مقاله پژوهشی

توازن بار در گره‌های مه با استفاده از الگوریتم یادگیری تقویتی

نیلوفر طهماسبی‌پویا و مهدی‌آقا صرام

چکیده: محاسبات مه، حوزه تحقیقاتی نوظهوری برای ارائه خدمات محاسبات ابری به لبه‌های شبکه است. گره‌های مه جریان داده و درخواست‌های کاربر را در زمان واقعی پردازش می‌کنند. به منظور بهینه‌سازی بهره‌وری منابع و زمان پاسخ و افزایش سرعت و کارایی، وظایف باید به صورت متوازن بین گره‌های مه توزیع شوند، لذا در این مقاله، روشی جدید جهت بهبود توازن بار در محیط محاسبات مه پیشنهاد شده است. در الگوریتم پیشنهادی، هنگامی که وظیفه‌ای از طریق دستگاه‌های موبایل برای گره مه ارسال می‌شود، گره مه با استفاده از یادگیری تقویتی تصمیم می‌گیرد که آن وظیفه را خودش پردازش کند، یا این که پردازش آن را به یکی از گره‌های مه همسایه یا به ابر واگذار نماید. در بخش ارزیابی نشان داده شده که الگوریتم پیشنهادی با توزیع مناسب وظایف بین گره‌ها، تأخیر کمتری را برای اجرای وظایف نسبت به سایر روش‌های مقایسه‌شده به دست آورده است.

***کلیدواژه:* تأخیر، توازن بار، گره مه، یادگیری تقویتی، Q-Learning.**

۱- مقدمه

محاسبات مه[1]الگوی محاسباتی توزیع‌شده‌ای است که خدمات ارائه‌شده توسط ابر[2]را به لبه[3]شبکه گسترش می‌دهد تا مدیریت و برنامه‌ریزی خدمات محاسباتی، شبکه‌ای و ذخیره‌سازی بین مراکز داده و دستگاه‌های نهایی را تسهیل کند. گره‌های مه (FN[4]) با قرارگرفتن در لایه بین منابع داده و مرکز داده ابر، جریان داده و درخواست‌های کاربر را در زمان واقعی پردازش می‌کنند و تأخیر و ازدحام شبکه را کاهش می‌دهند [۱] و [۲]. دستگاه‌های اینترنت اشیا[5]معمولاً پردازش وظایف را به نزدیک‌ترین گره مه واگذار می‌کنند. در این صورت ممکن است که بعضی از گره‌های مه وظایف بیشتری را نسبت به سایر گره‌های مه دریافت کنند و به مرور دچار سربار[6]شوند. برای جلوگیری از این وضعیت، از روش‌های توازن بار[7]برای توزیع یکنواخت وظایف در میان گره‌های مه استفاده می‌شود. گره‌های مه در محیط توزیع‌شده‌ای مستقر شده‌اند. توازن بار در محیط‌های توزیع‌شده به دو دسته توازن بار ایستا و پویا تقسیم می‌شود. در توازن بار ایستا هنگام تصمیم‌گیری برای توازن بار، وضعیت گره مه مقصد در نظر گرفته نمی‌شود. توازن بار ایستا یا به صورت قطعی انجام می‌شود و وظایف به طور دائم به گره مه خاصی تخصیص داده می‌شوند و یا به صورت احتمالی انجام می‌شود که در آن وظایف با احتمال مشخصی به هر گره مه اختصاص داده می‌شوند. اما در توازن بار پویا، وضعیت فعلی گره‌های مه در طول تصمیمات توازن بار در نظر گرفته می‌شود و با توجه به وضعیت آنها، گره مه مقصد انتخاب می‌شود [۳]. در این مقاله از روش توازن بار پویا استفاده شده است.

توازن بار بین گره‌های مه نقش مهمی در کاهش تأخیر، زمان پاسخ به کاربر و هزینه دارد و همچنین باعث افزایش بهره‌وری منابع، کارایی، تشخیص رویدادها در زمان واقعی و صرفه‌جویی در مصرف منابع می‌شود. در دهه‌های گذشته، روش‌های مختلفی برای توازن بار ارائه شده که هر یک در وضعیت خاصی از سیستم، نتیجه مناسبی دارند. الگوریتم‌های توازن بار را می‌توان بر اساس نوع الگوریتم مورد استفاده به سه نوع مختلف ابتکاری، فراابتکاری و ترکیبی طبقه‌بندی کرد. انواع الگوریتم‌های ابتکاری به شکل ایستا یا پویا وجود دارند. ابتکاری‌ها شامل محدودیت‌هایی هستند که قصد یافتن بهترین راه حل ممکن برای یک مسأله خاص را دارند. این الگوریتم‌ها به دلیل یافتن راه حل رضایت‌بخش، مفید و در مقایسه با الگوریتم‌های فراابتکاری به آسانی قابل اجرا هستند [۴] تا [۶]. الگوریتم‌های فراابتکاری ماهیتی پویا دارند و نسبت به الگوریتم‌های ابتکاری، به زمان بیشتری برای اجرا و حل‌کردن راه حل نهایی نیاز دارند، زیرا فضای راه حل آنها می‌تواند بسیار بزرگ باشد و علاوه بر این، فراابتکاری‌ها فرایندهای کاملاً تصادفی هستند. زمان و راه حل آنها تا حد زیادی به ماهیت مسأله، پیکربندی اولیه و روش جستجوی راه حل بستگی دارد [۷] تا [۹]. الگوریتم‌های ترکیبی، ترکیبی از الگوریتم‌های ابتکاری یا فراابتکاری مختلف هستند که زمان اجرا و نیز هزینه را کاهش می‌دهند و نتیجه مؤثرتری نسبت به الگوریتم‌های دیگر ارائه می‌دهند [۱۰] تا [۱۲].

با این حال، الگوریتم‌های توازن بار رایج به دلیل ماهیت توزیع‌شده و پویایی محاسبات مه، کارایی خود را از دست می‌دهند و به الگوریتمی نیاز است که بتواند در طول زمان با شرایط محیطی تطبیق یابد. به همین منظور در این مقاله، فرایند تصمیم‌گیری مبتنی بر یادگیری تقویتی[8]برای یافتن گره مه کم‌بار پیشنهاد شده است. روش پیشنهادی ارائه‌شده به گره مه اجازه می‌دهد تا وظیفه ورودی را با انتخاب یک گره مه همسایه در دسترس، با توجه به ظرفیت منابع، با هدف به حداقل رساندن زمان پردازش و احتمال سربار کلی، واگذار کند. طبق بررسی‌های انجام‌شده در اکثر روش‌های ارائه‌شده برای برقراری توازن بار در محیط مه، برای واگذاری وظایف به گره‌های همسایه، ابتدا باید یک سری محاسبات

این مقاله در تاریخ ۱۴ تیر ماه ۱۴۰۰ دریافت و در تاریخ ۳ خرداد ماه ۱۴۰۱ بازنگری شد.

نیلوفر طهماسبی‌پویا، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران، (email: niloofartahmasebi@stu.yazd.ac.ir).

مهدی‌آقا صرام (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران، (email: mehdi.sarram@yazd.ac.ir).

1. Fog Computing
2. Cloud
3. Edge
4. Fog Nodes
5. Internet of Things
6. Overload
7. Load Balancing
8. Reinforcement Learning

زمان‌بر برای آگاهی از ظرفیت و موقعیت گره‌های همسایه انجام شود که این امر نیز باعث ایجاد تأخیر در شبکه می‌شود. با این حال در روش پیشنهادی این مقاله، محاسبات اضافی و زمان‌بر حذف می‌شوند و عامل، تنها طبق تجربیاتی که از محیط کسب می‌کند در هر حالت، گره مه مناسب را انتخاب می‌کند و این امر باعث می‌شود که در روش پیشنهادی، تأخیر به شکل قابل توجهی کاهش یابد. در مقایسه با روش‌های سنتی، استفاده از یادگیری تقویتی برای توازن بار، نه تنها بدون در نظر گرفتن هیچ فرضی از مدل شبکه، چارچوب الگوریتم را ساده می‌کند بلکه با پیچیدگی زمانی چندجمله‌ای به سیاست بهینه همگرا می‌شود. نتایج به دست آمده نشان می‌دهند که روش توازن بار پیشنهادی، تأخیر و زمان پاسخ به کاربر را نسبت به روش‌های مقایسه‌شده به شکل قابل توجهی کاهش می‌دهد.

ادامه این مقاله به این صورت سازماندهی شده که در بخش ۲ کارهای مرتبط اخیر در زمینه توازن بار در محاسبات مه مورد بررسی قرار می‌گیرد. در بخش ۳ مدل سیستم پیشنهادی، الگوریتم یادگیری تقویتی و نحوه محاسبه تأخیر در سیستم پیشنهادی شرح داده می‌شوند. در بخش ۴ روش توازن بار پیشنهادی مطرح می‌شود. در بخش ۵ به ارزیابی نتایج حاصل از شبیه‌سازی و مقایسه با روش‌های پیشین پرداخته می‌شود. در نهایت در بخش ۶ نتیجه‌گیری و پیشنهادهای کارهای آتی ارائه خواهد شد.

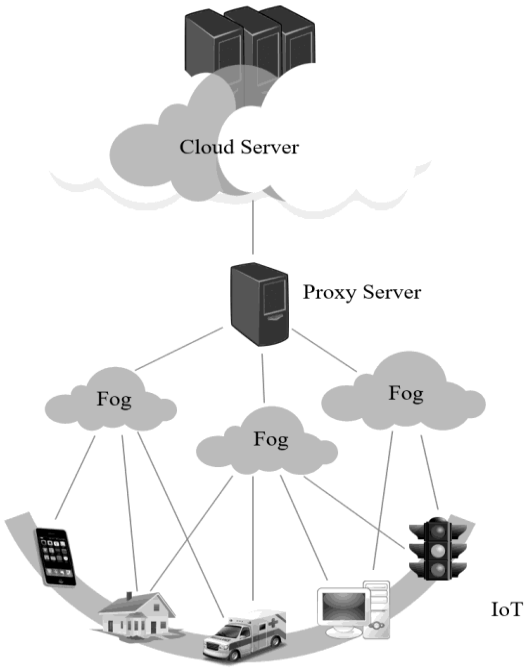
۲- کارهای پیشین

در محاسبات مه، دستگاه‌های اینترنت اشیا و کاربران موبایل به طور معمول وظایف را به نزدیک‌ترین گره مه واگذار می‌کنند. از آنجایی که این دستگاه‌ها اغلب متحرک هستند، ممکن است گره‌های مه مختلف بسته به موقعیتی که در شبکه دارند، دارای بارهای متفاوتی باشند. این مسأله موجب عدم توازن در توزیع وظایف بین گره‌های مه می‌شود و ممکن است برخی از گره‌های مه دچار سربار شوند، در حالی که دیگر گره‌های مه بیکار یا کم‌بار هستند. برخی از نویسندگان روش‌هایی را برای رسیدگی به مسأله توازن بار در محیط محاسبات مه ارائه کرده‌اند. این کارها را می‌توان از جنبه‌های مختلفی دسته‌بندی کرد.

۲-۱ بررسی کارهای پیشین

در اینجا کارهای پیشینی که در آنها برای واگذاری وظایف، گره‌ها نیاز به اطلاع از ظرفیت محاسباتی یکدیگر دارند، بررسی خواهند شد. بیک و همکاران [۱۳]، فرایند تصمیم‌گیری مبتنی بر یادگیری تقویتی را برای یافتن تصمیم بارگذاری بهینه با شرایط مجهول‌بودن مدل پاداش و احتمال گذار پیشنهاد دادند. در فرایند ارائه‌شده، گره‌های مه می‌توانند مقدار بهینه‌ای از وظایف ورودی را به یک گره مه همسایه در دسترس، با توجه به ظرفیت منابع واگذار کنند. هدف از این کار، به حداقل رساندن زمان پردازش و احتمال سربار است. سو و همکاران [۱۴]، روش تخصیص منابع پویا را برای توازن بار در محیط مه پیشنهاد دادند. آنها ابتدا از نظر فنی، چارچوب سیستم محاسبات مه را ارائه دادند و به بررسی عملیات توازن بار روی انواع مختلفی از گره‌های محاسباتی پرداختند. سپس روش تخصیص منابع متناظر را در محیط مه با تخصیص ایستای منابع و انتقال پویای خدمات برای دستیابی به توازن بار در سیستم‌های محاسبات مه طراحی کردند. مانجو و سوماتی [۱۵]، الگوریتم min-min را با در نظر گرفتن منابع شبکه پیشنهاد دادند و آن را در داخل هر خوشه اجرا کردند. هدف از اجرای این الگوریتم، توزیع یکنواخت بار در بین گره‌های مه و کاهش زمان پاسخ است. در صورتی که خوشه دچار سربار شود، عواملی مانند فاصله از خوشه تا گره همسایه، تعداد وظایف منتظر در صف هر خوشه و فاصله از گره خوشه تا نزدیک‌ترین مرکز داده ابر برای ارسال وظیفه در نظر گرفته می‌شود. شرما و سینی [۱۶] برای زمان‌بندی وظایف در گره مه، الگوریتم زمان‌بندی زودترین موعد- اول را پیشنهاد دادند. در این روش، ظرفیت فعلی گره مه با استفاده از شبکه عصبی تخمین زده می‌شود. اگر دستگاه اینترنت اشیا، منبع مورد نیاز را دریافت نکند، وظیفه خود را به ابر ارسال می‌کند. در این مقاله، معماری چهارلایه‌ای برای زمان‌بندی وظایف و توازن بار پیشنهاد شده است. لایه اول شامل اینترنت اشیا است که در آن حجم زیادی از داده‌ها هم‌زمان تولید و منتقل می‌شوند. در لایه دوم با استفاده از الگوریتم منطق فازی دوگانه، وظایف به دو دسته مهم و کم‌اهمیت دسته‌بندی می‌شوند. گره‌های با کمترین بار که نزدیک به کاربر هستند، برای اجرای وظایف در اولویت هستند. بار فعلی با استفاده از شبکه عصبی مصنوعی پیش‌بینی می‌شود و الگوریتم زمان‌بندی زودترین موعد- اول برای زمان‌بندی وظیفه در گره‌های مه در نظر گرفته شده است.

کارهای دیگر، مبتنی بر فرض آگاهی از بار گره یا پیش‌بینی بار آینده است. طلعت و همکاران [۱۷]، روش استراتژی بهینه‌سازی و توازن بار با استفاده از روش تخصیص منابع پویا مبتنی بر یادگیری تقویتی و الگوریتم ژنتیک را ارائه دادند. این الگوریتم ترافیک را به طور مداوم در شبکه، کنترل و اطلاعات مربوط به بار هر سرور را جمع‌آوری و درخواست‌های ورودی را کنترل و آنها را با استفاده از روش تخصیص منابع پویا بین سرورهای موجود به طور یکسان توزیع می‌کند. دیویا و لنا [۱۸] با ترکیب محاسبات مه و شبکه‌های نرم‌افزارمحور، روش توازن بار مبتنی بر یادگیری تقویتی را پیشنهاد دادند. این الگوریتم رفتار شبکه را درک کرده و با در نظر گرفتن بار کنونی شبکه و پیش‌بینی بار آینده در شبکه، بار را توزیع می‌کند تا حداکثر دسترسی ممکن به منابع را فراهم کند. ماهیت توزیع‌شده این معماری نیز باعث انعطاف‌پذیری شبکه می‌شود. در این مقاله برای اجرای روش توازن بار، حد آستانه‌ای در نظر گرفته شده که اگر بار سرور بیشتر از ۷۵٪ باشد، الگوریتم توازن بار فراخوانی می‌شود. لو و همکاران [۱۹] نیز از یادگیری تقویتی استفاده می‌کنند تا مسأله بارگیری وظایف را با هدف ایجاد توازن بار بهتر حل کنند. در این مقاله از شبکه LSTM برای بهبود Deep Q-Learning استفاده می‌شود. فضای حالت در این مقاله برابر با مقدار داده ورودی مورد نیاز برای زیروظیفه، پهنای باند downlink، مقدار داده خروجی تولیدشده توسط زیروظیفه و مقدار بار هر سرور است. فضای کنش به صورت برداری $m+2$ بعدی با درایه‌های صفر و یک است. m تعداد سرورهای لبه و ۲ شامل یک سرور ابر و یک دستگاه موبایل می‌شود. هر کدام از درایه‌ها که یک باشند، به معنای بارگیری داده‌ها به آن سرور است. تابع پاداش سه عامل توازن بار، هزینه و مصرف انرژی را در نظر می‌گیرد. برالدی و همکاران در [۲۰] با معرفی دو الگوریتم توازن بار توزیع‌شده Sequential Forwarding و Adaptive Forwarding که برای مقابله با ناهمگونی طراحی شده‌اند، به مسأله مدیریت منابع می‌پردازند. هدف آنها انتقال وظایف به گره‌های همسایه است. در روش اول، یعنی روش Sequential Forwarding یک وظیفه با توجه به حد آستانه θ و حداکثر تعداد قدم‌ها، M، تا زمانی که به دست گره مناسب برسد به صورت تصادفی به گره‌های مه همسایه ارسال می‌شود. به دلیل این که تعیین پارامترهای θ و M کار مشکلی است، روش دوم، یعنی روش Adaptive Forwarding پیشنهاد می‌شود که به طور خودکار و منطبق بر شرایط، این پارامترها را تنظیم می‌کند. طلعت و همکاران [۲۱]، استراتژی توازن بار مؤثر را برای محیط محاسبات مه ارائه دادند که برای کاربردهای مراقبت‌های بهداشتی مناسب است. این



شکل ۱: معماری لایه‌ای محاسبات مه.

الگوریتم تلاش می‌کند تا از طریق زمان‌بندی بلادرنگ و الگوریتم‌های ذخیره‌سازی، به توازن بار مؤثر در محیط مه دست یابد. به علاوه، این الگوریتم ارتباط مناسب بین سرورهای مه و سرورهای لایه ابر و شبنم[1]را تضمین می‌کند.

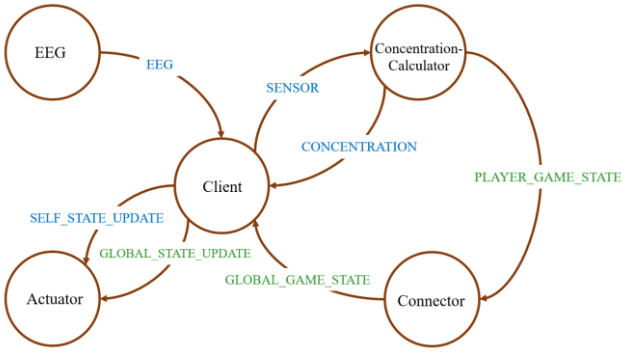
۲-۲ بحث و جمع‌بندی

روش‌های ذکرشده در تحقیقات پیشین نشان می‌دهند که در اکثر این روش‌ها برای تصمیم‌گیری، به اطلاعاتی در مورد ظرفیت یا بار گره‌ها نیاز است و این کار مستلزم انجام یک سری محاسبات زمان‌بر است که باعث افزایش تأخیر و ایجاد بار ترافیکی در شبکه می‌شود. علاوه بر این، در اکثر این روش‌ها معمولاً یک گره خاص عملیات توازن بار را انجام می‌دهد و بار را بین گره‌ها توزیع می‌کند. اما در روش پیشنهادی با توجه به این که خود گره مه به عنوان عامل در نظر گرفته شده است، دیگر نیازی به ارتباط مداوم بین گره‌ها و کنترلر و تأخیر ناشی از این ارتباط نیست. فرایند تصمیم‌گیری در این مقاله با کارهای قبلی متفاوت است، زیرا در روش پیشنهادی، گره مه فقط بر اساس اطلاعاتی که در طول دوره یادگیری از تأخیر و پاداش به دست می‌آورد، برای پردازش و واگذاری وظایف تصمیم می‌گیرد و از ظرفیت و موقعیت سایر گره‌ها اطلاعی ندارد. روش ارائه‌شده در این مقاله، توازن بار را در محیط مه که در آن گره‌ها هیچ اطلاعاتی در مورد یکدیگر ندارند، امکان‌پذیر می‌کند. کاربرد طرح پیشنهادی به سناریوی خاصی محدود نمی‌شود، بلکه هدف آن زیرمجموعه‌ای از مسائل است. علاوه بر این، روش پیشنهادی این مقاله یک روش پویا است و با توجه به شرایط تصمیمات عامل نیز تغییر می‌کند.

۳- مدل سیستم

در این بخش، ابتدا به شرح سیستم پیشنهادی و الگوریتم یادگیری تقویتی پرداخته می‌شود. سپس فرمول‌های مربوط به مسأله توازن بار از طریق یادگیری تقویتی ارائه می‌شوند.

1. Dew



شکل ۲: زیروظایف و وابستگی آنها.

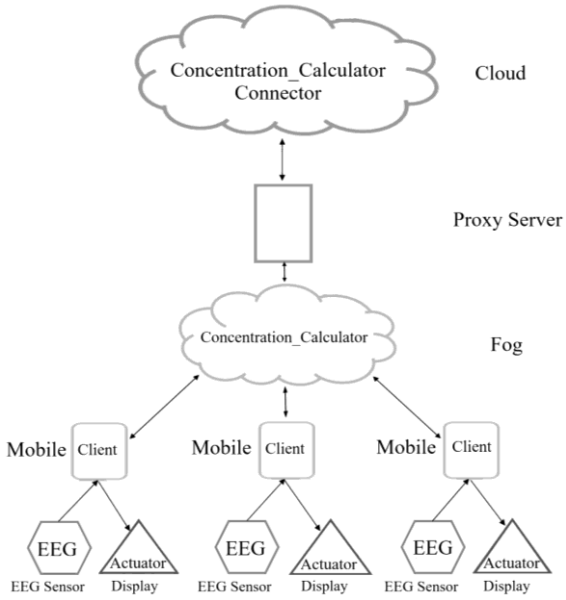
۳-۱ شرح سیستم

در این مقاله مطابق با شکل ۱، معماری چهارلایه‌ای برای سیستم پیشنهادی در نظر گرفته شده که متشکل از اینترنت اشیا، گره‌های مه، سرور پراکسی و مرکز داده ابر است. با توجه به این ساختار، لایه اول، لایه اینترنت اشیا است که شامل چندین دستگاه نهایی مانند گره‌های حسگر بی‌سیم، دستگاه‌های موبایل و غیره است. این دستگاه‌ها مستقیماً به گره‌های مه متصل می‌شوند و می‌توانند داده‌ها را به صورت محلی به آنها ارسال کنند. لایه دوم، لایه مه است که شامل دستگاه‌های خیلی هوشمند نظیر روترها، سوئیچ‌ها و گیت‌وی‌ها است که داده‌ها را از دستگاه‌های نهایی دریافت و پردازش می‌کنند. لایه سوم، شامل سرور پراکسی است که داده‌ها را از گره‌های مه دریافت و به ابر ارسال می‌کند. لایه چهارم، لایه مرکز داده ابر است که شامل چندین سرور و مرکز داده است. با توجه به این ساختار، نیازی به انتقال داده‌ها به ابر مرکزی نیست و در عوض پردازش داده‌ها و اطلاعات به صورت محلی در گره‌های مه انجام می‌شود.

در سیستم پیشنهادی، دستگاه‌های موبایل به دلیل محدودبودن منابع محاسباتی، پردازش بخشی از وظیفه (بازی واقعیت مجازی) را به گره‌های مه در محدوده خود واگذار می‌کنند. در اینجا هیچ گره مرکزی یا کنترلری برای بررسی وضعیت گره‌های مه و شرایط محیطی وجود ندارد و خود گره‌های مه، مسئول جمع‌آوری اطلاعات و تصمیم‌گیری هستند. علاوه بر این، گره‌های مه به طور مستقیم به درخواست‌های کاربران نهایی رسیدگی می‌کنند.

سیستم پیشنهادی به این صورت کار می‌کند: EEG[2] Tractor Beam (نوعی بازی انسان در مقابل انسان) به عنوان برنامه اندروید روی گوشی هوشمند کاربر اجرا می‌شود که تعامل مغز انسان و رایانه را نشان می‌دهد. برای انجام این بازی هر بازیکن باید از یک هدست استفاده کند که به گوشی هوشمند او متصل است. این برنامه، سیگنال‌های حس‌شده توسط هدست را به صورت بلادرنگ، پردازش و وضعیت مغز کاربر را بررسی می‌کند. ممکن است که برای پردازش برنامه به ظرفیت محاسباتی بالایی نیاز باشد. بنابراین برای بهبود زمان پردازش، زمان انتقال و افزایش نرخ بهره‌برداری از منابع شبکه، در این مقاله برنامه به چندین بخش به نام زیروظیفه[3]تقسیم می‌شود. این مقاله وابستگی زیروظایف بازی واقعیت مجازی را بر اساس [۲۲] ارائه می‌کند. همان طور که در شکل ۲ نشان داده شده است، برنامه از پنج زیروظیفه EEG، Client، Actuator، Concentration-Calculator و Connector تشکیل شده که داده‌های این زیروظایف به هم وابسته هستند.

2. Electro Encephalo Gram
3. Subtask



شکل ۳: جایگذاری ماژول‌ها در دستگاه‌های مختلف.

در این برنامه، ماژول‌های Client، Concentration-Calculator و Connector ماژول‌های اصلی هستند که پردازش را انجام می‌دهند. ماژول Client با حسگر ارتباط گرفته و سیگنال‌های EEG را دریافت و سپس مقادیر سیگنال‌های دریافتی را بررسی می‌کند و اگر مقدار آن سیگنال ثابت باشد، آن مقدار را به ماژول Concentration-Calculator که مسئول تعیین وضعیت مغز کاربر از طریق سیگنال دریافتی و محاسبه سطح تمرکز کاربر است، ارسال می‌کند. سپس ماژول Concentration-Calculator سطح تمرکز محاسبه‌شده را به ماژول Client اطلاع می‌دهد. ماژول Client نتایج را به ماژول Actuator ارسال می‌کند تا وضعیت بازی بازیکن روی صفحه نمایش دستگاه موبایل به‌روز شود. ماژول Connector در سطح جهانی کار می‌کند و بازی را بین بازیکنان متعددی که ممکن است در مکان‌های جغرافیایی توزیع‌شده حضور داشته باشند، هماهنگ می‌کند. Connector به طور مداوم وضعیت فعلی بازی را به ماژول Client همه کاربران متصل ارسال می‌کند [۲۲].

به دلیل این که این زیروظایف، پیچیدگی محاسباتی و میزان انتقال داده کمتری دارند، با توجه به منابع در دسترس می‌توان تعدادی از ماژول‌ها را در دستگاه‌های موبایل نگه داشت و آنهایی را که منابع محاسباتی بیشتری نیاز دارند به گره‌های مه یا ابر واگذار کرد. مهم‌ترین حلقه کنترلی در این برنامه، حلقه‌ای است که مسئول تبدیل وضعیت مغز کاربر به وضعیت بازی او در صفحه نمایش دستگاه موبایل است. این امر مستلزم برقراری ارتباط بلادرنگ بین دستگاه موبایل و دستگاهی است که ماژول محاسبه وضعیت مغز کاربر را میزبانی می‌کند. تأخیر در این حلقه به شدت تجربه کاربر را تحت تأثیر قرار می‌دهد، زیرا بر موجودیت‌هایی که کاربر مستقیماً با آنها تعامل دارد، تأثیر می‌گذارد.

به منظور کاهش تأخیر انتقال داده بین ماژول‌ها، ماژول‌های محاسباتی تا حد ممکن باید به منابع داده نزدیک باشند. طرح پیشنهادی این مقاله همان طور که در شکل ۳ نشان داده شده است، به این صورت می‌باشد که ماژول‌های EEG، Client و Actuator مرتبط با دستگاه موبایل هستند، ماژول Concentration-Calculator در گره‌های مه و دو ماژول Connector و Concentration-Calculator در ابر قرار می‌گیرند. به این منظور ماژول Concentration-Calculator نیز در ابر قرار می‌گیرد تا در صورت ایجاد سربار در گره‌های مه، ابر بتواند پردازش مربوط به این ماژول را انجام دهد. برای پردازش برنامه، هر ماژول نتایج پردازش را برای ماژول بعدی در حلقه ارسال می‌کند تا زمانی که نتایج نهایی از آن برنامه به ماژول Actuator در دستگاه موبایل برگردد.

به دلیل پویایی محیط، در هر لحظه به هر گره مه تعداد متغیری دستگاه موبایل متصل می‌شود و ممکن است که بعضی از گره‌های مه زیروظایف بیشتری را نسبت به سایر گره‌های مه دریافت کنند و به مرور دچار سربار شوند. برای جلوگیری از این وضعیت، از روش‌های توازن بار برای توزیع یکنواخت زیروظایف در میان گره‌های مه استفاده می‌شود. هدف اصلی الگوریتم توازن بار در محیط محاسبات مه، بهبود زمان پاسخ به کاربر به وسیله توزیع بار کلی سیستم است، به نحوی که بتواند در شرایط پویای سیستم نیز همچنان بهینه عمل کند. به این منظور در این مقاله، الگوریتم یادگیری تقویتی روی گره‌های مه اعمال می‌شود تا تأخیر، زمان پاسخ به کاربر و اتلاف منابع در شبکه بهبود یابد. هر گره مه بعد از این که زیروظیفه‌ای را دریافت می‌کند، به کمک الگوریتم یادگیری تقویتی تصمیم می‌گیرد که خودش آن را پردازش کند یا این که برای پردازش سریع‌تر، آن را برای گره مه همسایه یا ابر ارسال نماید.

۲-۳ الگوریتم یادگیری تقویتی

الگوریتم‌های یادگیری ماشین اساساً به ۴ دسته، یادگیری با نظارت[1]، یادگیری بدون نظارت[2]، یادگیری نیمه‌نظارتی[3]و یادگیری تقویتی تقسیم می‌شوند. یادگیری با نظارت نیازمند تعدادی داده ورودی برچسب‌دار برای آموزش سیستم است [۲۳]. برخلاف یادگیری با نظارت، در یادگیری بدون نظارت هیچ پاسخ مورد انتظار و هیچ برنامه‌ریزی وجود ندارد و یادگیری بر روی داده‌های بدون برچسب و برای یافتن الگوهای پنهان در این داده‌ها انجام می‌شود [۲۴]. در یادگیری نیمه‌نظارتی، از داده‌های بدون برچسب و داده‌های برچسب‌دار به صورت هم‌زمان استفاده می‌شود تا دقت یادگیری مقداری بهبود یابد [۲۵]. در یادگیری تقویتی، عامل می‌تواند کنش‌های بهینه با محیط را یاد بگیرد. این یادگیری از طریق کاوش در محیط، آزمون و خطا و استفاده از پاداش‌های دریافتی توسط محیط انجام می‌گیرد. در این مقاله برای اعمال توازن بار از الگوریتم Q-Learning استفاده شده است. الگوریتم پیشنهادی، مسأله توازن بار را به صورت فرایند تصمیم‌گیری مارکوف (MDP)[4] فرموله‌بندی می‌کند که در آن گره مه پس از دریافتن حالت[5]محیط، کنشی[6]را انجام می‌دهد و در قبال آن از محیط، پاداشی[7]را دریافت می‌کند. نتایج آزمون و خطا، تجربه[8]نام دارد و توسط چهارتایی 〈حالت فعلی، کنش، پاداش، حالت بعدی〉 بیان می‌شود [۲۴] و [۲۶].

سیاست[9]: سیاست، شیوه رفتار عامل است که به دو صورت سیاست فعال[10]که در آن عامل یادگیری با توجه به عملکرد فعلی ناشی از سیاست فعلی استفاده‌شده، تابع ارزش را می‌آموزد و سیاست غیر فعال[11]که در آن

1. Supervised Learning
2. Unsupervised Learning
3. Semi-Supervised Learning
4. Markov Decision Process
5. State
6. Action
7. Reward
8. Experience
9. Policy
1 . On Policy 0
1 . Off Policy 1

جدول ۱: نمادهای استفاده‌شده برای ارزیابی سیستم و فرمول محاسبه تأخیر.

پارامترها	تعریف	مقدار
W	تعداد زیروظایف پردازش‌شده در گره	–
L	سایز داده زیروظیفه	۳۵۰۰
B	پهنای باند به ازای هر گره	۱۰۰۰۰
$d_{i,j}$	فاصله بین دو گره i و j	–
β_1	ثابت اتلاف مسیر بین دو گره	$۱۰^{-۳}$
β_2	توان اتلاف مسیر	۴
P	توان انتقال گره	۲۰ dBm
N_0	چگالی طیفی توان نویز	۱۷۴ dBm/Hz
I	تعداد دستورات به ازای هر زیروظیفه	۲۰۰×۱۰^۶
$Cycle$	تعداد چرخه CPU به ازای هر دستور	۵
f	سرعت CPU گره مه	۲۸۰۰
	سرعت CPU ابر	۴۴۸۰۰
N	تعداد گره‌های مه	۴
n	تعداد دفعات مشاهده یک حالت	–
α	نرخ یادگیری	$۱/n$
γ	عامل تخفیف	۰٫۹

عامل یادگیری با توجه به عملی که از سیاست دیگری به دست می‌آید، تابع ارزش را می‌آموزد، تعریف می‌شود [۲۷]. الگوریتم Q-Learning، الگوریتمی با سیاست غیر فعال می‌باشد که از رویکرد حریصانه[1] برای یادگرفتن مقدار Q استفاده می‌کند.

تابع ارزش:[2] تابع پاداش، هدف را در تابع یادگیرنده تعیین می‌کند که هرچه به هدف نزدیک‌تر شود، پاداش بیشتری را به دست می‌آورد. با این حال، تابع ارزش در یادگیری تقویتی دیدی بلندمدت دارد و برای هر حالت ارزشی به صورت (۱) تعیین می‌کند که هرچه این مقدار بیشتر باشد، یعنی به هدف نزدیک‌تر شده است

$$v^*(s)=\max_a \sum_{s',r} p(s',r|s,a)[r+\gamma v^*(s')] \qquad (۱)$$

که در این رابطه، $۰<\gamma<۱$ عامل تخفیف[3] نام دارد و اهمیت پاداش‌های آینده را تعیین می‌کند و نشان‌دهنده آن می‌باشد که تصمیمی که در این لحظه گرفته می‌شود، ارزش بیشتری نسبت به تصمیمات آینده دارد. a کنشی است که عامل در حالت فعلی s محیط انجام می‌دهد و به حالت جدید s' گذار می‌کند و r پاداشی است که به ازای انتخاب کنش a در حالت s دریافت می‌کند که تصمیمات آینده را تحت تأثیر قرار خواهد داد.

مدل: مدل مسأله یادگیری تقویتی، تصادفی است و حالت‌های آن غیر قطعی هستند. هر کنش احتمالی است و رفتن از یک حالت به حالت دیگر هم احتمال است.

در مسأله یادگیری تقویتی، عامل از طریق سعی و خطا با محیط تعامل نموده و یاد می‌گیرد تا کنش بهینه را برای حداکثرکردن پاداش بلندمدت انتخاب کند. از این رو یادگیری تقویتی در محیط مه و ابر که محیط‌هایی پویا، در حال تغییر و تصادفی هستند، کاربردهای بسیاری برای بهینه‌سازی یافته است. علاوه بر این، می‌تواند روش مناسبی برای توزیع یکنواخت بارها در بین گره‌های مه باشد.

1. Greedy
2. Value Function
3. Discount Factor

۳-۳ فرمول مسأله

MDP یک فرایند کنترل تصادفی زمان گسسته است. یادگیری تقویتی یکی از رویکردهایی است که برای حل MDP به کار می‌رود و به نوبه خود از برنامه‌ریزی پویا استفاده می‌کند. برای دستیابی به عملکرد مورد نظر، مسأله توازن بار پیشنهادی به عنوان MDP فرموله شده است. معمولاً MDP شامل چهارتایی $\langle S,A,P,R\rangle$ است که برای مسأله توازن بار پیشنهادی به صورت زیر تعریف می‌شوند:

- $S=\{s=(C,Q,N)\}$، فضای حالت است که در آن، C ظرفیت گره مه، Q اندازه صف ارسال به بالا در گره مه و N تعداد دستگاه‌های موبایل متصل به گره مه است.

تصمیم‌گیری در الگوریتم Q-Learning بر اساس حالت فعلی سیستم انجام می‌شود. بسیاری از روش‌های قبلی برای تعریف فضای حالت به اطلاعاتی در مورد ظرفیت گره‌های همسایه نیاز دارند. اما در روش پیشنهادی، وضعیت سیستم تنها بر اساس وضعیت گره مه تصمیم‌گیرنده تعریف می‌شود و این باعث می‌گردد که تصمیم‌گیری بدون اطلاع از وضعیت گره‌های همسایه انجام شود.

- $A=\{a=(n)\}$: فضای کنش بوده که در آن n شامل انتخاب یک گره مه یا ابر برای واگذارکردن زیروظیفه است.
- P: احتمال گذار، مقداری در بازه $[۰,۱]$ است. توزیع احتمال گذار $P(s'|s,a)$ به حالت بعدی s' با انتخاب کنش a به شرطی است که در حالت s باشد.
- R: پاداش مربوط به انجام کنش a در حالت s است. هدف اصلی، انتخاب کنش بهینه در هر سیستم است به نحوی که ارزش بلندمدت، حداکثر و تأخیر پردازش و احتمال سربار حداقل شود.

همان طور که گفته شد، یک وظیفه (بازی واقعیت مجازی) به چندین زیروظیفه تقسیم می‌شود. تأخیر اجرای وظیفه، برابر با مجموع تأخیر انتقال و پردازش تمام زیروظایف مربوط به این وظیفه در دستگاه‌های مختلف است که به صورت زیر محاسبه می‌شود

$$T_{task}=t_{out}-t_{in} \qquad (۲)$$

که در آن t_{in} و t_{out} به ترتیب زمان ورود و زمان خروج یک وظیفه در سیستم پیشنهادی هستند. به دلیل این که تأخیر پردازش و انتقال در دستگاه‌های موبایل حدوداً مقداری ثابت است، در این مقاله برای محاسبه تأخیر اجرای وظیفه، تنها به محاسبه تأخیر پردازش زیروظیفه Concentration-Calculator در گره‌های مه یا ابر پرداخته و تأخیر پردازش زیروظیفه در دستگاه موبایل مقداری ثابت در نظر گرفته می‌شود. در روش پیشنهادی، الگوریتم Q-Learning بر روی گره‌های مه اجرا می‌شود که در آن تابع پاداش $R(s,a)$ برابر با منفی تأخیر پردازش زیروظیفه واگذارشده به گره مه در نظر گرفته شده و هرچه تأخیر پردازش زیروظیفه بیشتر باشد، در نتیجه پاداش کمتری دریافت می‌شود. $R(s,a)$ به صورت زیر محاسبه می‌شود

$$R(s,a)=-T_{subtask} \qquad (۳)$$

که $T_{subtask}$ تأخیر پردازش زیروظیفه Concentration-Calculator واگذارشده به گره مه است. نمادهای استفاده‌شده برای ارزیابی سیستم و فرمول محاسبه تأخیر در جدول ۱ آمده‌اند.

تأخیر پردازش زیروظیفه به این که پردازش آن در گره مهی که آن زیروظیفه را از دستگاه موبایل دریافت کرده (FN-I) انجام شود یا این که به گره مه همسایه (FN-J) یا ابر واگذار شود، بستگی دارد.

- اگر زیروظیفه توسط FN-I پردازش شود، $T_{subtask}$ برابر با تأخیر اجرای زیروظیفه است که به صورت زیر محاسبه می‌شود

$$T_{subtask} = \frac{I \times Cycle \times W}{f} \quad (۴)$$

- اگر زیروظیفه به FN-J یا ابر برای پردازش واگذار شود، $T_{subtask}$ به صورت زیر محاسبه می‌شود

$$T_{subtask} = t_{W_i} + t_{C_{ij}} + t_{E_j} + t_{W_j} + t_{C_{ji}} \quad (۵)$$

که در این رابطه، t_{W_i} و t_{W_j} به ترتیب تأخیر انتظار زیروظیفه در صف ارسال FN-I و تأخیر انتظار نتیجه زیروظیفه در صف ارسال FN-J یا ابر هستند. $t_{C_{ij}}$ تأخیر ارسال زیروظیفه روی کانال ارتباطی از FN-I به FN-J یا ابر است. $t_{C_{ji}}$ تأخیر ارسال نتیجه زیروظیفه روی کانال ارتباطی از FN-J یا ابر به FN-I و t_{E_j} تأخیر اجرای زیروظیفه در FN-J یا ابر است. به دلیل این که سرور پراکسی تنها زیروظیفه را به ابر ارسال می‌کند و پردازشی انجام نمی‌دهد، فرض گردیده که تأخیر ارسال روی کانال ارتباطی FN-I به ابر، مستقیماً و بدون در نظر گرفتن سرور پراکسی محاسبه می‌شود.

- تأخیر انتظار در صف ارسال گره‌های i و j یا ابر به صورت زیر محاسبه می‌شود

$$t_W = t_o - t_i \quad (۶)$$

که در آن t_i و t_o به ترتیب زمان ورود زیروظیفه m به صف گره و زمان خروج زیروظیفه m از صف آن گره است.

- تأخیر ارسال زیروظیفه روی کانال ارتباطی از FN-I به FN-J یا ابر و برعکس، برابر است با

$$t_C = \frac{L}{r_{i,j}} \quad (۷)$$

که $r_{i,j}$ نرخ انتقال بین دو گره i و j بوده و برابر است با

$$r_{i,j} = B\log(1 + \frac{g_{i,j} \times P}{B \times N_0}) \quad (۸)$$

که در آن $g_{i,j} = \beta_1 d_{i,j}^{-\beta_2}$ بهره کانال بین دو گره i و j است.

- تأخیر اجرای زیروظیفه در FN-J یا ابر t_{E_j} برابر است با

$$t_{E_j} = \frac{I \times Cycle \times W}{f} \quad (۹)$$

هر گره مه عاملی است که در محیطی با فضای حالت S به یادگیری مشغول است. ورود هر وظیفه جدید به سیستم باعث می‌شود تا عامل، کنشی در محیط انجام دهد و یکی از گره‌ها را برای تخصیص زیروظیفه جدید انتخاب کند. پاداش این تخصیص هم در هنگام به‌روزرسانی حالت محیط مشخص خواهد شد. در صورتی که حالت فعلی سیستم به توازن بار نزدیک‌تر باشد و زیروظایف با تأخیر کمتری پردازش شوند، پاداش به عامل تعلق خواهد گرفت و در غیر این صورت هیچ پاداشی به آن تعلق نمی‌گیرد. از طریق پاداش‌های به دست آمده، به مرور هر گره یاد می‌گیرد که به منظور پردازش زیروظیفه، بهترین تصمیم را بگیرد. علاوه بر این، کاهش تأخیر پردازش زیروظیفه در محیط مه، تأخیر کل و زمان پاسخ به کاربر را کاهش می‌دهد و به همین دلیل شبکه مدت زمان کمتری را درگیر پردازش وظیفه خواهد شد.

٤- روش توازن بار پیشنهادی

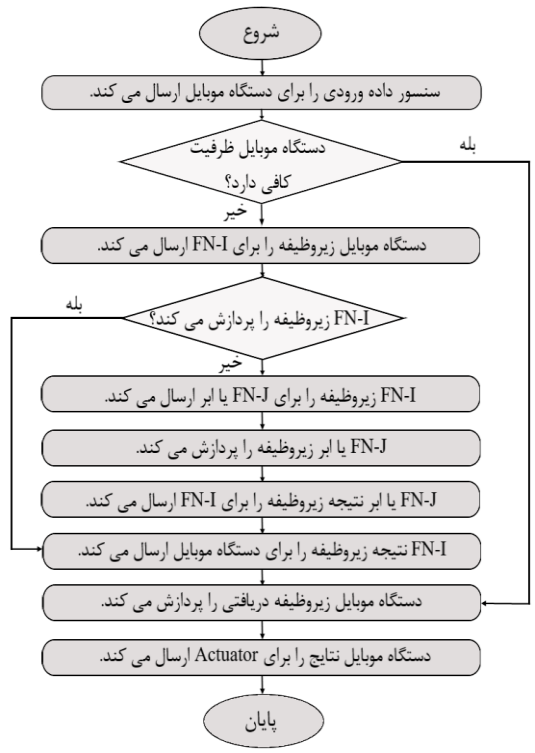
در این بخش، الگوریتم توازن بار مبتنی بر یادگیری تقویتی برای توزیع یکنواخت بار در میان گره‌های مه ارائه شده تا مشکلات روش‌های پیشین (از قبیل انجام محاسبات زمان‌بر برای اطلاع از ظرفیت و بار گره‌ها و ...) برطرف شود. هنگامی که سیستم برای تمام جفت‌های حالت- کنش، تابع گذار و تابع پاداش را داشته باشد، می‌توان از طریق برنامه‌نویسی پویا، MDP را حل کرد. اما اکثر مواقع سیستم نمی‌تواند مقدار دقیق تابع گذار و پاداش را برای بیشتر حالت‌ها پیش‌بینی کند که برای رفع این مشکلات، یادگیری تقویتی پیشنهاد شده است. در این مقاله از میان الگوریتم‌های یادگیری تقویتی، از الگوریتم Q-Learning برای یافتن حالت- کنش بهینه با حداقل هزینه محاسباتی استفاده شده که نبود اطلاعات کافی را با تجربه‌کردن جبران می‌کند. مدل الگوریتم Q-Learning تصادفی است و حالت‌های آن غیر قطعی هستند.

عامل در مسائل یادگیری، محیط را کاوش کرده و یاد می‌گیرد که در هر حالت، کنش بهینه را برای حداکثرسازی پاداش بلندمدت انتخاب کند. توازن بار بر روی گره‌های مه اعمال می‌شود و به این صورت است که بار را بین گره‌های مه توزیع می‌کند. گره مه به عنوان عامل در نظر گرفته شده که به یادگیری شبکه مشغول است. بعد از این که گره مه زیروظیفه جدیدی را دریافت می‌کند، به کمک یادگیری تقویتی برای پردازش آن تصمیم‌گیری خواهد کرد. به این صورت که گره مه زیروظیفه Concentration-Calculator را از دستگاه موبایل، دریافت و سپس گره مه حالت فعلی محیط را مشاهده می‌کند و به منظور حداکثرسازی پاداش بلندمدت، بر اساس تجربیات و پاداش‌هایی که تا کنون دریافت کرده و با توجه به ظرفیتی که دارد، تصمیم می‌گیرد که خودش آن زیروظیفه را پردازش کند و یا این که برای پردازش سریع‌تر، زیروظیفه را برای گره مه مجاورش ارسال نماید. در صورتی که تأخیر پردازش زیروظیفه Concentration-Calculator در گره‌های مه بیشتر از تأخیر پردازش آن زیروظیفه در ابر باشد، گره مه تصمیم می‌گیرد که پردازش این زیروظیفه را به ابر واگذار کند تا زیروظیفه سریع‌تر پردازش شده و بار روی گره‌های مه نیز کاهش یابد. با توجه به مدل سیستم، هر گره مه پس از مشاهده حالت فعلی s، کنش a را انجام می‌دهد، سپس گذاری صورت می‌گیرد و حالت جدید s' مشاهده می‌شود و در قبال آن از محیط پاداش $R(s,a)$ را دریافت می‌کند. از طریق این مشاهدات می‌توان تابع ارزش را برای حالت s و کنش a به صورت زیر تخمین زد

$$Q_{new}(s,a) = Q_{old}(s,a) + \alpha[R(s,a) + \gamma \max_a Q(s',a') - Q_{old}(s,a)] \quad (۱۰)$$

که $0 < \alpha < 1$ نرخ یادگیری[1]است و بین مشاهدات جدید و چیزی که یاد گرفته است، توازن ایجاد می‌کند. در روش حریصانه با استفاده از دانش فعلی، کنشی انتخاب می‌شود که حداکثر پاداش را در یک گام نتیجه دهد. به منظور حداکثرسازی ارزش بلندمدت، در الگوریتم Q-Learning از سیاست $\varepsilon-$greedy استفاده می‌شود که در آن ε احتمال انتخاب کنش با پاداش بالاتر است. برای پردازش زیروظیفه، گره مهی انتخاب می‌شود که حداکثر ارزش بلندمدت را به دست آورد. به این صورت که در روش $\varepsilon-$greedy در هر گام زمانی، کنش تصادفی با احتمال ثابت $0 \le \varepsilon \le 1$ و کنش با بالاترین ارزش با احتمال $1-\varepsilon$ انتخاب می‌شود. مزیت استفاده از الگوریتم $\varepsilon-$greedy آن است که هرچه مراحل بیشتر شوند، هر

1. Learning Rate



شکل ۴: روندنمای روش توازن بار پیشنهادی.

کنشی بی‌نهایت بار مشاهده شده و می‌توان اطمینان یافت که $Q(s,a)$ به مقدار بهینه همگرا می‌شود. بنابراین گره مه با استفاده از الگوریتم Q-Learning یاد می‌گیرد که مناسب‌ترین گره را برای پردازش زیروظیفه انتخاب کند. تابع پاداش مناسبی که روش پیشنهادی در نظر گرفته است، با استفاده از (۳) محاسبه می‌شود و در شکل ۴، روندنمای روش توازن بار پیشنهادی آمده است.

همان طور که گره مه از طریق الگوریتم یادگیری در شبکه کاوش می‌کند، کل شبکه و احتمال بارگذاری به هر گره را یاد می‌گیرد و می‌تواند مناسب‌ترین گره را برای واگذاری زیروظیفه انتخاب کند. در ابتدا الگوریتم از طریق روش حریصانه به کاوش شبکه می‌پردازد و پس از آن که درک کاملی از شبکه به دست آمد، توازن بار بهینه صورت می‌گیرد. فضای حالت برابر با ظرفیت گره مه، اندازه صف ارسال به بالا در گره مه و تعداد دستگاه‌های موبایل متصل به گره مه است. کنش انتخاب یکی از گره‌های مه یا ابر و پاداش، تابعی برای کمینه‌کردن تأخیر پردازش زیروظیفه است.

۵- ارزیابی نتایج

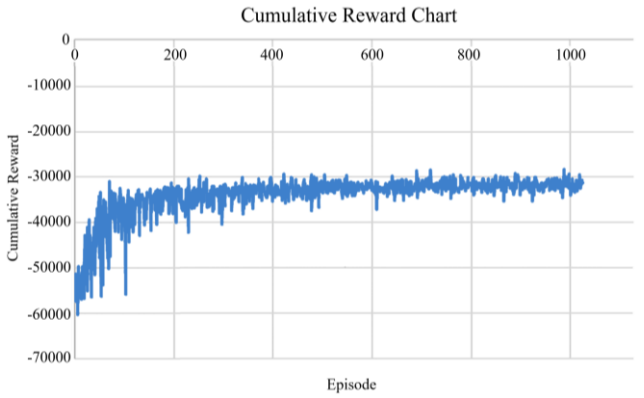
در این مقاله، از شبیه‌ساز iFogSim [۲۸] برای شبیه‌سازی مسأله توازن بار مبتنی بر الگوریتم یادگیری تقویتی استفاده می‌شود. این برنامه در رایانه ایسوس با پردازنده Intel Core i۷ و با RAM ۸ GB اجرا شده است. سیستم پیشنهادی شامل N گره مه و تعداد متغیری دستگاه موبایل است که به صورت تصادفی به گره‌های مه مجاورشان متصل می‌شوند و زیروظایف را به آنها ارسال می‌کنند. در این مقاله، توازن بار در محیط مه با استفاده از الگوریتم Q-Learning پیشنهاد شده است. ابتدا در الگوریتم Q-Learning، مقادیر Q-table همگی صفر هستند و گره مه هیچ اطلاعاتی در مورد شبکه ندارد. برای یادگیری از روش $\varepsilon-$greedy استفاده می‌شود. ابتدا مقدار ε برابر با ۱ است و الگوریتم به صورت کاملاً حریصانه به اکتشاف شبکه می‌پردازد. به تدریج و پس از آن که اطمینان گره مه در تخمین Q-tableها افزایش یافت، مقدار آن به ۰٫۳ تغییر می‌کند. به علاوه، مقدار پاداش دریافتی برابر با منفی تأخیر پردازش زیروظیفه Concentration-Calculator در گره مه یا ابر در نظر گرفته شده است.

در شبیه‌سازی، فرض گردیده است که ایجاد و ارسال وظایف اکنون انجام شده و از همین ابتدا الگوریتم یادگیری اجرا می‌شود که تا حد ممکن از ایجاد سربار در گره‌ها جلوگیری کند. با استفاده از الگوریتم یادگیری Q-Learning، گره مه می‌تواند کنش‌های بهینه با محیط را یاد بگیرد. این یادگیری از طریق کاوش در محیط، آزمون و خطا و استفاده از پاداش‌های دریافتی توسط محیط انجام می‌گیرد. در حالت کلی هر گره مه پس از بررسی حالت محیط، گرهی را برای واگذاری زیروظیفه انتخاب می‌کند و در قبال آن از محیط پاداشی را دریافت می‌کند. در هر تکرار الگوریتم، تجربیات گره مه از شبکه افزایش می‌یابد و به مرور یاد می‌گیرد که زیروظیفه را به گرهی که بار کمتری دارد و می‌تواند پردازش زیروظیفه را با تأخیر کمتری انجام دهد، واگذار کند. اما در سایر روش‌ها (برای مثال [۳] و [۲۹]) برخلاف روش پیشنهادی، الگوریتم توازن بار بعد از ایجاد سربار در گره مه، اجرا می‌شود که این امر باعث افت عملکرد سیستم‌های مطرح و افزایش تأخیر در آنها می‌شود. مزیت دیگر این است که در روش‌های مقایسه‌شده برای واگذاری زیروظایف به گره‌های همسایه، نیاز به اطلاع از ظرفیت و موقعیت این گره‌هاست که اینها از طریق انجام یک سری محاسبات زمان‌بر به دست می‌آیند. اما در سیستم پیشنهادی این محاسبات اضافی و زمان‌بر حذف می‌شوند و گره مه فقط بر اساس تجربیاتی که از محیط کسب کرده است، عمل می‌کند. این امر باعث می‌شود تا حد ممکن تأخیر سیستم پیشنهادی کاهش یابد.

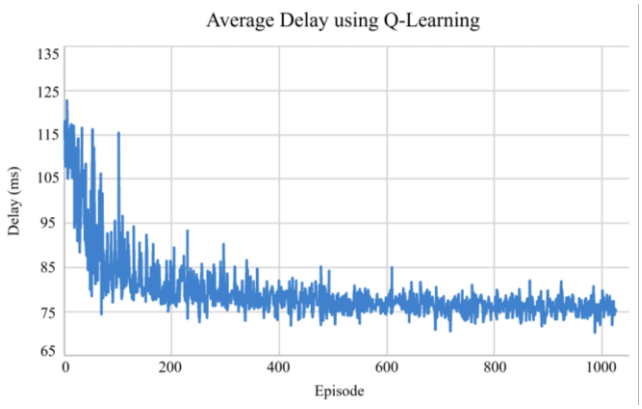
عملکرد روش پیشنهادی با تعدادی از روش‌های توازن بار موجود مقایسه گردیده که در این شبیه‌سازی از روش‌های توازن بار SALB[1] [۳]، تصادفی و Proportional [۲۹] به عنوان معیار استفاده شده است. روش SALB بعد از این که گره مه دچار سربار شد، ظرفیت سایر گره‌های همسایه را با هم مقایسه کرده و زیروظیفه را به گرهی که حداقل ۴۰٪ ظرفیت آن خالی است و بیشترین ظرفیت را دارد، ارسال می‌کند. در روش تصادفی، زیروظایف به صورت تصادفی برای گره‌های مه ارسال می‌شوند. در روش Proportional، اطلاعات ظرفیت همه همسایه‌ها دریافت شده و مناسب‌ترین گره به نسبت اندازه زیروظیفه انتخاب می‌شود. در ادامه، نمودارهای حاصل از پیاده‌سازی الگوریتم Q-Learning بر روی مسأله توازن بار آورده شده و در انتها به بررسی نتایج اجرای هر چهار الگوریتم و مقایسه عملکرد آنها در تأخیر، زمان پاسخ به کاربر و توازن بار پرداخته می‌شود.

شکل ۵، افزایش پاداش تجمعی در هر بار تکرار الگوریتم پیشنهادی را نشان می‌دهد. همان طور که گفته شد، پاداش برابر با منفی تأخیر پردازش زیروظیفه واگذارشده به گره مه است. بسته به این که زیروظیفه توسط گره مه یا ابر پردازش شود، افزایش تأخیر پردازش زیروظیفه باعث کاهش پاداش دریافتی می‌شود. در تصمیم‌گیری برای واگذاری بار، کنشی انتخاب می‌شود که بالاترین Q-value را به دست آورد. در روش پیشنهادی، تصمیم واگذاری بار مبتنی بر Q-Learning، تأخیر پردازش و احتمال سربار را با توجه به عملکرد پاداش پیشنهادی به حداقل می‌رساند. پاداش تجمعی با افزایش نرخ پردازش وظایف، افزایش می‌یابد. در عین حال با افزایش تعداد وظایف ورودی، پاداش تجمعی به طور مداوم کاهش می‌یابد، زیرا وظایف زیادی در صف گره‌ها قرار می‌گیرند.

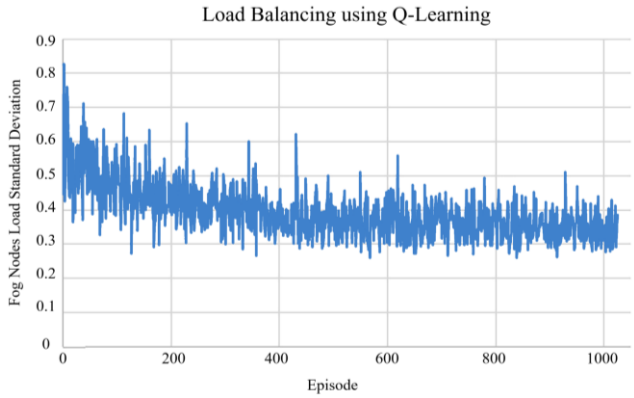
1. Secure Authentication and Load Balancing



شکل ۵: پاداش تجمعی در هر بار تکرار الگوریتم پیشنهادی.



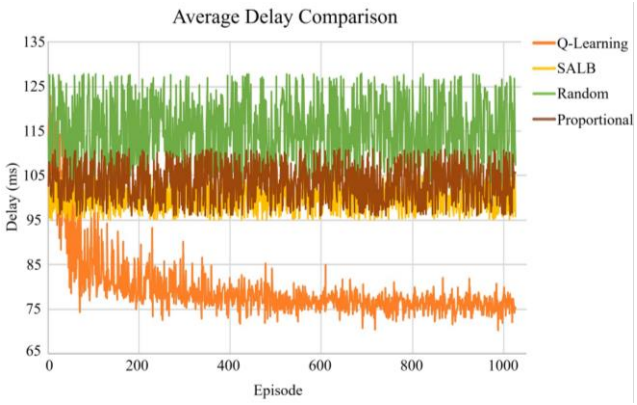
شکل ۶: میانگین تأخیر اجرای وظایف در روش Q-Learning.



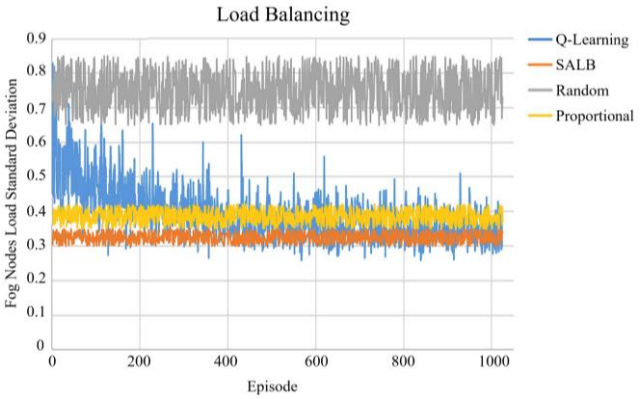
شکل ۷: انحراف معیار بار روی گره‌ها در روش Q-Learning.

شکل ۶ نشان می‌دهد که با تکرار برنامه و افزایش یادگیری، میانگین تأخیر اجرای وظایف به شکل قابل توجهی کاهش یافته است. به این صورت که گره مه به مرور یاد می‌گیرد که پردازش زیروظیفه را به گرهی واگذار کند که حداقل تأخیر را نتیجه دهد و این امر نیز باعث کاهش تأخیر شبکه و زمان پاسخ به کاربر می‌شود.

در شکل ۷ نیز نشان داده شده که با افزایش یادگیری، انحراف معیار بار روی گره‌ها کاهش می‌یابد. در نتیجه توازن بار بهبود یافته و می‌توان اطمینان یافت که وظایف به صورت متوازن در شبکه توزیع و پردازش می‌شوند. همان طور که گفته شد، گره مه به تدریج یاد می‌گیرد پردازش زیروظیفه را به گرهی واگذار کند که حداقل تأخیر را نتیجه دهد. تأخیر پردازش زیروظیفه در صورتی کاهش می‌یابد که آن زیروظیفه به گرهی واگذار شود که ظرفیت بیشتری دارد و در نتیجه می‌تواند زیروظیفه ورودی را سریع‌تر پردازش کند. واگذارکردن زیروظیفه به گره مه با بیشترین ظرفیت، از ایجاد سربار و کم‌باری سایر گره‌ها جلوگیری می‌کند.



شکل ۸: میانگین تأخیر اجرای وظایف.

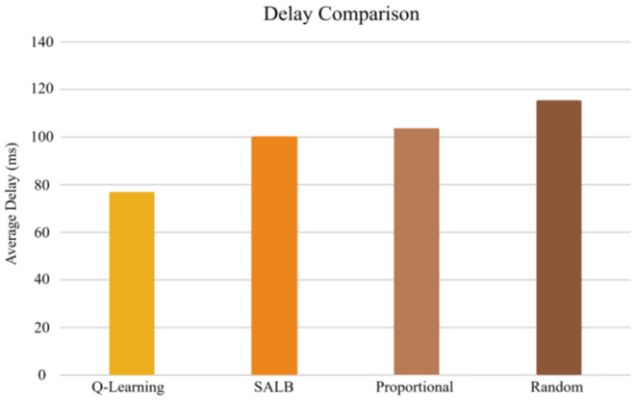


شکل ۹: انحراف معیار بار روی گره‌ها.

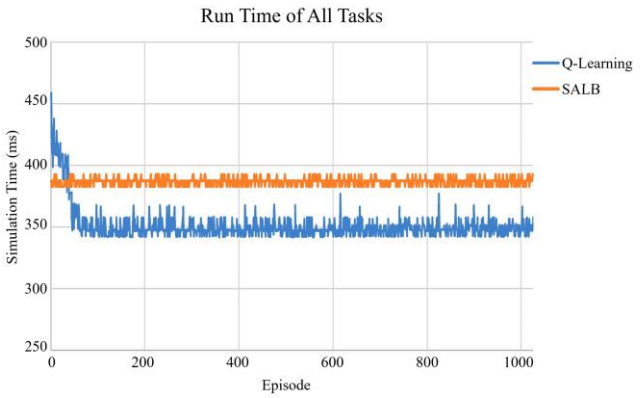
حال در ادامه به بررسی نتایج اجرای هر ۴ الگوریتم و مقایسه عملکرد آنها پرداخته می‌شود.

با استفاده از الگوریتم Q-Learning انتظار می‌رود که عملکرد توازن بار در شبکه به شکل قابل توجهی بهبود یابد. در ارزیابی انجام‌گردیده، در شکل ۸ نشان داده شده است که تصمیم واگذاری وظایف مبتنی بر Q-Learning، تأخیر اجرای وظایف را با توجه به تابع پاداش پیشنهادی به حداقل می‌رساند. الگوریتم توازن بار پیشنهادی با استفاده از یادگیری تقویتی، بار را به شکل متوازنی روی گره‌ها توزیع می‌کند که این امر باعث می‌شود که گره‌ها بتوانند زیروظایف را با تأخیر کمتری پردازش کنند. پاداش تجمعی با افزایش نرخ ورود وظایف کاهش می‌یابد، زیرا تعداد نسبتاً زیادی زیروظیفه در صف گره‌های مه قرار می‌گیرد و بنابراین تعداد کمتری از وظایف را می‌توان توسط آنها پردازش کرد. با این حال، الگوریتم توازن بار پیشنهادی با توزیع مناسب بار بین گره‌های مه، پاداش مطلوبی را نتیجه می‌دهد که تأخیر هم به همان نسبت بهبود می‌یابد. علاوه بر این، در روش توازن بار پیشنهادی به علت این که برخلاف روش‌های مقایسه‌شده هیچ محاسبات زمان‌بری برای آگاهی از ظرفیت و موقعیت گره‌های همسایه انجام نمی‌شود، در نتیجه همان طور که در این شکل مشاهده می‌کنید، تأخیر روش پیشنهادی به شکل چشم‌گیری نسبت به سایر روش‌ها کاهش می‌یابد.

پس از بررسی میانگین تأخیر اجرای وظایف، انحراف معیار بار روی گره‌ها در هر ۴ روش با هم مقایسه می‌شود. با توجه به شکل ۹، ابتدا در الگوریتم SALB انحراف معیار بار روی گره‌ها کمتر از سایر روش‌هاست، اما با افزایش یادگیری عامل، انحراف معیار بار روی گره‌ها در روش پیشنهادی به شکل قابل توجهی کاهش یافته است. این امر نشان می‌دهد که در روش پیشنهادی، وظایف به صورت متوازن در شبکه توزیع می‌شوند و احتمال ایجاد سربار در گره‌ها کاهش می‌یابد.



شکل ۱۰: تأخیر کل.



شکل ۱۱: زمان اجرای تمام وظایف.

در شکل ۱۰، تأخیر کل اجرای وظایف در هر ۴ روش بررسی شده است. تأخیر کل از طریق میانگین تأخیر اجرای وظایف ورودی از ابتدا تا آخرین تکرار اجرای الگوریتم به دست می‌آید. ارزیابی انجام‌شده نشان می‌دهد که روش Q-Learning، تأخیر کل کمتری نسبت به روش‌های دیگر دارد و این بدین معنی است که این روش نسبت به روش‌های دیگر عملکرد بهینه‌تری دارد.

در شکل ۱۱، زمان اجرای تمام وظایفی که وارد سیستم می‌شوند، در روش پیشنهادی و روش SALB با هم مقایسه شده است. همان طور که مشاهده می‌شود روش پیشنهادی نسبت به روش SALB، زمان اجرای تمام وظایفی را که در طول زمان شبیه‌سازی وارد سیستم شده‌اند، به شکل قابل توجهی کاهش داده است. این امر باعث می‌شود که روش پیشنهادی به نتایج بهینه‌ای دست یابد و سیستم پیشنهادی، عملکرد بهتری را نسبت به سایر روش‌های مقایسه‌شده داشته باشد.

نتایج، این واقعیت را نشان می‌دهند که وقتی گره مه با استفاده از الگوریتم Q-Learning تصمیم به واگذاری بار می‌گیرد، نه تنها حالت صف گره‌ها، بلکه ظرفیت گره‌ها و تعداد دستگاه‌های موبایل متصل به هر گره را نیز در نظر می‌گیرد. بنابراین الگوریتم پیشنهادی می‌تواند تخصیص ناموفق را به حداقل برساند. از ارزیابی فوق می‌توان نتیجه گرفت که روش توازن بار پیشنهادی، پایدارتر از سایر روش‌های توازن بار است و تأخیر شبکه و زمان پاسخ به کاربر را به شکل قابل توجهی کاهش می‌دهد.

٦- نتیجه‌گیری

هدف این مقاله ارائه طرحی در راستای بهبود توازن بار در گره‌های مه است. در محاسبات مه به دلیل ویژگی‌هایی خاص مانند پویایی توپولوژی و ناهمگن‌بودن منابع، مسائلی همچون واگذاری وظایف و توازن بار در این سیستم‌ها، چالش‌برانگیز و استفاده از روش‌های رایج برای حل این مسائل ناکارآمد است. از رویکردهای نوظهور برای حل مسائل پیچیده، استفاده از الگوریتم‌های یادگیری ماشین و از میان آنها یادگیری تقویتی است. در این مقاله، توازن بار در محیط مه با استفاده از الگوریتم Q-Learning ارائه شده است. این الگوریتم با استفاده از تجربه‌ای که عامل یادگیرنده از تعامل با محیط کسب می‌کند به سیاست بهینه بلندمدتی دست می‌یابد. در روش پیشنهادی با کمک فرایند تصمیم‌گیری مارکوف، هر گره مه به عنوان عامل، به کاوش در محیط مه پرداخته و به دنبال یافتن گره کم‌بار و مناسب برای واگذاری زیروظایف است تا حداقل زمان پردازش و احتمال ایجاد سربار حاصل شود. روش پیشنهادی برای تعداد مختلف گره مه و دستگاه موبایل درون شبکه آزمایش شده و بازده خوبی داشته است. نتایج شبیه‌سازی نشان می‌دهند که الگوریتم پیشنهادی در مقایسه با روش‌های موجود، تأخیر پردازش، زمان پاسخ به کاربر و احتمال تخصیص ناموفق وظایف را به شکل قابل توجهی کاهش می‌دهد. با توجه به ساختار شبکه، به کارگیری الگوریتم یادگیری در هر دستگاه اینترنت اشیا به منظور بهبود بیشتر توازن بار و کاهش تأخیر، از جمله جهت‌های تحقیقاتی آتی است که این مقاله آن را برای محققین باز می‌نماید.

مراجع

[1] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management, and evaluation of fog computing systems: a survey," *IEEE Internet of Things J.*, vol. 8, no. 4, pp. 2494-2516, Feb. 2020.

[2] A. Yousefpour, *et al.*, "All one needs to know about fog computing and related edge computing paradigms: a complete survey," *J. of Systems Architecture*, vol. 98, pp. 289-330, Sept. 2019.

[3] D. Puthal, *et al.*, "Secure authentication and load balancing of distributed edge data centers," *J. of Parallel and Distributed Computing*, vol. 124, pp. 60-69, Feb. 2019.

[4] N. Khattar, J. Sidhu, and J. Singh, "Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques," *The J. of Supercomputing*, vol. 75, no. 8, pp. 4750-4810, Aug. 2019.

[5] M. Adhikari and T. Amgoth, "Heuristic-based load-balancing algorithm for IaaS cloud," *Future Generation Computer Systems*, vol. 81, pp. 156-165, Apr. 2018.

[6] S. Rasheed, *et al.*, " A cloud-fog based smart grid model using max-min scheduling algorithm for efficient resource allocation," in *Proc. Int. Conf. on Network-Based Information Systems*, vol. 22, pp. 273-285, Sept. 2018.

[7] L. Abualigah *et al.*, "Advances in meta-heuristic optimization algorithms in big data text clustering," *Electronics*, vol. 10, no. 2, Article ID: 101, 2021.

[8] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud," *J. of Network and Computer Applications*, vol. 128, pp. 64-77, Feb. 2019.

[9] L. Shen, J. Li, Y. Wu, Z. Tang, and Y. Wang, "Optimization of artificial bee colony algorithm based load balancing in smart grid cloud," in *Proc. of the IEEE Inovative Smart Grid Technologies-Asia, ISGT Asia*, pp. 1131-1134, Chengdu, China, 21-24 May 2019.

[10] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803-17818, 2022.

[11] S. R. Deshmukh, S. K. Yadav, and D. N. Kyatanvar, "Load balancing in cloud environs: optimal task scheduling via hybrid algorithm," *International J. of Modeling, Simulation, and Scientific Computing*, vol. 12, no. 2, Article ID: 2150008, Apr. 2021.

[12] M. M. Golchi, S. Saraeian, and M. Heydari, "A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: performance evaluation," *Computer Networks*, vol. 162, Article ID: 106860, Oct. 2019.

[13] J. Y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *Proc. of the IEEE Wireless Communications and Networking Conf., WCNC'19*, 7 pp., Marrakesh, Morocco, 15-18 Apr. 2019.

[14] X. Xu, S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, X. Sun, and A. X. Liu, "Dynamic resource allocation for load balancing in fog

environment," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID: 6421607, Apr. 2018.
[15] A. B. Manju and S. Sumathy, "Efficient load balancing algorithm for task preprocessing in fog computing environment," *In Smart Intelligent Computing and Applications*, Springer, Singapore, vol. 105, pp. 291-298, 2019.
[16] S. Sharma and H. Saini, "A novel four-tier architecture for delay aware scheduling and load balancing in fog environment," *Sustainable Computing: Informatics and Systems*, vol. 24, Article ID: 100355, Dec. 2019.
[17] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali, and S. H. Ali, "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment," *J. of Ambient Intelligence and Humanized Computing*, vol. 11, no. 11, pp. 4951-4966, Nov. 2020.
[18] V. Divya and R. L. Sri, "ReTra: reinforcement based traffic load balancer in fog based network," in *Proc. of the 10th Int. Conf. on Computing, Communication and Networking Technologies, ICCCNT'19*, 6 pp., Kanpur, India, 6-8 Jul. 2019.
[19] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning," *Future Generation Computer Systems*, vol. 102, pp. 847-861, Jan. 2020.
[20] R. Beraldi, C. Canali, R. Lancellotti, and G. P. Mattia, "A random walk based load balancing algorithm for fog computing," in *Proc. of the Fifth Int. Conf. on Fog and Mobile Edge Computing, FMEC'20*, pp. 46-53, Paris, France 20-23 Apr. 2020.
[21] F. M. Talaat, S. H. Ali, A. I. Saleh, and H. A. Ali, "Effective load balancing strategy (ELBS) for real-time fog computing environment using fuzzy and probabilistic neural networks," *J. of Network and Systems Management*, vol. 27, no. 4, pp. 883-929, Oct. 2019.
[22] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Software-Practice and Experience*, vol. 47, no. 9, pp. 1275-1296, Sep. 2017.
[23] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): research issues and challenges," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 393-430, Aug. 2019.
[24] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: recent advances and application examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94-101, Apr. 2018.
[25] R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki, "A survey on semi-supervised feature selection methods," *Pattern Recognition*, vol. 64, pp. 141-158, Apr. 2017.
[26] A. Mebrek, M. Esseghir, and L. Merghem-Boulahia, "Energy-efficient solution based on reinforcement learning approach in fog networks," *Proc. of the Fifth 15th Int. Wireless Communications & Mobile Computing Conf., IWCMC'19*, pp. 2019-2024, Tangier, Morocco, 24-28 Jun. 2019.
[27] Y. Xu, W. Xu, Z. Wang, J. Lin, and S. Cui, "Load balancing for ultradense networks: a deep reinforcement learning-based approach," *IEEE Internet of Things J.*, vol. 6, no. 6, pp. 9399-9412, Aug. 2019.
[28] R. Mahmud and R. Buyya, "Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit," Fog and Edge Computing, pp. 433-465, Jan. 2019.
[29] I. Tellioglu and H. A. Mantar, "A proportional load balancing for wireless sensor networks," in *Proc. of the Fifth 3rd Int. Conf. on Sensor Technologies and Applications*, pp. 514-519, Athens, Greece, 18-23 Jun. 2009.

نیلوفر طهماسبی‌پویا در سال ۱۳۹۶ مدرک کارشناسی مهندسی فناوری اطلاعات خود را از دانشگاه ایلام و در سال ۱۴۰۰ مدرک کارشناسی ارشد مهندسی فناوری اطلاعات- شبکه‌های کامپیوتری خود را از دانشگاه یزد دریافت نمود. زمینه‌های علمی مورد علاقه نام‌برده متنوع بوده و شامل موضوعاتی مانند محاسبات مه، محاسبات ابری، شبکه‌های بی‌سیم، محاسبات لبه موبایل و یادگیری ماشین است.

مهدی‌آقا صرام در سال ۱۳۵۳ مدرک کارشناسی مهندسی صنایع خود را از دانشگاه صنعتی شریف و در سال ۱۳۵۵ مدرک کارشناسی ارشد خود را در رشته علوم کامپیوتر از دانشگاه صنعتی سیدنی استرالیا و مدرک دکتری خود را در رشته علوم کامپیوتر از دانشگاه ولز بریتانیا در سال ۱۳۵۸ دریافت نمود. وی از سال ۱۳۸۰ در دانشگاه یزد به عنوان عضو هیئت علمی دانشکده مهندسی کامپیوتر مشغول به کار شده و هم‌اکنون در مرتبه دانشیاری مشغول به فعالیت است. زمینه‌های علمی مورد علاقه نام‌برده متنوع بوده و شامل موضوعاتی مانند شبکه‌های سیار و بی‌سیم، محاسبات نرم، یادگیری ماشین، داده‌کاوی و کدگذاری شبکه است.