های پیاده‌سازی الگوریتم‌های مسیریابی، به عنوان مثال NoP[۷]، RCA[۸] ، DBAR[۹] به ترتیب در مراجع [۶-۸]، روی جمع‌آوری اطلاعات ازدحام محلی و سراسری و تخمین مناطق متراکم در شبکه متمرکز شده‌اند. همچنین با استفاده از روش‌های یادگیری، می‌توان اطلاعات ازدحام محلی و سراسری شبکه را برای هر یک از سوئیچ‌ها ارائه کرد. Q-learning یکی از پیشرفت های مهم تقویت یادگیری است که توسط Watkins و Dayan توسعه یافته است [۹]. الگوریتم Q-learning برای ایجاد یک الگوریتم مسیریابی تطبیقی به نام Q-routing استفاده شده است. Q-Routing اجازه می‌دهد شبکه به طور مداوم با تغییر شرایط ازدحام مطابقت پیدا کند. در این روش، هر سوئیچ براساس اطلاعات تاخیر زمانی، که در یک جدول از Q-value ها ارائه شده است برای مسیریابی تصمیم‌گیری می‌کند، این مقادیر زمانی که سوئیچ یک بسته را به یکی از همسایگان خود می‌فرستد به روز می‌شود. بسته به الگوهای ترافیک و سطح بار، تنها چند Q-value به طور منظم به روز می‌شوند در حالی که بسیاری از Q-value ها در شبکه به روز نشده و در نتیجه غیرقابل اعتماد باقی می‌مانند. پیشنهاداتی برای حل این مشکل مانند CQ-Routing[۱۰]، PQ- Routing[۱۱] و DRQ Routing[۱۲] در مراجع [۱۰-۱۲] ارائه شده است. با این حال، رویکرد Q-Learning در NoC با توجه به نیاز به یک جدول مسیریابی بزرگ در هر سوئیچ سربار زیادی دارد.

به منظور کاهش حجم جدول مسیریابی در مقایسه با الگوریتم‌های معمولی Q-Routing الگوریتم C-Routing که یک روش مبتنی‌بر Q-Learning است و از روش خوشه‌بندی بهره می‌برد ارائه شده است [۱۳]. اما این روش هم جدول‌های مسیریابی بزرگی دارد، در مراجع [۱۴, ۱۵] با استفاده از روش خوشه‌بندی و کاهش تعداد و اندازه جدول‌های مسیریابی سربار کاهش چشمگیری می‌یابد. با استفاده از روش‌های یادگیری ازدحام در خوشه‌های مجاور خوشه فعلی بررسی می‌شود در مقاله [۱۵] برای یافتن مسیر مناسب خوشه‌ای که با استفاده از آن کمترین زمان برای رسیدن به مقصد مورد نیاز است انتخاب می‌شود، با وجود بهبود کارایی نسبت به روش‌های قبلی در این مقاله فرض شده که لینک‌ها سالم هستند و هیچ خرابی وجود ندارد.

همانطور که قابلیت اطمینان ارتباطات تراشه یک عامل بسیار مهم در سیستم‌های چند هسته‌ای است [۱۶]، NoC هم باید مسائل مربوط به قابلیت اطمینان را درنظر بگیرد. در مورد شبکه‌ی روی تراشه یک سری از تحقیقات مرتبط با ارزیابی قابلیت اطمینان، در ارتباط با مصرف توان و کارایی انجام شده است [۱۷]. استفاده از روش‌های بهبود قابلیت اطمینان معمولاً باعث افزایش مصرف توان و کاهش کارایی می‌گردد که بایستی یک مصالحه میان این مشخصات مهم سیستمی برقرار گردد. از سوی دیگر، عملکرد NoC به شدت تحت تأثیر ازدحام شبکه قرار دارد. الگوریتم مسیریابی در مرجع [۱۸] توانایی دور زدن مسیریاب معیوب را پس از شناسایی مسیریاب معیوب و غیرفعال دارد. اما، فقط تحمل خرابی مسیریاب (گره) را دارد و نمی‌تواند کانال‌های معیوب را پشتیبانی کند. یک الگوریتم مسیریابی تحمل‌پذیر خطا برای پشتیبانی از خطاهای گره و لینک در [۱۹] ارائه شده است.

در مرجع [۲۰]، یک الگوریتم مسیریابی قابل پیکربندی مجدد برای شبکه‌ی روی تراشه تحمل‌پذیر خطا ارائه شده است. این الگوریتم مسیریابی برای انطباق با تغییر همبندی ناشی از یک سوئیچ معیوب به صورت پویا پیکربندی می‌شود. الگوریتم دیگری در [۲۱] برای افزایش قابلیت اطمینان و اجتناب از اجزای معیوب با استفاده از پیکربندی مجدد ارائه شده است که جدول‌های مسیریابی را در یک فرآیند آفلاین تنظیم می‌کند و از کانال‌های مجازی یا مسیریابی انطباقی استفاده نمی‌کند. این الگوریتم شامل یک مرحله اولیه مسیریابی و بررسی قانون است. این قوانین، الگوریتم مسیریابی را بسته به همبندی شبکه و خطاهای موجود محدود می‌کنند. هر مسیریاب در شبکه شامل یک جدول مسیریابی است، که لیستی از پورت‌های خروجی برای هر مقصد در شبکه را نگه می‌دارد. یک روش مسیریابی تحمل‌پذیر خطا با هزینه بسیار کم نیز برای تحمل خطای لینک و مسیریاب در شبکه‌ی روی تراشه ارائه شده است [۲۲]. این الگوریتم در صورت وجود اجزای معیوب در شبکه به صورت پویا پیکربندی می‌شود. علاوه‌بر این، یک الگوریتم مسیریابی توزیع شده، تطبیقی و آگاه از ازدحام است و از دو کانال مجازی استفاده می‌کند. اشکال اصلی این روش‌ها این است که یک مکانیزم کنترل ازدحام کارآمد برای مسیریابی در شرایط ترافیک پیچیده، به منظور حفظ عملکرد سیستم تحت ترافیک سنگین ارائه نمی‌دهند. در مرجع [۲۳] یک الگوریتم مسیریابی

---

[۷] Neighbor-on-Path
[۸] Regional Congestion Awareness
[۹] Destination-Based Adaptive Routing
[۱۰] Confidence -based Q-routing
[۱۱] Predictive Q-routing
[۱۲] Dual Reinforcement Q-routing

تحمل‌پذیر خطا ارائه شده است که تاثیر منفی اجزای خرابی را که روی توان و بازدهی NoC تاثیر می‌گذارند، به حداقل می‌رساند. این کار به کمک پایش NoC و تطبیق مسیر انجام می‌شود.

در این مقاله، یک الگوریتم مسیریابی به منظور مسیریابی بسته از طریق کوتاه‌ترین مسیر در حضور لینک‌های معیوب پیشنهاد می‌شود. روش پیشنهادی قابلیت تحمل پذیری خرابی بهتری را ارائه می‌دهد و از وقوع ازدحام درشبکه جلوگیری می‌کند. این مقاله در انتخاب خروجی مناسب با قرار دادن تعدادی سوئیچ در یک خوشه لینک‌های خوشه‌های اطراف به سمت مقصد را بررسی می‌کند تا ناحیه‌ای انتخاب شود که با اطمینان بیشتری بسته‌ها به مقصد برسند، همانطور که قبلا اشاره شد عملکرد شبکه‌ی روی تراشه به میزان ازدحام بستگی دارد پس باید علاوه‌بر لینک‌های خوشه مجاور شرایط ترافیکی ناحیه‌های اطراف هم بررسی شود و بعد از بررسی تمام شرایط بدست آمده مسیر مناسب به سمت مقصد انتخاب شود.

این مقاله به چهار بخش تقسیم شده است. در بخش دوم، اطلاعاتی در مورد تکنیک‌های بررسی قابلیت اطمینان و تحمل‌پذیری خطا و ازدحام در شبکه مورد بررسی قرار می‌گیرند. بخش سوم، الگوریتم مسیریابی پیشنهادی توضیح داده شده است. در بخش چهارم، نتایج بدست آمده بیان می‌شود. در انتها در بخش پنجم نتیجه‌گیری مقاله ارائه می‌شود.

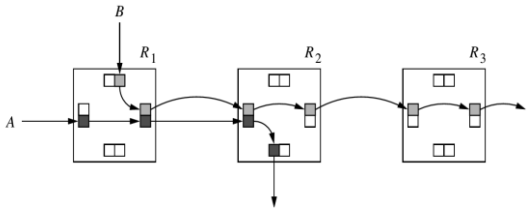
پیش زمینه

الگوریتم‌های مسیریابی پویا با توجه به وضعیت ترافیک تصمیم‌گیری می‌کنند. در این بخش، مروری از روش‌های فعلی ارائه شده است که شامل بررسی بن‌بست و الگوریتم‌های مسیریابی آگاه از ازدحام و مقاوم در برابر خطا است.

بررسی بن‌بست

به علت کمبود منابع، بن‌بست ایجاد می‌شود. بن‌بست وضعیتی است که مجموعه‌ای از بسته‌ها به طور دائمی در انتظار یکدیگر هستند و پیشرفتی در شبکه انجام نمی‌شود. بن‌بست به‌وسیله انتظار چرخشی بین بسته‌ها ایجاد می‌شود که در آن هر بسته دارای یک کانال است و منتظر کانال بعدی است. یک راه‌حل برای از بین بردن وابستگی‌های سیکل، این است که دو بسته‌ی درگیر در بن‌بست دور انداخته شوند. اما، روش خوبی نیست، زیرا ممکن است این وضعیت خیلی از اوقات در شبکه وجود داشته باشد. مدل‌های نوبت[۱۳]پیشگیری از بن‌بست را تضمین می‌کنند. راه‌حل دیگر این است که از الگوریتم مسیریابی در واحد مسیریابی استفاده شود که برمبنای مدل‌های نوبت یا اضافه کردن کانال‌های مجازی باشد [۲۴] در این مقاله از کانال مجازی به منظور پیشگیری از بن‌بست استفاده شده است.

شکل ۱ دو پیام را که از کانال فیزیکی بین مسیریاب‌های R1 و R2 عبور می‌کند نشان می‌دهد. بدون کانال مجازی، تا زمانی که پیام A به پایان برسد از پیشروی پیام B جلوگیری می‌شود. فرض کنید که کانال فیزیکی دو کانال مجازی را پیاده‌سازی می‌کند پس از ورود B، کانال فیزیکی به اشتراک گذاشته می‌شود و هر دو بسته با نصف سرعت ادامه می‌یابند [۲۴].



شکل ۱. استفاده از دو کانال مجازی برای هر کانال فیزیکی [۲۴].

مسیریابی تحمل‌پذیر خطا

NoCها با مسائلی همچون توان مصرفی، مساحت، قابلیت اطمینان و کارایی روبرو هستند. در این میان قابلیت اطمینان نیز مثل صرفه‌جویی در توان و فضای مصرفی یکی از مسائل مطرح در مباحث مربوط به NoC می‌باشد. قابلیت اطمینان یا به عبارتی تحمل‌پذیری خطا به معنی "رساندن بسته‌ها حتی وقتی پیوندها یا مسیریاب‌های معیوب در شبکه وجود دارند" است. طراحی الگوریتم‌های مسیریابی با قابلیت تحمل‌پذیری خطا در شبکه‌ی روی تراشه یکی از کارآمدترین روش‌ها برای افزایش قابلیت اطمینان تراشه‌ها است. با افزایش تعداد هسته‌ها، سیستم NoC در معرض خطاهای ساخت قرار می‌گیرد و از این رو، NoC معیوب می‌شود. خطا ممکن است در هسته‌ها (مانند عناصر پردازشی و ماژول‌های حافظه)، مسیریاب‌ها و / یا لینک‌ها رخ دهد. هنگامی که هسته معیوب است، می‌توان آن را غیرفعال کرد؛ اما سوئیچ و لینک‌های متصل شده می‌توانند به کار خود ادامه دهند. بنابراین هسته‌ی معیوب بسته‌های در حال انتقال بین هسته‌های دیگر را تحت تأثیر قرار نمی‌دهد [۲۵, ۲۶]. اما، هنگامی که یک لینک یا سوئیچ خراب است، جزء معیوب را نمی‌توان به سادگی از بین برد؛ زیرا باعث مسدود شدن بسته‌های دیگر داخل شبکه می‌شود. نقص در این اجزا ممکن است تعداد مسیرهای موجود برای مسیریابی را کاهش دهد که باعث تحویل بسته به صورت خراب و حتی خرابی سیستم شود. با توجه به مسیریاب و لینک‌های معیوب و کاهش تعداد مسیرهای موجود، توزیع ترافیک بسیار نامتعادل و ترافیک سنگین است. بنابراین، الگوریتم مسیریابی تحمل‌پذیر خطا برای حفظ صحیح عملکرد سیستم مورد نیاز است.

[۱۳] Turn Models

مسیریابی آگاه از ازدحام

الگوریتم مسیریابی آگاه از ازدحام مبتنی‌بر روش Q-learning برای اجتناب از مناطق متراکم در شبکه استفاده می‌شود. برای بدست آوردن تاخیر از روش ارائه شده در [۱۵] استفاده شده است. با در نظر گرفتن شبکه به مناطق مختلف (خوشه) به جای یک جدول Q برای هر سوئیچ هر کدام از خوشه‌ها یک جدول CQ[۱۴] دارند که کیفیت مسیرهای دیگر را تخمین می‌زند. از آنجا که کوتاه‌ترین مسیر برای رسیدن به مقصد نیاز است، حداکثر دو جهت نیاز به بررسی دارد. بنابراین، جدول CQ همانطور که در شکل ۲ نشان داده شده شامل $C-1$ سطر و ۲ ستون است؛ که در آن C تعداد خوشه‌های شبکه است. مقادیر Q در هر سطر مقدار تاخیر از خوشه فعلی برای رسیدن به خوشه مورد نظر از طریق هر یک از خوشه‌های همسایه در جهت x یا y را نشان می‌دهد. در رابطه ۱ زمان تخمینی برای رسیدن به مقصد مطابق با جدول‌های مربوط به هر خوشه در دو جهت x و y بدست می‌آید.

$$Latency\_Dir = CQ(Dir, Cd) \qquad (۱)$$

در این رابطه Cd خوشه مقصد و Dir، جهت x و y است.

خوشه‌ها	X-Dir	Y-Dir
C0		
C1		
C2		
. . .		
C15		

شکل ۲. جدول CQ برای خوشه ۱ [۱۵].

با استفاده از روش‌های یادگیری، اطلاعات ازدحام محلی و سراسری شبکه به هر یک از خوشه‌ها داده شده است. این مقادیر زمانی که خوشه یک بسته را به یکی از خوشه‌های همسایه می‌فرستد به روز می‌شود. اگر مقادیر Q برای مدت زمان طولانی به‌روز نشوند، وضعیت ازدحام واقعی شبکه را منعکس نمی‌کند. برای حل این مشکل، در هر خوشه مقادیر Q، با دریافت بسته‌ی داده با استفاده از رابطه ۲ به روز می‌شود.

$$Q_{Cc}(Cu, Cs)_{new} = Q_{Cc}(Cu, Cs)_{old} + 0.5(Q_{Cu}(Cy, Cs) + q_{Cu} - Q_{Cc}(Cu, Cs)_{old}) \qquad (۲)$$

در رابطه ۲، $Q_{Cu}(Cy, Cs)$ حداقل مقدار تاخیر تخمینی از خوشه قبلی به خوشه مبدا از طریق همسایه‌های آن نشان می‌دهد و q_{Cu} میانگین تعداد اسلات‌های اشغالی در پورت خروجی بافرهای ورودی است.

با خروج از خوشه Cc بسته یادگیری تولید شده و به خوشه قبلی تحویل داده می‌شود و این خوشه با استفاده از رابطه ۳ به سمت خوشه مقصد به‌روزرسانی می‌شود. به عبارت دیگر، بسته یادگیری و بسته داده اطلاعات ازدحام را در طول مسیر خود در یک خوشه جمع‌آوری می‌کنند. سپس، این اطلاعات برای به روز رسانی مقدار Q در خوشه قبلی و بعدی استفاده می‌شود.

$$Q_{Cu}(Cc, Cd)_{new} = Q_{Cu}(Cc, Cd)_{old} + 0.5(Q_{Cc}(Cn, Cd) + q_{Cc} - Q_{Cu}(Cc, Cd)_{old}) \qquad (۳)$$

در این رابطه، $Q_{Cc}(Cn, Cd)$ حداقل مقدار تاخیر تخمینی را از خوشه Cc به خوشه مقصد از طریق همسایه‌های آن نشان می‌دهد و q_{Cc} میانگین تعداد اسلات‌های اشغالی در پورت ورودی بافرهای ورودی است.

فرمت بسته

همان‌طور که بیان شد، بسته‌های داده و بسته‌های یادگیری در شبکه عبور می‌کنند. فرمت بسته داده مطابق شکل ۳ است. که در آن:

جهت: تعیین جهتی که در آن بسته داده از خوشه فرستنده به گیرنده ارسال می‌شود. از آن‌جایی که هر خوشه به حداکثر چهار خوشه همسایه متصل است، دو بیت برای شناسایی جهت شناسه خوشه همسایه کافی است.

اندازه بافر: میانگین تعداد اسلات‌های بافر اشغالی در بافرهای ورودی درون یک خوشه از جایی که بسته وارد خوشه می‌شود تا وقتی که آن را ترک می‌کند.

...	اندازه بافر	جهت	شناسه نود مقصد	شناسه نود مبدا
	٤ بیت	٢ بیت	٦ بیت	٦ بیت

شکل ۳. فرمت بسته داده [۱۵].

فرمت بسته یادگیری مطابق شکل ۴ است. در اینجا:

جهت: با استفاده از بسته‌ی داده از روی جهت شناسه خوشه گیرنده یا با استفاده از بسته یادگیری از روی جهت شناسه خوشه فرستنده تعیین می‌شود.

ازدحام تخمینی جدید: از مجموع مقادیر ازدحام سراسری و محلی بدست می‌آید. ازدحام محلی از تقسیم مجموع اسلات‌های بافر اشغالی به تعداد گام‌های بسته در خوشه بدست می‌آید.

شناسه خوشه مقصد	ازدحام تخمینی جدید	جهت
٤ بیت	٤ بیت	٢ بیت

شکل ۴. فرمت بسته یادگیری [۱۵].

[۱۴] Cluster Q-table

الگوریتم مسیریابی آگاه از ازدحام با قابلیت اطمینان

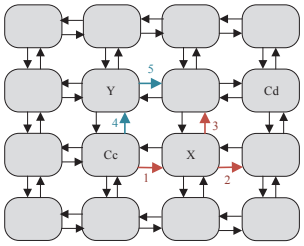
ازدحام شبکه تأثیر منفی بر روی عملکرد شبکه‌ی روی تراشه با توجه به افزایش زمان تاخیر بسته دارد. الگوریتم‌های مسیریابی آگاه از ازدحام برای کاهش ازدحام شبکه‌ی روی تراشه توسعه یافته‌اند. در این مقاله، از یک الگوریتم مسیریابی آگاه از ازدحام مبتنی‌بر روش‌های Q-learning برای اجتناب از مناطق متراکم در شبکه استفاده شده است. با استفاده از روش‌های یادگیری، اطلاعات ازدحام محلی و سراسری شبکه برای هر خوشه که شامل تعدادی سوئیچ است، ارائه شده است. در این الگوریتم واحد مسیریابی که در آن تصمیم گرفته می‌شود از کدام کانال خروجی یک بسته تحویل داده شود، براساس انتخاب خوشه‌ای با کمترین هزینه است؛ طبق رابطه ۴ هزینه شامل تاخیر ارسال پیام (Latency) و هزینه قابلیت اطمینان (Reliability Cost) است، که در ادامه قابلیت اطمینان بررسی می‌شود. از آن‌جایی که در مسیریابی کمینه بسته‌ها می‌توانند فقط در دو جهت تحویل داده شوند بنابراین این رابطه فقط برای دو جهت x و y نیاز به محاسبه دارد. در این رابطه با توجه به اهمیت قابلیت اطمینان پارامتر α باید مقداری در محدوده‌ی [۰,۵-۱] داشته باشد. از آنجایی که مقادیر نزدیک به یک وزن ازدحام را تقریبا نادیده می‌گیرد و مقادیر نزدیک به ۰,۵ به ازدحام وزن قابل توجهی می‌دهد، از اینرو با توجه به اهمیت بیشتر قابلیت اطمینان در این مقاله، مقدار α، ۰,۸ درنظر گرفته شده است.

$$Cost\_Dir = \alpha\, R\_Dir + (1-\alpha) Latency\_Dir \qquad (۴)$$

قابلیت اطمینان

قابلیت اطمینان با استفاده از هزینه قابلیت اطمینان بیان می‌شود، بالاترین هزینه قابلیت اطمینان کمترین قابلیت اطمینان را دارد. هزینه قابلیت اطمینان از مبدا به مقصد به این صورت است که اگر مسیری از مبدا به مقصد وجود داشته باشد هزینه قابلیت اطمینان '۰' و در غیر این صورت '۱' در نظر گرفته شده است [۲۷]. ابتدا لینک‌های خروجی از خوشه فعلی و خوشه‌های مجاور به سمت مقصد بررسی می‌شوند و سپس، از احتمال خرابی لینک‌ها و تعداد لینک‌های معیوب داخل خوشه‌های جهت x و y برای محاسبه هزینه قابلیت اطمینان کل مسیر استفاده می‌شود. به این ترتیب خوشه فعلی اطلاعاتی راجع‌به لینک‌های خوشه‌های مجاور به سمت مقصد بدست می‌آورد. شبه کد ارائه شده در الگوریتم ۱ نحوه بدست آوردن قابلیت اطمینان را در الگوریتم پیشنهادی نشان می‌دهد. خطوط (۱-۱۰) لینک‌های خروجی از خوشه فعلی در جهت x (Rx_1) و لینک‌های خروجی از خوشه جهت x به سمت مقصد (Rx_2) را بررسی می‌کند (شکل ۵ لینک‌های شماره‌ی ۱،۲،۳) و سپس هزینه قابلیت اطمینان (R1)، از مجموع Rx_1 و Rx_2 بدست می‌آید. خطوط (۱۱-۲۱) لینک‌های خروجی از خوشه فعلی درجهت y (Ry_1) و لینک‌های خروجی از خوشه جهت y (Ry_2) به سمت مقصد را بیان می‌کنند (شکل ۵ لینک‌های شماره‌ی ۴،۵) و سپس هزینه قابلیت اطمینان (R2)، از مجموع Ry_1 و Ry_2 بدست می‌آید. بعد از بررسی لینک‌های خوشه‌های جهت x و y احتمال خرابی گره‌های داخل هر خوشه در جهت x و y (ProbabilityFault_Dir) و تعداد لینک‌های معیوب داخل هر خوشه در جهت x و y (NumberFaultyLink_Dir) تعیین کننده قابلیت اطمینان کل در جهت x و y است (خطوط ۲۲-۲۸).

بعد از بررسی تاخیر و قابلیت اطمینان با توجه به شرایط لینک‌ها و با استفاده از رابطه‌ی ۱ هزینه در جهت x و y بدست می‌آید و خوشه‌ای که هزینه کمتری دارد انتخاب می‌شود و مسیریابی مطابق با جهت انتخاب شده انجام می‌گیرد.



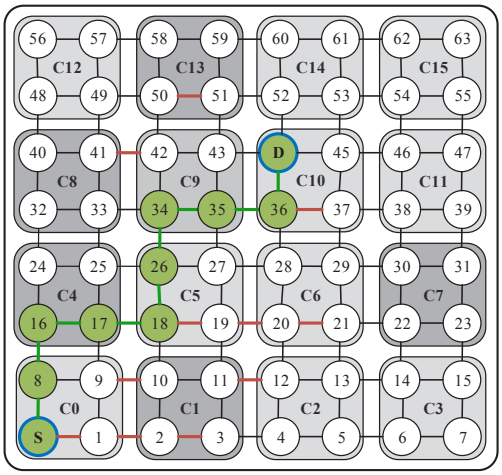
شکل ۵. اطلاعات لینک‌های مورد نیاز خوشه فعلی (Cc)

الگوریتم ۱. بررسی قابلیت اطمینان خوشه‌های مجاور

```
1. if (link (x_dir) = faulty) then
2.        Rx_1=1;
3. else
4.        Rx_1=0;
5. end if;
5. if (link neighbor (x_dir) =faulty) then
6.        Rx_2=1;
7. else
8.        Rx_2=0;
9. end if;
10. R1 = Rx_1+ Rx_2;
11. if (link (y_dir) = faulty) then
12.        Ry_1=1;
13. else
14.        Ry_1=0;
15. end if;
16. if (link neighbor (y_dir) =faulty) then
17.        Ry_2=1;
18. else
19.        Ry_2=0;
20. end if;
21. R2 = Ry_1+ Ry_2;
22. if (R1=0 and R2=0) then
23.
R_X=ProbabilityFault_X×NumberFaultyLink_X
24.
R_Y=ProbabilityFault_Y×NumberFaultyLink_Y
25. else
26.R_X=R1×ProbabilityFault_X×NumberFaultyLink_
X
27.R_Y=R2×ProbabilityFault_Y×NumberFaultyLink_
Y
28. end if;
```

الگوریتم مسیریابی

در این بخش، عملکرد الگوریتم مسیریابی توصیف می‌شود، که چگونه یک بسته از سوئیچ مبدا به سوئیچ مقصد مسیریابی می‌شود. الگوریتم خوشه‌بندی که براساس تقسیم شبکه به مجموعه‌ای از گره‌ها در شرایط خاص است، برای حل مشکلات شبکه استفاده می‌شود. خوشه‌بندی می‌تواند ایستا یا پویا باشد، در خوشه‌بندی ایستا شکل و اندازه‌ی خوشه در زمان اجرا تغییر نمی‌کند ولی در خوشه‌بندی پویا اندازه خوشه در زمان اجرا تغییر می‌کند. در اینجا خوشه‌بندی به طور ایستا انجام می‌شود. به عنوان مثال نمونه‌ای از خوشه‌بندی در شبکه مش ۸×۸ در شکل ۶ نشان داده شده است که در آن شبکه به ۱۶ خوشه تقسیم شده است و در هر خوشه ۴ سوئیچ قرار دارد. وضعیت ترافیک در داخل هر خوشه مشابه است.



شکل ۶. خوشه بندی مش ۸×۸

در این الگوریتم، هر سوئیچ ابتدا سوئیچ مقصد بسته اطلاعاتی را تعیین می‌کند. اگر سوئیچ مقصد در همان خوشه‌ای باشد که سوئیچ فعلی قرارگرفته، از الگوریتم مسیریابی XY استفاده می‌شود زیرا سوئیچ مقصد به سوئیچ فعلی نزدیک است و به الگوریتم مسیریابی پیچیده‌تری نیاز ندارد. در غیر این صورت، اگر سوئیچ مقصد در خوشه دیگری از سوئیچ فعلی قرار گرفته باشد، الگوریتم مسیریابی خوشه همسایه‌ای که دارای کمترین هزینه است را انتخاب می‌کند.

برای مثال فرض کنید که یک بسته در $C0$ به سمت مقصد $C15$ تولید می‌شود. بسته را می‌توان از طریق خوشه $C1$ یا $C4$ ارسال کرد. این تصمیم براساس مقادیر تاخیر ذخیره شده در جدول CQ خوشه $C0$ و همچنین وضعیت لینک‌های خوشه‌های مجاور گرفته می‌شود. فرض می‌کنیم که خوشه $C1$ مقدار تاخیر زمانی پایین‌تری دارد (رنگ خاکستری تیره یعنی ازدحام بیشتر) ولی از خوشه $C0$ به خوشه $C1$ مسیری برای رسیدن به خوشه مقصد وجود ندارد. خوشه دیگر مجاور خوشه فعلی $C4$ است که خوشه $C4$ تاخیر زمانی بیشتری نسبت به خوشه $C1$ دارد ولی لینک‌های آن به سمت مقصد سالم هستند بنابراین مطابق رابطه ۴ هزینه ارسال بسته از طریق خوشه $C4$ کمتر است. با ارسال بسته به خوشه همسایه $C4$ ، بسته‌ی داده که شامل تاخیر تخمینی جدیدی است با استفاده از رابطه ۲ به‌روز می‌شود. برای خروج از خوشه $C4$ خوشه $C5$ هزینه‌ی کمتری دارد بنابراین بسته وارد خوشه $C5$ می‌شود. به محض خروج بسته از خوشه $C4$ یک بسته یادگیری در خوشه C4 تولید شده و به خوشه $C0$ تحویل داده می‌شود. با دریافت بسته یادگیری در خوشه $C0$ ، مقدار قدیمی تاخیر در جدول CQ (با استفاده از رابطه ۳) با مقدار جدید ترکیب و به‌روز می‌شود.

مشابه الگوریتم مسیریابی Q الگوریتم پیشنهادی سه مرحله دارد: ۱. هنگامی که بسته داده به سوئیچ تحویل داده می‌شود، ۲. زمانی که سوئیچ بسته داده دریافت می‌کند و ۳. هنگامی که سوئیچ یک بسته یادگیری دریافت می‌کند.

عملکرد مرحله اول در الگوریتم۲ شرح داده شده است. خوشه‌ای با کمترین هزینه مطابق رابطه (۴) براساس قابلیت اطمینان و وضعیت ازدحام خوشه‌های مجاور برای رسیدن به خوشه مقصد تعیین می‌شود. (خطوط ۱۲-۱). سپس مجموع تعداد اسلات‌های اشغالی بافرهای ورودی در پورت خروجی را بدست می‌آورد و سپس بسته را به سوئیچ بعدی می‌فرستد و تا زمانی که از خوشه خارج نشده است این کار را ادامه می‌دهد. (خطوط ۱۷-۱۳). با خروج از خوشه فعلی، بسته داده به خوشه‌ی مجاور تحویل داده می‌شود. بسته داده باید تاخیر جدید تخمین زده شده که متشکل از تاخیرهای محلی و سراسری است را بدست آورد. تاخیر محلی با تقسیم مجموع اسلات‌های بافر اشغالی به تعداد گام‌های طی شده توسط یک بسته در داخل خوشه فعلی بدست می‌آید. تاخیر سراسری از جدول CQ خوشه فعلی بدست می‌آید؛ این مقدار تاخیر تخمینی بسته از خوشه فعلی به خوشه مبدا را نشان می‌دهد. این مقادیر برای به‌روزرسانی ورودی متناظر از جدول CQ در خوشه بعدی استفاده می‌شوند. ردیف جدول با شناسه خوشه مبدا تعیین می‌شود و ستون جهتی است که در آن بسته داده تحویل داده شده است. (خطوط ۲۳-۱۸).

الگوریتم۲. مرحله اول

Inputs: Current cluster: Cc, destination Cluster: Cd;
Output: Selected Cn ;

Procedure:
1. **–Determining the neighboring cluster (Cn) with minimum Cost**
2. **if** ((Ry_1=1 & Rx_1=0) | (Rx_1=0 & Rx_2==1 & cluster_dirX = Cd)) **then**
3. Cost_X=0;
4. Cost_Y=1;
5. **else if** ((Rx_1=1 & Ry_1=0) | (Ry_1=0 & Ry_2=1 & cluster_dirY = Cd)) **then**
6. Cost_X=1;
7. Cost_Y=0;
8. **else**
9. Cost_X= 0.8×R_X + 0.2×Latency_X;
10. Cost_Y= 0.8×R_Y + 0.2×Latency_Y;
11. **end if**
12. Cn = min (Cost_X, Cost_Y);
13. **–Measuring total occupied buffer slots in input buffers**
When the packet traveling inside cluster Cc
14. **Do**
15. BufferSizes←OccupiedBufferSlotsinOutputBuffer +BufferSizes;
16. Forward packet to the neighboring switch;
17. **Until** (packet is going to leave the cluster Cc);
18.**–Upon leaving the packet, packet send to neighboring cluster**
19.LocalLatencyData←BufferSizesData/NumberOfHop;
20. GlobalLatencyData ← $Q_{Cc}(C_m,C_s)$
$//Q_{Cc}(Cm,Cs) = \min Q_{Cc}(Cn,Cs)$
$//Cn \in neighboringCc$
21. NewEstimatedLatencyData ← LocalLatencyData + GlobalLatencyData;
22. RecivingCluster_ID ← Cn ;
23. SourceCluster_ID ← Cs ;

در الگوریتم ۳ شبه کد دومین مرحله مسیریابی یعنی زمانی‌که سوئیچ بسته‌ی داده‌ای را دریافت می‌کند آورده شده است. اطلاعات بسته از هدر بسته، شامل تاخیر تخمین زده جدید، اطلاعات سطر و ستون (خطوط ۱۱-۱) خارج می‌شود. سپس جدول CQ با استفاده از مقادیر برآورد قدیمی و جدید (خطوط ۱۵-۱۲) به‌روزرسانی می‌شود (از رابطه ۲ برای به‌روز کردن جدول مسیریابی استفاده می‌شود). همزمان، یک خوشه همسایه با کمترین هزینه (خطوط ۱۷-۱۶) انتخاب می‌شود (مشابه شبه کد قبلی). با پیمایش بسته در همان خوشه، مجموع تعداد اسلات‌های اشغالی بافر ورودی در پورت ورودی (خطوط ۲۲-۱۸) بدست می‌آید. با خروج از خوشه فعلی، بسته یادگیری تولید شده و به خوشه بالادستی تحویل داده می‌شود. بسته یادگیری باید تاخیر جدید تخمین زده شده که متشکل از تاخیرهای محلی و سراسری است را بدست آورد. تاخیر سراسری از جدول CQ خوشه فعلی بدست می‌آید؛ این مقدار تاخیر تخمینی بسته را در مسیر باقی مانده از خوشه بعدی به خوشه مقصد Cd نشان می‌دهد. این مقادیر برای به‌روزرسانی ورودی متناظر از جدول CQ در خوشه قبلی استفاده می‌شوند. ردیف جدول با شناسه خوشه مقصد تعیین می‌شود و ستون جهتی است که در آن بسته داده تحویل داده شده است. (خطوط ۲۸-۲۳).

در مرحله سوم، زمانی که سوئیچ بسته یادگیری را دریافت می‌کند، جدول CQ با استفاده از مقادیر قدیمی و جدید به‌روزرسانی می‌شود (از رابطه (۳) برای به‌روز کردن جدول مسیریابی استفاده می‌شود).

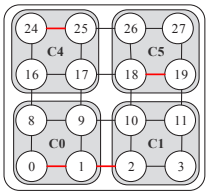
الگوریتم ۳. مرحله دوم

1. **–Extracting information from data packet**
2. Cu←UpstreamCluster_ID;
3. BufferSizes← BufferSizes;
4. $Q_{estData}$ ← NewEstimatedLatencyData;
5. Column ←direction (ReceivingCluster_ID);
6. Row ←sourceCluster_ID;
7. **if** (yReceivingCluster-yCu! = 0) **then**
8. Column← yDir;
9. **else**
10. Column ← xDir;
11. **end if;**
12. **–Update the corresponding entry of the CQ-table using Eq. (2) as follow:**
13. Q_{old} ←CQ_table (row,column);
14. $Q_{newData}$← Q_{old}+ 0.5 ($Q_{estData}$-Q_{old});
15. CQ_table (row,column)← $Q_{newData}$;
16. **–Determining the neighboring cluster (Cn) with minimum Cost Similar previous Pseudo code**
17. Cn = min (Cost_X, Cost_Y);
18. **–Measuring total occupied buffer slots in input buffers**
When the packet traveling inside cluster Cc
19. **Do**
20. BufferSizes←OccupiedBufferSlotsinInputBuffer +BufferSizes;
21. Forward packet to the neighboring switch;
22. **Until** (packet is going to leave the cluster Cc);
23. **–Upon leaving the packet, generate learning packet and send it back to upstream cluster Cu**
24. LocalLatency ← BufferSizes /NumberOfHops;
25. GlobalLatency← $Q_{Cc}(Cn,Cd)$
$//Q_{Cc}(Cn,Cd) = \min Q_{Cc}(Cn,Cd)$
$//Cn \in neighboringCc$
26.NewEstimatedLatency←LocalLatency+GlobalLatency;
27. RecivingCluster_ID ← Cu ;
28. DestinationCluster_ID ← Cd ;

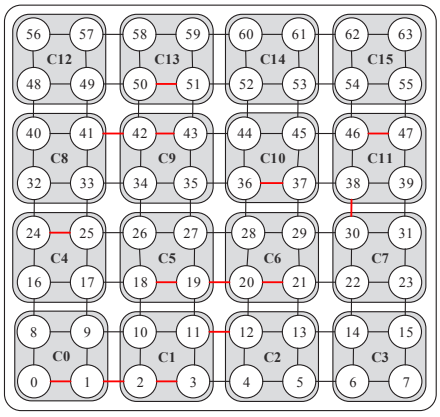
شبیه‌سازی و توصیف نتایج

برای ارزیابی کارایی و قابلیت اطمینان الگوریتم مسیریابی پیشنهادی، از شبیه‌ساز شبکه‌های روی تراشه ناهمگن (HNOCS) [۲۸] استفاده شده است، HNOCS مبتنی‌بر OMNET++ [۲۹] است. الگوریتم مورد مقایسه با الگوریتم پیشنهادی، Bi-LCQ [۱۵]، است نتایج شبیه‌سازی در شرایطی بدست آمده که در شبکه تعدادی از لینک‌ها معیوب هستند. در شرایطی که خطایی در شبکه وجود نداشته باشد الگوریتم پیشنهادی نتایج مشابهی با

روش مقایسه شده دارد. ترافیک یکنواخت[۱۵]و نقطه داغ[۱۶]بر دو ساختار مش دو بعدی ۴×۴ و ۸×۸ اعمال شده است. مکان و تعداد لینک‌های خراب برای مش ۴×۴ و ۸×۸ در شکل‌های ۷ و ۸ آورده شده است؛ لینک‌های معیوب با قرمز مشخص شدند.



شکل ۷. تعداد و موقعیت لینک‌های خراب در مش ۴×۴



شکل ۸. تعداد و موقعیت لینک‌های خراب در مش ۸×۸

در هر مرحله از زمان شبیه‌سازی (نانو ثانیه)، بسته‌ها در بافر FIFO سوئیچ منبع ذخیره می‌شوند. مقدار بسته تزریق شده بستگی به بار ارائه شده در شبکه دارد. از معادله زیر برای محاسبه بار ارائه شده به شبکه استفاده می‌شود.

$$Networkofferedload = \frac{flit\_size}{flit\_arrival\_delay} \qquad (۵)$$

$flit\_size$ نشان دهنده اندازه فلیت است. $flit\_arrival\_delay$ نشان دهنده تاخیر زمانی بین تولید فلیت قبلی و بعدی است. جدول ۱ تنظیمات عمومی پارامترهای ابزار شبیه‌سازی را نشان می‌دهد. از آن‌جایی که بسته یادگیری فقط بین سوئیچ‌های همسایه وجود دارد، تأخیر آن در مقایسه با بسته‌های داده که از مبدا به مقصد حرکت می‌کنند بسیار کوچک است. به همین دلیل، در نتایج شبیه‌سازی، تاخیر فقط برای بسته‌های داده محاسبه شده است.

جدول ۱. تنظیمات پارامترهای شبیه‌سازی

پارامترها	تنظیمات
طول پیام	۴ بسته
طول بسته	۸ فلیت
طول فلیت	۳۲ بیت
اندازه بافر ورودی	۸ فلیت
تعداد کانال‌های مجازی	۲
سوئیچینگ	خزشی[۱۷]
فرکانس کاری تراشه	۰٫۵ گیگاهرتز
همبندی	مش ۴×۴ و۸×۸

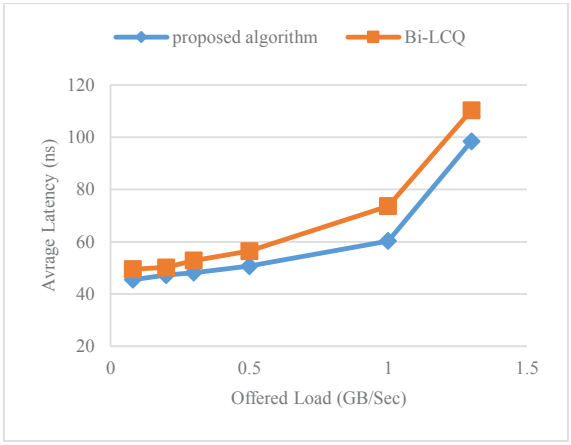
ارزیابی میانگین تاخیر بسته‌ها

میانگین تاخیر بسته یکی از مهم‌ترین پارامترهایی است که کارایی الگوریتم مسیریابی را مشخص می‌کند. روند افزایش میانگین تاخیر بسته زمانی که بار شبکه را افزایش می‌دهیم به وسیله یک منحنی در شکل‌های ۹ و ۱۰ نشان داده شده است. الگوریتم پیشنهادی و Bi-LCQ بر حسب میانگین تاخیر بسته، تحت الگوهای ترافیک تصادفی یکنواخت و نقطه داغ، برای شبکه‌های با اندازه ۴×۴ و ۸×۸ با یکدیگر مقایسه شده‌اند.

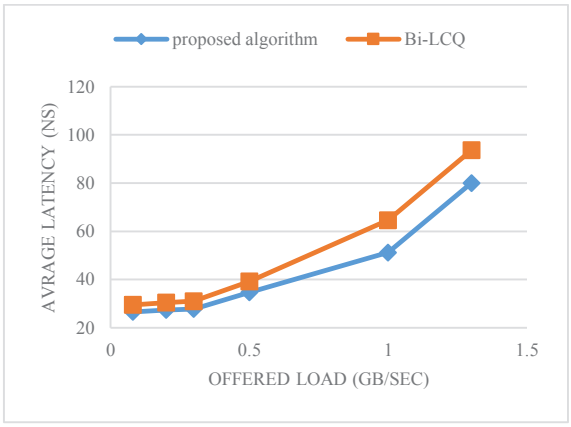
رایج‌ترین الگوی ترافیکی که در اکثر ارزیابی‌ها مورد استفاده قرار می‌گیرد، الگوی ترافیکی یکنواخت است. در مدل ترافیک یکنواخت هر عنصر پردازشی می‌تواند بسته خود را با احتمال مساوی به هر عنصر پردازشی دیگر ارسال کند. به عبارت دیگر در این مدل احتمال اینکه مقصد یک بسته یک گره خاص باشد، با سایر گره‌ها برابر است. نتایج حاصل از بررسی حداکثر میزان تاخیر در الگوریتم مسیریابی مذکور تحت این الگوی ترافیکی، در شکل ۹ قابل مشاهده است.

در نمودار شکل ۹ محور افقی نشان دهنده بار ارائه شده در شبکه و محور عمودی، میانگین تاخیر می باشد. از مقایسه فوق این نکته برداشت می‌شود که میزان تاخیر برای الگوریتم مسیریابی جدید، در صورت وجود لینک‌های معیوب کمتر از الگوریتم Bi-LCQ است. با توجه به شکل با افزایش بار الگوریتم پیشنهادی عملکرد بهتری دارد. در الگوریتم ارائه شده به دلیل اینکه مسیریاب از وضعیت لینک‌ها و ازدحام در یک منطقه آگاه است ناحیه‌ای که شرایط بهتری دارد را برای ادامه مسیریابی انتخاب می‌کند. در الگوریتم Bi-LCQ فقط وضعیت ازدحام را در مناطق مجاور بررسی می‌شود.

[۱۵] Uniform random traffic
[۱۶] Hotspot traffic
[۱۷] Wormhole (WH)



الف) شبکه ۸×۸

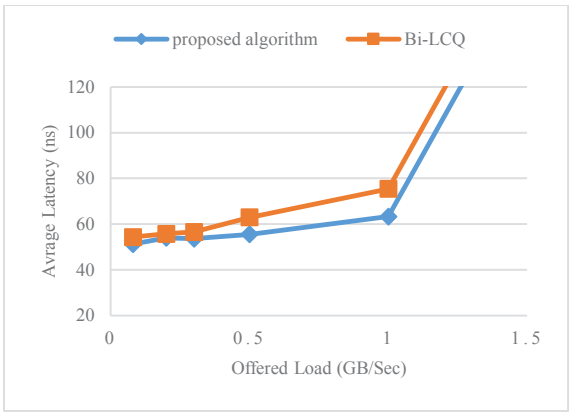


ب) شبکه ۴×۴

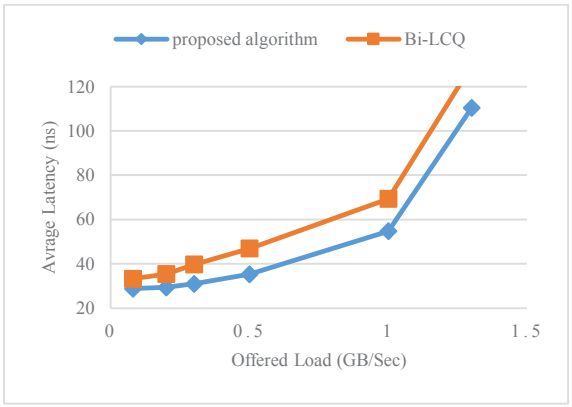
شکل ۹. میانگین تاخیر بسته تحت ترافیک یکنواخت در الف) شبکه ۸×۸ ب) شبکه ۴×۴

یکی دیگر از الگوهای ترافیکی که بسیار مورد توجه است، الگوی ترافیکی نقطه داغ می‌باشد. در این الگو یک یا چند گره به عنوان نقطه داغ درنظر گرفته می‌شوند و بسته‌های بیشتری را دریافت می‌کنند. زمانی که بسته جدیدی تولید می‌شود با احتمال H به یک گره مشخص ارسال می‌شود، بنابراین گرهی که به عنوان نقطه داغ درنظر گرفته شده است با احتمال H بسته‌های بیشتری دریافت می‌کند و بقیه گره‌ها به صورت تصادفی مانند ترافیک یکنواخت بسته دریافت می‌کنند. بنابراین ترافیک اطراف یک گره بسیار سنگین خواهد شد که همین امر موجب کاهش کارایی شبکه می‌شود. با در نظر گرفتن سوئیچ (۴،۴) و (۲،۲) به عنوان hotspot به ترتیب در مش ۸×۸ و ۴×۴ شبیه‌سازی انجام شده است. میانگین تاخیر هر شبکه با H=10%، در شکل ۱۰ قابل مشاهده است.

همانطور که در شکل ۱۰، مشاهده می‌شود طرح مسیریابی پیشنهادی عملکرد بهتری در مقایسه با طرح دیگر دارد. با استفاده از مسیریابی کمینه همراه با سیاست مسیریابی هوشمند به طور متوسط زمان تاخیر شبکه کاهش می‌یابد.



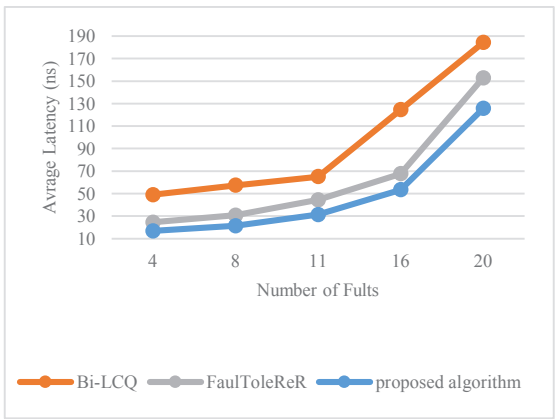
الف) شبکه ۸×۸



ب) شبکه ۴×۴

شکل ۱۰. میانگین تاخیر بسته تحت ترافیک نقطه داغ در الف) در شبکه ۸×۸. ب) شبکه ۴×۴

در شکل ۱۱ میانگین زمان تأخیر الگوریتم پیشنهادی با الگوریتم‌های مسیریابی FaulToleReR [۲۳] و Bi-LCQ برای تعداد مختلف لینک‌های خطا مقایسه شده است. همانطور که در شکل ۱۱ نشان داده شده است، تاخیر تمام الگوریتم‌های مسیریابی با افزایش تعداد لینک‌های معیوب افزایش می‌یابد. با این حال، الگوریتم پیشنهادی دارای تاخیر کمتری نسبت به سایر طرح‌های مسیریابی در نظر گرفته شده است؛ زیرا همیشه بهترین مسیر را براساس لینک‌های معیوب و ازدحام در شبکه پیدا می‌کند.

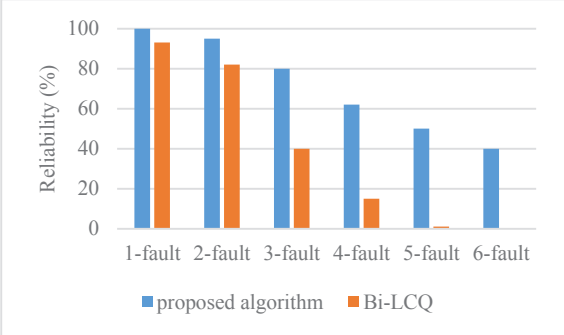


شکل ۱۱. میانگین تاخیر بسته تحت ترافیک یکنواخت در مش ۸×۸.

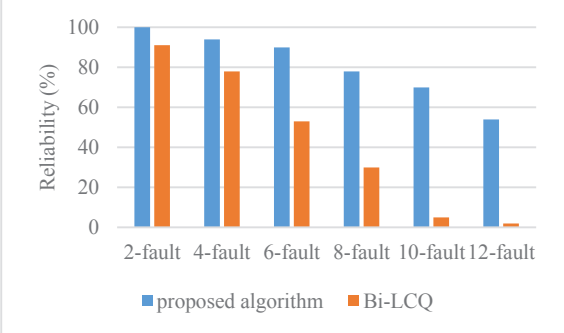
نمودارهای نشان داده شده، بهبودی الگوریتم مسیریابی پیشنهادی را نسبت به الگوریتم Bi-LCQ از نظر میانگین تاخیر بسته نشان می‌دهد. الگوریتم پیشنهادی دارای منطق ساده‌ای است و سعی می‌کند کوتاه‌ترین مسیر امن و با ازدحام کمتر به سمت مقصد را بیابد.

ارزیابی قابلیت اطمینان تحت ترافیک یکنواخت

برای اینکه قابلیت اطمینان اندازه‌گیری شود، تعداد لینک‌های خراب افزایش می‌یابد. لینک‌های خراب به صورت تصادفی به شبکه اعمال شده اما حداکثر یکی از خطاها در لینک‌های بین خوشه‌ها است تا عملکرد روش پیشنهادی در برابر لینک‌های خراب بهتر نشان داده شود. نسبت تعداد بسته‌هایی که به مقصد رسیده‌اند به تعداد کل بسته‌ها برحسب درصد نشان دهنده قابلیت اطمینان است.



الف) مش ۴×۴



ب) مش ۸×۸

شکل ۱۲. مقایسه قابلیت اطمینان الگوریتم‌های مسیریابی تحت ترافیک یکنواخت در الف) مش ۴×۴ ب) مش ۸×۸

همان‌طور که در شکل ۱۲ مشاهده می‌شود، الگوریتم پیشنهادی قابلیت اطمینان بیشتری نسبت به Bi-LCQ دارد و هرچه تعداد خطا بیشتر می‌شود این اختلاف افزایش می‌یابد. به دلیل این که می‌تواند بسته‌ها را به منطقه‌ای هدایت کند که تعداد لینک‌های خطای آن منطقه کمتر است. در واقع در مسیریابی پیشنهادی، یک مصالحه بین ازدحام و قابلیت اطمینان لینک‌های منطقه مجاور برقرار است.

نتیجه‌گیری

در این مقاله، یک الگوریتم مسیریابی آگاه از ازدحام بر اساس روش یادگیری Q ارائه شده است. در الگوریتم مسیریابی ارائه شده شبکه به چند خوشه تقسیم می‌شود و هر خوشه یک جدول CQ را نگه می‌دارد، این جدول اطلاعات ازدحام محلی و سراسری را ذخیره می‌کند. در مرحله اول خوشه فعلی براساس اطلاعات جدول CQ وضعیت ازدحام در خوشه‌های مجاور را بررسی می‌کند، سپس لینک‌های خوشه‌های مجاور به سمت مقصد بررسی می‌شود و در نهایت با توجه به وضعیت ازدحام و قابلیت اطمینان خوشه‌ها، کانال خروجی مناسب انتخاب می‌شود. به منظور به روز رسانی جدول‌های CQ از هر دو بسته یادگیری و داده در انتشار اطلاعات ازدحام استفاده شده است. با استفاده از روش پیشنهادی، تعداد بسته‌های از دست رفته کاهش می‌یابد و همچنین بسته‌ها از مسیری با ازدحام کمتر عبور می‌کنند.

مراجع

[1] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 28, no. 1, pp. 3-21, 2009.

[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, 2001, pp. 684-689.

[3] A. Jantsch and H. Tenhunen, *Networks on chip*. Springer, 2003.

[4] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, 2008, pp. 203-214: IEEE.

[5] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proceedings of the 43rd annual Design Automation Conference*, 2006, pp. 849-852: ACM.

[6] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip," *IEEE Transactions on Computers,* vol. 57, no. 6, pp. 809-820, 2008.

[7] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Agent-based on-chip network using efficient selection method," in *VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on*, 2011, pp. 284-289: IEEE.

[8] S. Ma, N. Enright Jerger, and Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *ACM SIGARCH Computer Architecture News*, 2011, vol. 39, no. 3, pp. 413-424: ACM.

[9] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning,* vol. 8, no. 3-4, pp. 279-292, 1992.

[10] S. P. Choi and D.-Y. Yeung, "Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control," in *Advances in Neural Information Processing Systems*, 1996, pp. 945-951.

[11] S. Kumar and R. Miikkulainen, "Dual reinforcement Q-routing: An on-line adaptive routing algorithm," in *Proceedings of the artificial neural networks in engineering Conference*, 1997, pp. 231-238.

[12] S. Kumar and R. Miikkulainen, "Confidence based dual reinforcement q-routing: An adaptive online network routing algorithm," in *IJCAI*, 1999, vol. 99, pp. 758-763: Citeseer.

[13] M. K. Puthal, V. Singh, M. S. Gaur, and V. Laxmi, "C-Routing: An adaptive hierarchical NoC routing methodology," in *VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on*, 2011, pp. 392-397: IEEE.

[14] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila, and P. Liljeberg, "Optimized Q-learning model for distributing traffic in on-Chip Networks," in *Networked Embedded Systems for Every Application (NESEA), 2012 IEEE 3rd International Conference on*, 2012, pp. 1-8: IEEE.

[15] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Bi-LCQ: A low-weight clustering-based Q-learning approach for NoCs," *Microprocessors and Microsystems,* vol. 38, no. 1, pp. 64-75, 2014.

[16] آ. امیدوار. ک. محمدی, "الگوریتم مسیریابی با افزایش قابلیت اطمینان در شبکه‌های تحمل‌پذیر تاخیر," فصلنامه صنایع الکترونیک, vol. 5, no. 1, ۱۳۹۳.

[17] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das, "Design and analysis of an NoC architecture from performance, reliability and energy perspective," in *Proceedings of the 2005 ACM symposium on Architecture for networking and communications systems*, 2005, pp. 173-182: ACM.

[18] M. Ebrahimi, M. Daneshtalab, J. Plosila, and F. Mehdipour, "MD: minimal path-based fault-tolerant routing in on-chip networks," in *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, 2013, pp. 35-40: IEEE.

[19] J.-x. Wang, F.-f. Fu, T.-S. Zhang, and Y.-P. Chen, "A small-granularity solution on fault-tolerant in 2D-mesh network-on-chip," in *Solid-State and Integrated Circuit Technology (ICSICT), 2010 10th IEEE International Conference on*, 2010, pp. 382-384: IEEE.

[20] Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, 2008, pp. 441-446: IEEE.

[21] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant NoCs," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2009, pp. 21-26: European Design and Automation Association.

[22] M. Valinataj, S. Mohammadi, J. Plosila, P. Liljeberg, and H. Tenhunen, "A reconfigurable and adaptive routing method for fault-tolerant mesh-based networks-on-chip," *AEU-International Journal of Electronics and Communications,* vol. 65, no. 7, pp. 630-640, 2011.

[23] R. J. Behrouz, M. Modarressi, and H. S. Azad, "A Reconfigurable Fault-Tolerant Routing Algorithm to Optimize the Network-on-Chip Performance and Latency in Presence of Intermittent and Permanent Faults," in *2011 IEEE 29th International Conference on Computer Design (ICCD)*, 2011, pp. 433-434: IEEE.

[24] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[25] E. Wachter, V. Fochi, F. Barreto, A. Amory, and F. Moraes, "A Hierarchical and Distributed Fault Tolerant Proposal for NoC-based MPSoCs," *IEEE Transactions on Emerging Topics in Computing,* pp. 1-1, 2018.

[26] A. B. Gabis, P. Bomel, and M. Sevaux, "Application-aware Multi-Objective Routing based on Genetic Algorithm for 2D Network-on-Chip," *Microprocessors and Microsystems,* 2018.

[27] C. Wu *et al.*, "An efficient application mapping approach for the co-optimization of reliability, energy, and performance in reconfigurable NoC architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 34, no. 8, pp. 1264-1277, 2015.

[28] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny, "HNOCS: modular open-source simulator for heterogeneous NoCs," in *Embedded Computer Systems (SAMOS), 2012 International Conference on*, 2012, pp. 51-57: IEEE.

[29] A. Varga, "Using the OMNeT++ discrete event simulation system in education," *IEEE Transactions on Education,* vol. 42, no. 4, p. 11 pp., 1999.