Islamic Azad University , Shiraz Branch
نشریه تحلیل مدارها، داده ها و سامانه ها
Journal of Circuits, Data and Systems Analysis
sanad.iau.ir/journal/jcdsa

An Efficient Approach for Dynamic IoT service Provisioning on the Fog Infrastructure

Meysam Tekiyehband[1], Mostafa Ghobaei-Arani[2*], Ali Shahidinejad[3]

[1] Department of Computer Engineering, Qo.C., Islamic Azad University, Qom, Iran
m.tekiyehband@iau.ac.ir
[2] Department of Computer Engineering, Qo.C., Islamic Azad University, Qom, Iran
mo.ghobaei@iau.ac.ir
[3] Department of Computer Engineering, Qo.C., Islamic Azad University, Qom, Iran
a.shahidinejad@gmail.com

Abstract: Recent advancements in Internet of Things (IoT) technology have led to its widespread adoption across various domains, such as smart buildings, cities, and healthcare. Fog computing, as a distributed platform at the network edge, enables real-time execution of IoT applications. Due to the dynamic nature of fog environments and the continuously changing behavior of IoT applications, one of the key challenges is the optimal and dynamic provisioning of IoT services over available fog resources. This study aims to address this challenge by proposing a dynamic and optimal service provisioning mechanism using reinforcement learning techniques, such as learning automata. The proposed framework consists of a three-tier architecture (IoT, fog, and cloud layers) and a Dynamic Service Provisioning Manager (DSPM) component. The DSPM comprises three subcomponents: A service request and fog node status monitor, A workload analyzer, and A service provider, which collectively handle service provisioning at the fog layer. The mechanism is evaluated under three distinct scenarios: Analysis of real-world traffic flow, Comparison with optimal solutions, and The impact of service delay thresholds.Simulation results demonstrate that the proposed approach effectively reduces service delay, cost, and delay violation compared to existing mechanisms.

Keywords: Fog Computing, Dynamic Service Provisioning, Learning Automata, IoT Applications, Sevice Delay.

JCDSA, Vol. 3, No. 2, Summer 2025	Online ISSN: 2981-1295	Journal Homepage: https://sanad.iau.ir/en/Journal/jcdsa
Received: 2025-05-09	Accepted: 2025-07-08	Published: 2025-09-22

CITATION

Tekiyehband, M., Ghobaei-Arani, M., Shahidinejad, A.," An Efficient Approach for Dynamic IoT service Provisioning on the Fog Infrastructure", Journal of Circuits, Data and Systems Analysis (JCDSA), Vol. 3, No. 2, pp. 1-22, 2025.
DOI: 00.00000/0000

* Corresponding author

Extended Abstract

1- Introduction

Recent advancements in Internet of Things (IoT) technology have facilitated its widespread adoption across domains such as smart buildings, smart cities, and healthcare. However, the stringent latency requirements of real-time applications, including augmented reality, autonomous vehicles, and industrial robotics, pose significant challenges to traditional cloud computing. Fog computing, as a distributed platform at the network edge, offers a novel solution by reducing latency and enhancing Quality of Service (QoS) for IoT applications. Given the dynamic nature of fog environments and the continuously changes behavior of IoT applications, the optimal and dynamic provisioning of services over available fog resources emerges as a critical issue. This study aims to address this challenge by proposing a dynamic and efficient service provisioning mechanism leveraging reinforcement learning techniques, particularly learning automata. The proposed framework incorporates a three-tier architecture (IoT, fog, and cloud layers) and a Dynamic Service Provisioning Manager (DSPM) component responsible for intelligent resource management.

2- Methodology

The proposed approach is built upon a three-tier architecture comprising the IoT, fog, and cloud layers. The IoT layer includes smart devices such as sensors and gadgets, the fog layer consists of distributed computational nodes, and the cloud layer encompasses powerful servers for heavy processing. The DSPM, the core component of this framework, comprises three subcomponents: a monitoring component for node status and service requests, a workload analyzer, and a service provider. To predict future needs, time series analysis (ARIMA model) is used, and for dynamic decision-making, an algorithm based on learning automata is used. Simulations were conducted using the iFogSim toolkit, evaluating various scenarios with real traffic data (e.g., MAWI archive) and synthetic data (based on Markov chains). The time complexity of the proposed algorithm is formulated as $O(|A|+|F|)$, indicating its suitability for dynamic environments.

To increase the flexibility of the method, the DSPM component uses a feedback-based learning process in which the learning automata dynamically adjusts service placement decisions based on real-world performance metrics such as latency and resource utilization. This adaptive mechanism is optimized through repeated interactions with the fog environment, ensuring optimal resource allocation under varying loads. Also, the workload analyzer improves the accuracy of workload forecasts using the ARIMA model by integrating historical data patterns and enables proactive resource provisioning to mitigate potential bottlenecks.

3- Results and discussion

Performance evaluation was conducted across three scenarios: analysis of real-world traffic, comparison with optimal solutions, and assessment of service delay threshold impacts. The results showed that the proposed method significantly reduced the service delay (close to the optimal method), greatly optimized the cost, and reduced the delay violation to about 3%. Compared to baseline methods such as All-Cloud, Min-Viol, and Fog-Static, the proposed approach exhibits efficiency improvements of 15.99%, 18.21%, and 15.53%, respectively. Traffic prediction via ARIMA, combined with learning automata-based decision-making, enhances load balancing across fog nodes and reduces reliance on the cloud. Increasing the delay threshold also showed that the proposed method still performs better in resource management and cost reduction.

For a deeper analysis, a real-world traffic scenario focusing on varying patterns throughout the day showed that the proposed method, by dynamically adapting, avoids resource congestion and maintains system stability. This adaptation represents a significant advantage, especially compared to the Fog-Static method, which becomes inefficient due to its static nature as the load increases. Also, the cost analysis in the second scenario showed that the proposed method reduced communication costs by up to 20% compared to the All-Cloud method by optimizing the deployment of services in fog nodes close to the devices.

In the third scenario, examining the impact of different latency thresholds showed that the proposed method, by intelligently adjusting resources, even at high thresholds, minimizes latency violations and prevents unnecessary migration of services to the cloud. This is achieved using ARIMA-based forecasting and decision-making based on learning automata. Also, comparing the distribution of services between fog and cloud layers in all scenarios showed that the proposed method uses fog nodes on average 30% more than the baseline methods, which leads to improved response time and reduced load on cloud data centers.

4- Conclusion

The proposed approach, integrating time-series analysis and learning automata, provides an effective solution for the dynamic provisioning of IoT services in fog environments. By optimizing resource utilization, minimizing latency, and improving QoS, it effectively addresses the needs of real-time applications. The computational simplicity and high decision-making speed of learning automata render this method suitable for latency-sensitive environments, thereby enhancing user satisfaction. Simulation results confirm that this framework not only boosts system efficiency but also enables flexible service deployment near IoT devices. Future research directions include integrating deep learning techniques for enhanced prediction accuracy and extending the framework to real-world multi-fog environments.

Islamic Azad University , Shiraz Branch
نشریه تحلیل مدارها، داده ها و سامانه ها
Journal of Circuits, Data and Systems Analysis
sanad.iau.ir/journal/jcdsa

رویکردی کارآمد برای تامین پویای سرویس‌های اینترنت اشیاء روی زیرساخت رایانش مه

میثم تکیه بند[1]، مصطفی قبائی آرانی[1,2]، علی شهیدی نژاد[3]

۱- گروه مهندسی کامپیوتر، واحد قم، دانشگاه آزاد اسلامی، قم، ایران (m.tekiyehband@iau.ac.ir)
۲- گروه مهندسی کامپیوتر، واحد قم، دانشگاه آزاد اسلامی، قم، ایران (mo.ghobaei@iau.ac.ir)
۳- گروه مهندسی کامپیوتر، واحد قم، دانشگاه آزاد اسلامی، قم، ایران (a.shahidinejad@gmail.com)

چکیده: پیشرفت‌های اخیر در فناوری اینترنت اشیاء منجر به گسترش کاربردهای آن در حوزه‌های مختلفی مانند ساختمان‌ها، شهرها، مراقبت‌های بهداشتی و غیره شده است. رایانش مه به عنوان یک پلتفرم توزیع‌شده در لبه شبکه، امکان اجرای برنامه‌های کاربردی اینترنت اشیاء را به صورت بلادرنگ فراهم می‌کند. به دلیل پویایی محیط مه و تغییرات مداوم در رفتار برنامه‌های اینترنت اشیاء، یکی از چالش‌های اصلی، تامین بهینه و پویای سرویس‌های اینترنت اشیاء بر روی منابع موجود در محیط مه است. این مطالعه با هدف ارائه یک مکانیزم تامین سرویس بهینه و پویا، از تکنیک‌های یادگیری تقویتی مانند اتوماتای یادگیر استفاده کرده است. چارچوب پیشنهادی این مطالعه شامل یک معماری سه‌لایه (اشیاء، مه و ابر) و یک مولفه مدیر تامین سرویس پویا است که از سه زیرمولفه مانیتورینگ درخواست‌های سرویس و وضعیت گره‌های مه، تحلیل‌گر بار کاری و ارائه‌دهنده سرویس، تشکیل شده و وظیفه تامین سرویس‌های مورد نیاز در لایه‌ی مه را بر عهده دارد. این مکانیزم در سه سناریوی مختلف ارزیابی شده است که شامل بررسی جریان ترافیک واقعی، مقایسه با راه‌حل‌های بهینه و تاثیر آستانه تاخیر سرویس می‌شود. نتایج شبیه‌سازی نشان می‌دهد که روش پیشنهادی در مقایسه با سایر مکانیزم‌ها، تاخیر سرویس، هزینه و نقض تاخیر را به طور موثری کاهش می‌دهد.

واژه های کلیدی: رایانش مه، تامین پویای سرویس، اتوماتای یادگیر، برنامه‌های کاربردی اینرتنت اشیاء، تاخیر سرویس.

DOI: 00.00000/0000

نوع مقاله: پژوهشی
تاریخ ارسال مقاله: ۱۴۰۴/۰۲/۱۹ تاریخ پذیرش مقاله: ۱۴۰۴/۰۴/۱۷ تاریخ چاپ مقاله: ۱۴۰۴/۰۶/۳۱

۱- مقدمه

بسیاری از برنامه‌های کاربردی اینترنت اشیاء مانند واقعیت افزوده، اتومبیل‌های متصل و خودران، هواپیماهای بدون سرنشین، رباتیک صنعتی، نظارت و تولید در زمان واقعی به الزامات تاخیر سختگیرانه‌ای نیاز دارند. در بعضی موارد این الزامات به زیر ۱۰ میلی‌ثانیه می‌رسد [۱]. این برنامه‌های کاربردی نمی‌توانند تاخیر زیاد و غیرقابل پیش‌بینی ابر را در هنگام استقرار منابع ابری دور از محل تولید داده‌های برنامه کاربردی تحمل کنند. اخیراً رایانش مه [۲]، رایانش لبه [۳] و رایانش لبه موبایل[2] [۴] پیشنهاد شده‌اند تا با قراردادن نزدیکتر منابع محاسبه، ذخیره‌سازی و شبکه به کاربران، تاخیر را کم کرده و پهنای باند شبکه‌های اینترنت اشیاء را کاهش دهند. مه می‌تواند تاخیر، میزان استفاده از پهنای باند و هزینه‌ها را کاهش داده و آگاهی از موقعیت مکانی را فراهم کند و برای برنامه‌های کاربردی حساس به تاخیر به خصوص در مناطق نوظهور مانند اینترنت اشیاء، اینترنت انرژی[3]، شهر هوشمند، پزشکی هوشمند، صنعت نسل ۴[4] و جریان کلان‌داده‌ها[5]، کیفیت سرویس[6] (QoS) را تقویت کند [۵،۶]. یک برنامه کاربردی اینترنت اشیاء ممکن است از چندین سرویس تشکیل شده باشد که می‌توانند در مکان‌های مختلفی اجرا شوند. به عنوان مثال یک برنامه کاربردی ممکن است سرویس‌هایی مثل تایید هویت، فایروال، حافظه نهان و رمزنگاری داشته باشد. برخی از سرویس‌های برنامه کاربردی، حساس به تاخیر هستند و آستانه‌ی تحمل تاخیر سختی دارند و ممکن است نیاز باشد تا نزدیک‌تر به کاربران یا منابع داده (به عنوان مثال سرویس حافظه نهان) اجرا شوند (برای مثال در لبه شبکه تا در دستگاه‌های رایانش مه). از طرف دیگر، برخی سرویس‌های برنامه‌های

[1] نویسنده مسئول

[2] Mobile Edge Computing (MEC)
[3] Internet Of Energy
[4] Industry 4.0
[5] BigData Stream
[6] Quality of Service

کاربردی، تحمل‌پذیری خطا را دارند و نیاز به دسترس‌پذیری بالایی دارند و ممکن است دورتر از کاربران یا منابع داده در امتداد زنجیره مه به ابر یا در ابر مستقر شوند (مانند سرویس تایید هویت). این سرویس‌های اینترنت اشیاء با قابلیت مه که می‌توانند بر روی دستگاه‌های رایانش مه یا سرورهای ابری اجرا شوند، به عنوان سرویس‌های مه نام‌گذاری می‌شوند. تامین پویای چنین سرویس‌هایی در دستگاه‌های رایانش مه (گره‌های مه) یا سرورهای ابری تاثیر بسزایی در استفاده از شبکه و تأخیر انتها به انتها دارد. اگرچه رایانش ابری به عنوان راه‌حل برتر برای الگوهای استفاده از منابع پویا به نظر می‌رسد، اما قادر به تامین نیازهای تأخیر فوق‌العاده کم برنامه‌های کاربردی اینترنت اشیاء خاص، ارائه خدمات آگاهی از موقعیت مکانی یا مقیاس‌پذیری با بزرگی داده‌هایی که برنامه‌های کاربردی اینترنت اشیاء تولید می‌کنند، نیست[۷]. سرویس‌های مه می‌توانند برای فراهم‌کردن تأخیر کم و قابل پیش‌بینی بر روی گره‌های مه یا برای فراهم‌کردن دسترس‌پذیری بالا با هزینه استقرار کم در ابر مستقر شوند.

یکی از مسائل مهم در رایانش مه، چگونگی توزیع سرویس‌های اینترنت اشیاء بر روی منابع موجود در مه، با توجه به الزامات مشخص‌شده توسط کاربران (مانند کیفیت سرویس، هزینه‌ی اجرا، انرژی مصرفی و غیره) می‌باشد. این مساله به عنوان مساله تامین سرویس مه[1] (FSPP) ، شناخته می‌شود و هدف آن یافتن نگاشتی بهینه بین برنامه‌های کاربردی اینترنت اشیاء و منابع محاسباتی مه است، به گونه‌ای که الزامات مشخص‌شده کاربران نیز رعایت شود. اینکه کدام سرویس‌ها به صورت محلی روی منابع مه و کدام سرویس‌ها روی منابع ابر مستقر شود نیز یکی دیگر از وظایف این مساله است. نگاشت بین سرویس‌های برنامه کاربردی و منابع محاسباتی مه، می‌تواند چند به چند باشد، یعنی ممکن است یک سرویس در یک یا چند گره مه مستقر شود یا یک گره مه، میزبان یک یا چند سرویس مختلف باشد. تامین سرویس‌های مه می‌تواند به صورت ایستا انجام شود (به عنوان مثال با هدف به‌حداقل‌رساندن هزینه کل، درحالی‌که محدودیت‌های تاخیر مربوطه را برآورده می‌کند). با این وجود، از آنجا که الگوی استفاده از منابع برنامه‌های کاربردی اینترنت اشیاء خاص، پویا بوده و با گذشت زمان تغییر می‌کند، یک تامین سرویس ایستا قادر به سازگاری با چنین تغییراتی نخواهد بود. بنابراین سرویس‌های مه باید به صورت پویا تامین شوند تا برنامه‌های کاربردی اینترنت اشیاء مبتنی بر مه، از منابع مه به شکل بهینه استفاده کنند. این ذات تامین پویای سرویس‌های مه است که سرویس‌های اینترنت اشیاء را بر روی دستگاه‌های رایانش مه یا سرورهای ابری مستقر یا رهاسازی می‌کند تا هزینه منابع را به حداقل برساند و با محدودیت‌های تاخیر و کیفیت سرویس (QoS) برنامه‌های کاربردی اینترنت اشیاء مواجه شود[۷]. تامین پویای سرویس مه[2] (DFSP) یک کار آنلاین است که هدفش ارائه پویای (استقرار[3] یا رهاسازی[4]) سرویس‌ها روی گره‌های مه، به منظور مطابقت با الزامات کیفیت سرویس و در عین حال به‌حداقل‌رساندن هزینه منابع می‌باشد. بنابراین مساله تامین پویای سرویس به صورت بهینه در رایانش مه با هدف استفاده حداکثری از منابع مه و کاهش زمان اجرای نرم‌افزارها چالش مهمی می‌باشد. لذا در این پژوهش به بررسی این مساله پرداخته و به دنبال ارائه روشی بهینه در تامین پویای سرویس‌های اینترنت اشیاء در محیط مه هستیم.

سازمان‌دهی ادامه مقاله به‌صورت زیر می‌باشد: در بخش دوم، تحقیقات در زمینه تامین سرویس‌های اینترنت اشیاء در محیط مه بررسی می‌شود. در بخش سوم، روش پیشنهادی برای تامین پویای سرویس‌های اینترنت اشیاء در محیط مه با استفاده از تکنیک اتوماتای یادگیر ارائه می‌گردد. در بخش چهارم، نتایج عملکرد روش پیشنهادی در قالب نتایج شبیه‌سازی مورد ارزیابی قرارگرفته و با روش‌های مرجع مقایسه می‌شود. درنهایت، در بخش آخر نتیجه‌گیری مطرح می‌گردد.

۲- مبانی نظری و پیشینه پژوهش

رایانش مه با وجود مزیت‌های فراوانی که دارد با چالش‌هایی نیز روبرو است که از آن جمله می‌توان به تأمین سرویس اشاره کرد. تأمین سرویس یک بحث بسیار مهم در رایانش مه می‌باشد. در این بخش، ما مساله تامین سرویس در مه را شرح داده و مکانیسم‌های مختلف برای تامین پویای سرویس‌ها در محیط رایانش مه/لبه را بررسی می‌کنیم. سپس، کارهای مختلف را برای حل مساله تامین سرویس پویا به صورت خلاصه بیان می‌کنیم.

۲-۱- مساله تامین سرویس مه

برنامه‌های کاربردی اینترنت اشیاء ممکن است در زمان‌ها و مکان‌های مختلف، درخواست‌های متفاوتی داشته باشند و در طول زمان حجم بار کاری آنها تغییر کند. برای مثال ممکن است یک برنامه کاربردی در ابتدای شروع خود به منابع کمتری نیاز داشته باشد ولی در طول اجرا دائماً منابع موردنیازش تغییر کند. در نتیجه نحوه استفاده از منابع ابر یا مه توسط برنامه‌های کاربردی اینترنت اشیاء در زمان‌ها و مکان‌های مختلف به صورت پویا می‌باشد. این پویا بودن استفاده از منابع، ما را با چالش‌هایی روبرو می‌کند. استقرار یک برنامه کاربردی در یک گره مه با منابع قوی در شروع اجرای برنامه کاربردی ممکن است باعث بلااستفاده ماندن منابع آن گره در ادامه‌ی اجرای برنامه شود. ضمن اینکه با استقرار این برنامه در گره موردنظر ممکن است دیگر نتوانیم برنامه جدیدی را چنانچه به منابع زیادی جهت اجرا نیاز داشته باشد، اجرا کنیم. حتی ممکن است در طول زمان اجرای یک برنامه کاربردی در شرایطی قرار گیریم که کلاً لایه‌ی مه جوابگوی آن نباشد و مجبور باشیم تا برنامه کاربردی را جهت اجرا به لایه‌ی ابر ارسال کنیم. چنانچه برنامه را در یک گره مه با منابع ضعیف مستقر کنیم نیز ممکن است در ادامه‌ی اجرا به منابع بیشتری احتیاج داشته باشد و از آنجا که گره

[1] Fog Service Provisioning Problem
[2] Dynamic Fog Service Provisioning
[3] Deploy
[4] Release

موردنظر دیگر نمی‌تواند جوابگوی احتیاجات برنامه باشد، می‌بایست برنامه را برای ادامه به گره دیگری منتقل کنیم. این کار نیز چالش‌های خود را دارد. در نتیجه استقرار ایستای برنامه‌ها بر روی گره‌های مه عملکرد مناسبی به همراه ندارد و می‌بایست به روشی پویا عمل کنیم. برنامه‌های کاربردی اینترنت اشیاء از سرویس‌های مختلفی تشکیل می‌شوند. برای مثال یک برنامه کاربردی ممکن است شامل سرویس‌های فایروال، تایید هویت، حافظه نهان، رمزنگاری و غیره باشد. هر یک از این سرویس‌های مربوط به برنامه‌های کاربردی اینترنت اشیاء می‌توانند در محل‌های متفاوتی اجرا شوند. برای مثال ممکن است سرویس موردنیاز یک برنامه در یکی از سرورهای ابر و یا در هر یک از گره‌های مه قابلیت اجراشدن داشته باشد. برخی از این سرویس‌ها می‌بایست با سرعت بالا اجرا شوند، به عبارت دیگر این سرویس‌ها به تأخیر حساس هستند. سرویس‌های حساس به تاخیر می‌بایست در نزدیکی محل درخواست سرویس و با فاصله کمی نسبت به کاربر و دستگاه‌های اینترنت اشیاء اجرا شوند، مثلاً در لبه شبکه یا در دستگاه‌های موجود در لایه‌ی مه، تا با تأخیر کمتری اجرا شده و الزامات آستانه تحمل تاخیر مشخص‌شده توسط کاربر یا برنامه کاربردی نیز برآورده شود. برخی دیگر از سرویس‌های برنامه‌های کاربردی، آستانه تحمل تاخیر سختی ندارند و اصطلاحاً نسبت به خطای تاخیر تحمل پذیرند. این نوع از سرویس‌های اینترنت اشیاء می‌توانند در مکان‌های دورتر از لایه‌ی مه یا ابر مستقر شوند. اگرچه استقرار سرویس‌ها در لایه‌ی ابر به دلیل ماهیت پویای استفاده از منابع در آنها به نظر یک راه‌حل مناسب می‌باشد اما استقرار سرویس‌های دارای الزامات تاخیر بسیار کم و سرویس‌هایی که به ناچار می‌بایست در نزدیکی کاربران اجرا شوند، در لایه‌ی ابر ممکن نیست و می‌بایست از گره‌های موجود در لایه‌ی مه نیز استفاده کنیم. لذا سرویس‌های اینترنت اشیاء می‌توانند برای فراهم‌کردن دسترس‌پذیری بالا با هزینه استقرار کم، در لایه‌ی ابر مستقر شوند یا اینکه برای فراهم‌کردن تأخیر کم بر روی گره‌های مه مستقر شوند.

یکی از چالش‌های اساسی در رایانش مه چگونگی استقرار یا توزیع سرویس‌های اینترنت اشیاء با توجه به الزامات مشخص‌شده توسط کاربران یا برنامه‌های کاربردی می باشد. این چالش به مساله تامین سرویس مه معروف است و هدف آن پیداکردن بهترین نگاشت ممکن بین سرویس‌های اینترنت اشیاء و گره‌های مه با توجه به الزامات مشخص‌شده توسط کاربران می‌باشد. یکی دیگر از وظایف این مساله اینست که مشخص می‌کند کدام سرویس‌ها بر روی سرورهای ابری و کدام سرویس‌ها به صورت محلی و بر روی گره‌های مه می‌بایست مستقر شوند. استقرار پویای چنین سرویس‌هایی در سرورهای ابری و گره‌های مه به شدت در میزان استفاده از منابع شبکه و تأخیر اجرای برنامه‌های کاربردی موثر است. در مساله تامین سرویس مه، یک سرویس مهم ممکن است بر روی یک یا چند گره مه مستقر شود یا اینکه ممکن است در یک گره مه، یک یا چند سرویس مه متفاوت ارائه شود. لذا این مساله در واقع یک نگاشت چند به چند است. از آنجا که محیط رایانش مه و ابر ماهیتا سلسله مراتبی بوده و منابع آن ناهمگن و توزیع‌شده هستند، مساله استقرار سرویس‌های برنامه‌های کاربردی اینترنت اشیاء یک مساله چالش‌برانگیز بوده و ویژگی‌های مختلف دستگاه‌های اینترنت اشیاء و انتظارات مختلف کاربران نیز چالش‌های این مساله را بیشتر می‌کند. اجرای ایستای مساله تامین سرویس‌های مه ممکن است برخی از محدودیت‌های تاخیر برنامه‌های کاربردی را برآورده کند و در کل هزینه اجرای الگوریتم را کاهش دهد، اما با توجه به اینکه الگوی برنامه‌های کاربردی اینترنت اشیاء خاص در استفاده از منابع محاسباتی مه و ابر پویا می‌باشد و در طول زمان متغیر است لذا تأمین سرویس‌های مه می‌بایست به صورت پویا انجام شود، زیرا اجرای این مساله به صورت ایستا در قبال این تغییرات انعطاف‌پذیر نیست و باعث افزایش هزینه تاخیر در کل سیستم می‌شود. با اجرای پویای این مساله استفاده از منابع مه و ابر توسط برنامه‌های کاربردی اینترنت اشیاء به شکلی بهینه انجام می‌شود و هزینه استفاده از منابع کاهش پیدا می‌کند. در ضمن الزامات تاخیر برنامه‌های کاربردی حساس به تأخیر بیشتر رعایت شده و باعث افزایش کیفیت سرویس می‌شود.

۲-۲- کارهای مرتبط

در این قسمت از پژوهش ما مکانیسم‌های مختلف ارائه شده توسط سایر پژوهشگران برای تامین پویای سرویس‌ها در محیط رایانش مه/لبه را به طور خلاصه بررسی می‌کنیم. یوسف‌پور و همکاران [۷] یک چارچوب ارائه سرویس پویا برای استقرار و آزادسازی برنامه‌های اینترنت اشیاء در گره‌های مه معرفی کرده‌اند. آنها یک فرمول‌بندی برنامه‌نویسی خطی برای مساله ارائه سرویس پویای آگاه به کیفیت سرویس (QoS) ارائه کرده‌اند و دو استراتژی ابتکاری حریصانه را برای کنترل ارائه سرویس به صورت پویا پیشنهاد داده‌اند. روش‌های پیشنهادی آنها گره‌های مه را با توجه به نرخ ترافیک ورودی مرتب می‌کنند تا نقض تاخیر و هزینه کل را به حداقل برسانند. علاوه بر این، آنها استراتژی‌های پیشنهادی خود را بر روی گزارشات ترافیک در دنیای واقعی ارزیابی کرده و نشان داده‌اند که راه‌حل آنها از نظر تأخیر سرویس با فواصل زمانی و آستانه‌های تحمل متفاوت در مقایسه با سایر استراتژی‌ها بهتر عمل می‌کند. مارتین و همکاران [۸] روشی مستقل برای انتقال ظروف بین گره‌های مه برای برآوردن الزامات کیفیت سرویس کاربر طراحی کرده‌اند. رویکرد آنها از حلقه Monitor-Analize-Plan-Execute (MAPE) برای پشتیبانی از تحرک کاربر برای حل مساله استقرار سرویس پویای محیط رایانش مه استفاده می‌کند. آن‌ها مشکل خود را با استفاده از یک مدل برنامه‌ریزی خطی فرموله‌بندی کرده‌اند و یک الگوریتم ژنتیک را برای تعیین گره‌های مه مقصد برای مهاجرت ظروف مربوط به برنامه اینترنت اشیاء پیشنهاد کرده‌اند. علاوه براین، آنها یک چارچوب آگاه از تحرک الهام‌گرفته از مدل محاسبه خودکار برای مهاجرت ظروف به منابع مه تولید کرده‌اند. مدهوسودهان و همکاران

در [۹]، یک الگوریتم الهام‌گرفته از شبکه‌های عصبی مصنوعی[1] مبتنی بر الگوریتم مورچه‌شب‌پره[2] را برای هماهنگی وظایف در محیط‌های لبه پیشنهاد می‌دهند. هدف این الگوریتم، بهبود بهره‌وری منابع و کاهش مصرف انرژی است. تحلیل مقایسه‌ای با الگوریتم‌های مختلف نشان می‌دهد که الگوریتم پیشنهادی آن‌ها بار را در لایه‌ی لبه متعادل می‌کند، که نتیجه آن کاهش بار بر روی ابر، بهبود مصرف توان، بهره‌وری CPU، بهره‌وری شبکه و کاهش زمان انتظار متوسط برای درخواست‌ها است. آن‌ها مدل پیشنهادی خود را در محیط محاسبات لبه برای کاربردهای حوزه سلامت آزمایش کرده‌اند. ارزیابی‌های آن‌ها نشان می‌دهد که الگوریتم پیشنهادی نسبت به الگوریتم‌های موجود مبتنی بر منطق فازی و الگوریتم Round Robin عملکرد بهتری از نظر توان مصرفی، بهره‌وری CPU، بهره‌وری شبکه و زمان انتظار متوسط درخواست‌ها دارد. ابوفتحی و همکاران در [۱۰] روشی مبتنی بر اتوماتای یادگیر توزیع‌شده پیشنهاد می‌کنند که با استفاده از ظرفیت حداکثری مه‌ها در یک شبکه‌ی مه ناهمگن، فضای جستجو را کاهش می‌دهد. در این روش، توپولوژی مه به یک اتوماتای یادگیر توزیع‌شده نگاشت می‌شود. با همکاری اتوماتا مشکل قرارگیری ماژول‌ها حل شده است تا مصرف انرژی و تأخیر برنامه‌ها کاهش یابد. برای ارزیابی میزان مصرف انرژی و زمان اجرای برنامه‌های اینترنت اشیاء، دو تابع هزینه تک‌هدفی برای انرژی و تأخیر، و یک تابع تک‌هدفی دیگر با درنظرگرفتن همزمان انرژی و تأخیر به کار گرفته شده‌اند. نتایج آن‌ها نشان می‌دهد که کارایی میانگین روش پیشنهادی نسبت به سایر روش‌ها بهبود یافته است.

در [۱۱]، لی و همکارانش، چارچوب AMPHI را ارائه می‌دهند که یک سیستم سازگارپذیر برای تأمین میکروسرویس‌ها می‌باشد و امکان اجرای انعطاف‌پذیر سرویس‌های کانتینری در محیط‌های ترکیبی ثابت-متحرک را فراهم می‌کند. AMPHI با بهره‌گیری از تنوع هزینه و عملکرد اپراتورهای مختلف، از تکنیک‌های پیش‌بینانه و بلادرنگ برای انتخاب هوشمند، استقرار و اشتراک‌گذاری اپراتورها در میان دستگاه‌های ترکیبی (حسگرهای ثابت، پهپادها و ربات‌های متحرک) استفاده می‌کند تا کیفیت خدمات (QoS) و بهره‌وری منابع را در شرایط پویا به حداکثر برساند. ارزیابی AMPHI در سناریوی آتش‌نشانی هوشمند با استفاده از حسگرهای ساختمانی و واحدهای متحرک هوایی، انعطاف‌پذیری و مقرون‌به‌صرفه بودن آن را در اجرای گردش کارهای پویای IoT نشان می‌دهد. هوکیو و همکارانش در [۱۲]، چارچوب IoT-as-a-Service (IoTaaS) مبتنی بر پهپاد را پیشنهاد می‌دهند که امکان استقرار پویای گره‌های IoT و دروازه‌های وب را از طریق پهپادها فراهم می‌کند. این راه‌حل با توزیع هوشمند دستگاه‌ها براساس نیاز کاربر و ارائه‌ی سرویس ابری توزیع‌شده، هزینه‌های راه‌اندازی را کاهش می‌دهد. تحلیل اقتصادی و پیاده‌سازی عملی آن‌ها در دو سناریوی کشاورزی هوشمند و پایش آلودگی هوا نشان می‌دهد IoTaaS نه‌تنها صرفه‌جویی هزینه‌ای دارد، بلکه بهره‌وری استفاده از دستگاه‌های اینترنت اشیاء را افزایش می‌دهد. کوین و همکارانش [۱۳]، یک مدل برنامه‌ریزی خطی عدد صحیح مختلط برای حل مسئله استقرار سرویس‌های مه ارائه می‌دهند که به صورت چندهدفه و با اهداف حداقل‌سازی تخلف از مهلت زمانی، تلفات منابع و هزینه سرویس‌ها و حداکثرسازی استفاده از منابع طراحی شده است. همچنین یک الگوریتم ژنتیک نخبه‌گرا با معماری موازی اشتراکی (EGASPE) برای حل این مسئله پیشنهاد داده‌اند که توزیع کارآمد منابع تحت الزامات کیفیت سرویس را در نظر گرفته و به تعادل مناسبی بین اکتشاف و بهره‌برداری دست می‌یابد. الگوریتم پیشنهادی آن‌ها با درنظرگرفتن مهلت زمانی سرویس‌های اینترنت اشیاء، امکان ذخیره‌ی منابع برای درخواست‌های آینده را فراهم می‌کند.

شخار و همکارانش، در [۱۴] یک رویکرد مبتنی بر میان‌افزار برای مدیریت پویای منابع در بین لایه‌های لبه، مه و ابر ایجاد کرده‌اند. رویکرد آن‌ها گره مه مناسب را برای سرویس به برنامه‌های اینترنت اشیاء برای به‌حداقل‌رساندن نقض SLO در حالی که الزامات تاخیر آنها را برآورده می‌کند، انتخاب می‌کند. همچنین، آن‌ها اثربخشی راه‌حل خود را با استفاده از دو برنامه اینترنت اشیاء تلفن‌همراه در دنیای واقعی ارزیابی کرده‌اند و نشان داده‌اند که از نظر برآورد تأخیر و معیارهای عملکرد دقت در مقایسه با روش‌های مرتبط مربوطه، بهتر عمل می‌کند. ابینایا و همکارانش در [۱۵]، به مسئله استقرار سرویس‌های توابع مجازی شبکه در محیط‌های پویای رایانش لبه‌ای همراه (MEC) می‌پردازند. آن‌ها برای تعادل‌بخشی به نیازهای کاربران در شرایط متغیر شبکه، یک سیستم تأمین پویای خدمات مبتنی بر الگوریتم ترکیبی بهینه‌سازی ازدحام ذرات-یادگیری تقویتی (PSO-RL) پیشنهاد می‌دهند. چارچوب ارائه‌شده آن‌ها از مراحل متوالی بهینه‌سازی سودمندی با الگوریتم کرم‌های شب‌تاب، بهینه‌سازی بلادرنگ توان عملیاتی با تکنیک‌های کاهش گرادیان تصادفی و تبرید شبیه‌سازی‌شده، و انطباق با الگوهای تحرک کاربران و الزامات تأخیر سرویس تشکیل شده است که بهبودهای قابل توجهی در معیارهای توان عملیاتی، بهره‌وری منابع و تأخیر انتها به انتها داشته است. امسدی و همکارانش در [۱۶]، مسائل ارائه سرویس و جایابی ظروف را در محیط مه پویا مورد مطالعه قرار داده‌اند. آن‌ها پراکندگی و تحرک گره‌های مه را در نظر گرفته‌اند و مساله خود را به عنوان یک فرمول برنامه‌نویسی خطی بیان کرده‌اند و از یک روش فرابتکاری مبتنی بر بهینه‌سازی ازدحام ذرات و یک روش ابتکاری مبتنی بر الگوریتم‌های حریصانه برای حل مساله بهینه‌سازی خود استفاده کرده‌اند. نتایج شبیه‌سازی آن‌ها با استفاده از IBM CPLEX Optimizer بر روی فایل‌های ثبت وقایع مربوط به تحرکات واقعی سیستم کنار جاده‌ای هوشمند به دست آمده است که نشان می‌دهد روش پیشنهادی مبتنی بر فرااکتشافی راه‌حل‌های تقریباً مطلوب را بدست می‌آورد، در حالی که زمان اجرا، طولانی‌تر از روش ابتکاری مبتنی بر حریصانه است. دوناسولو و همکارانش در [۱۷]، رویکردی برای ارائه سرویس اینترنت اشیاء مبتنی

[1] Artificial Neural Network

[2] Antlion Algorithm

بر بار بر روی زیرساخت مه ناهمگن پیشنهاد کرده‌اند. رویکرد آن‌ها از یک روش مبتنی بر روش‌های جستجوی تطبیقی تصادفی حریصانه برای بهینه‌سازی جایابی سرویس‌های اینترنت اشیاء برای به حداقل‌رساندن هزینه استقرار و در عین حال برآوردن سهم بار بین منابع مه استفاده می‌کند. علاوه براین، برای ارزیابی رویکرد پیشنهادی خود، آنها از یک سیستم هماهنگ‌کننده به نام FITOR استفاده کرده‌اند که یک اکوسیستم مه واقعی را ایجاد می‌کند و نشان می‌دهد که از نظر نرخ گذردهی، هزینه تأمین سرویس و تأخیر سرویس، عملکرد بهتری در مقایسه با روش‌های مرتبط دارد.

تران و همکارانش [۱۸]، مکانیزمی برای جایابی سرویس‌های اینترنت اشیاء آگاه به سابقه در چشم‌انداز مه پیشنهاد کرده‌اند. مکانیسم پیشنهادی آنها از ویژگی‌های آگاه به سابقه مانند میزان استفاده از منابع، موقعیت مکانی، میزان استفاده از شبکه برای به‌حداقل‌رساندن مصرف انرژی، هزینه و تأخیر استفاده می‌کند. ویژگی‌های آگاه به سابقه در شناسایی محلی که سرویس اینترنت اشیاء در زیرساخت مه باید در آن مستقر شود، کمک می‌کند. علاوه براین، آن‌ها مجموعه‌ای از آزمایش‌ها را با استفاده از حل‌کننده CPLEX بر روی برنامه شهر هوشمند در شهر هوشی‌مین انجام داده‌اند تا استحکام مکانیسم پیشنهادی را اثبات کرده و نشان دهند که از منابع مه، استفاده حداکثری می‌کند. لباوجی و همکاران در [۱۹]، یک رویکرد زمان‌بندی پویا و مقاوم در برابر خرابی مبتنی بر یادگیری پیشنهاد می‌کنند که سه ویژگی ذاتی محیط‌های مه (پویایی، ناهمگنی و عدم قطعیت) را در نظر می‌گیرد تا قابلیت اطمینان خدمات مه برای وظایف اینترنت اشیاء با محدودیت زمانی را بهبود بخشد. این رویکرد یک راه‌حل ترکیبی برای تحمل‌پذیری خرابی ارائه می‌دهد که بر پایه‌ی رویکردهای پیش‌فعال، واکنشی و تکراری برای مدیریت خرابی‌ها استوار است. همچنین، این روش زمان اجرا و آگاهی از مصرف انرژی را به‌صورت پویا در فرآیند تصمیم‌گیری ادغام می‌کند تا چالش‌های مرتبط با پویایی، عدم قطعیت‌ها و مصرف انرژی در محیط رایانش مه را برطرف سازد. آن‌ها برای اعتبارسنجی عملکرد رویکرد پیشنهادی خود زمان‌بندی مقاوم در برابر خرابی، تحلیل تجربی گسترده‌ای انجام داده‌اند و آن را با چندین روش پایه مقایسه کرده‌اند. رویکرد پیشنهادی زمان‌بندی وظایف آن‌ها، در مقایسه با روش‌های پایه، قابلیت اطمینان خدمات مه، کارایی انرژی و بهره‌وری منابع را بهبود می‌بخشد. در [۲۰]، اویانگ و همکارانش یک استراتژی جایابی سرویس سطح کاربر انطباقی جدید با توجه به رتبه‌بندی ترجیحات کاربر پیشنهاد داده‌اند. آن‌ها یک مساله جایابی سرویس تطبیقی را در قالب یک مدل multi-armed bandit زمینه‌ای تدوین کرده‌اند و از تکنیک یادگیری آنلاین نمونه‌گیری تامپسون برای ارائه تصمیمات کارآمد در زمینه جایابی سرویس استفاده کرده‌اند. علاوه براین، آنها مجموعه‌ای از آزمایشات گسترده را برای نشان‌دادن استحکام راه‌حل پیشنهادی خود از نظر هزینه مهاجرت طراحی کرده‌اند و آن را با استراتژی‌های آفلاین و آنلاین مقایسه کرده‌اند.

کیو و همکارانش در [۲۱]، ارائه سرویس انعطاف‌پذیر برای خرابی‌های نامشخص سرویس در محیط رایانش لبه را مورد بحث قرار داده‌اند. در واقع، مساله آن‌ها مکانیزم جایابی سرویس را برای حداکثرسازی مطلوبیت موردانتظار کلی با استقرار سرویس‌های اینترنت اشیاء در گره‌های لبه تعیین می‌کند. آن‌ها مساله خود را در قالب مساله بهینه‌سازی حداقل-حداکثر [1] تنظیم کرده‌اند و دو روش ابتکاری مبتنی بر راهبرد حریصانه را برای دستیابی به نسبت تقریبی در زمان چند جمله‌ای پیشنهاد کرده‌اند. در [۲۲]، شن و همکارانش، جایابی سرویس پویا برای اینترنت برنامه‌های کاربردی وسایل نقلیه در اکوسیستم رایانش لبه را مطالعه کرده‌اند. آن‌ها از تکنیک الگوریتم ژنتیک مرتب‌سازی غیرمسلط ۳ [2] برای دستیابی به عملکرد بهتر استفاده کرده‌اند. علاوه براین، آن‌ها از تکنیک‌های خوشه‌بندی برای مقداردهی اولیه جمعیت برای دستیابی به سرعت همگرایی و دقت بیشتر استفاده کرده‌اند. نتایج شبیه‌سازی آن‌ها نشان می‌دهد که راه‌حل پیشنهادی آن‌ها از نظر هزینه بازسازی و تاخیر در مقایسه با روش‌های موجود جایابی سرویس بهتر عمل می‌کند. دانگ و همکارانش [۲۳]، چارچوبی را برای به‌حداقل‌رساندن تأخیر در ارائه سرویس‌های اینترنت اشیاء در اکوسیستم‌های مه-ابر طراحی کرده‌اند. راه حل آنها از تصمیم‌گیری تخلیه کار تطبیقی در قالب تخلیه کامل کار در گره‌های مه، تخلیه کامل کار در سرورهای ابری و تخلیه وظایف مشارکتی در اکوسیستم های مه-ابر استفاده می‌کند. علاوه براین، راه‌حل آن‌ها از تکنیک بهینه‌سازی ازدحام ذرات برای نگاشت کارهای محاسباتی فشرده در سرورهای مه و ابر استفاده می‌کند. روی و همکارانش در [۲۴]، یک مکانیسم ارائه سرویس مه پویا را برای برنامه‌های اینترنت اشیاء خودرو در شبکه‌های لبه پیشنهاد کرده‌اند. مکانیسم پیشنهادی آن‌ها نسبت ظرفیت سرورهای لبه را محاسبه می‌کند. سپس از نظریه مارکوف برای تعیین سرور لبه مناسب و تقسیم بار بین سرورهای لبه براساس قابلیت منابع، سود و اعتبار آن‌ها استفاده می‌کند. علاوه براین، آن‌ها از ضریب لاگرانژی برای تنظیم مجدد تابع هدف برای دستیابی به مقدار بهینه استفاده از حافظه سرورهای لبه استفاده کرده‌اند.

در مطالعه دیگری از روی و همکارانش [۲۵]، یک الگوریتم نگاشت پرس‌وجوی تصمیم‌گیری پویا را برای تعیین حداقل تعداد گره‌های لبه موردنیاز برای تصمیم‌گیری توسعه داده‌اند. علاوه براین، آن‌ها تصمیمات مربوط به ایمنی را به پرسش‌های تصمیم‌گیری اضطراری و غیراضطراری طبقه‌بندی می‌کنند. رادهیکا و همکارانش [۲۶]، یک استراتژی تأمین منابع پویای مقرون به صرفه در بین ارائه‌دهندگان ابر در محیط ابر ترکیبی پیشنهاد کرده‌اند. استراتژی پیشنهادی آن‌ها ارائه‌دهندگان ابر مناسب را با استفاده از یک فرآیند سلسله مراتبی تحلیلی فازی براساس الزامات کیفیت سرویس برای هر نوع حجم کار انتخاب می‌کند. نتایج عددی آن‌ها نشان می‌دهد که استراتژی پیشنهادی آن‌ها هزینه و زمان تأمین را در مقایسه با سایر استراتژی‌ها کاهش می‌دهد. به طور کلی، در اکثر کارهای تحقیقاتی در مورد مساله

[1] min-max

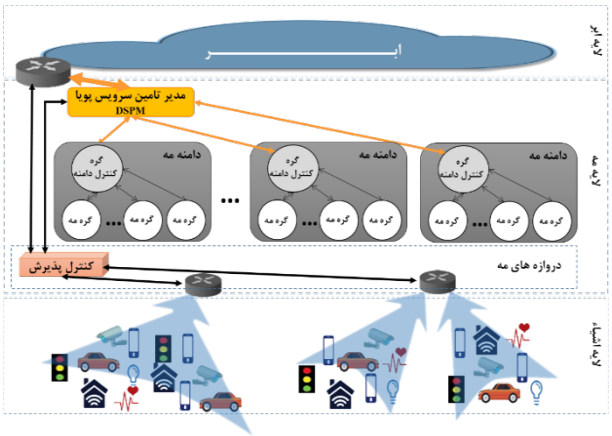
[2] non-dominated sorting genetic algorithm 3

ارائه سرویس مه برای به‌حداقل‌رساندن هزینه کل، جایابی سرویس‌های مه به‌صورت ایستا انجام می‌شود. اما از آنجا که الگوی استفاده از منابع در برخی از برنامه‌های اینترنت اشیاء پویا است و در طول زمان تغییر می‌کند، ممکن است استراتژی‌های ایستا در چنین شرایطی مناسب نباشد. بنابراین، سرویس‌های مه باید به صورت پویا ارائه شود تا نتایج بهتری در محیط‌های پویای مه داشته باشد. اگرچه برخی از مطالعات قبلی بر ارائه سرویس پویا بر اساس معیارهای کیفیت سرویس متمرکز شده‌اند، اما هنوز تلاش بیشتری برای رسیدگی موثر به این مسئله موردنیاز است. بنابراین، در این پژوهش، ما از تکنیک‌های پیش‌بینی (به عنوان مثال سری‌های زمانی) برای پیش‌بینی بار کاری اینترنت اشیاء با تئوری اتوماتای یادگیر برای استقرار سرویس‌های اینترنت اشیاء در گره‌های مه یا سرورهای ابری برای به‌حداقل‌رساندن هزینه کل و تأخیر و برآوردن الزامات کیفیت سرویس سرویس‌های اینترنت اشیاء استفاده کرده‌ایم. با توجه به بررسی و خلاصه مکانیزم‌های پویا برای جایابی سرویس، مقایسه آنها از نظر ابزار ارزیابی، تکنیک‌های مورداستفاده، معیارهای عملکرد و استراتژی استقرار (به عنوان مثال، متمرکز و غیرمتمرکز)، و همچنین مطالعه موردی هر مکانیسم در جدول (۱) نشان داده شده است.

۳- روش‌شناسی پژوهش

در این پژوهش سعی شده مقالات و پژوهش‌های جدید، معتبر و مرتبط با موضوع رایانش مه و مساله‌ی تامین سرویس در مه، مورد مطالعه و بررسی قرار گرفته و انواع روش‌های تامین سرویس در مه به همراه مزایا و معایب آن‌ها مشخص و مقایسه شود تا چالش‌ها و مسائل مهم در رابطه با این موضوع، تعیین شود. در ادامه، چارچوب، مدل و الگوریتم پیشنهادی بر پایه‌ی اتوماتای یادگیر، برای حل این مساله معرفی و در نهایت شبیه‌سازی آن انجام شده و به تجزیه وتحلیل نتایج حاصل از شبیه‌سازی پرداخته شده است. برای ایجاد سناریوها و تجزیه‌وتحلیل نتایج، از شبیه‌ساز iFogSim استفاده شده است. iFogSim مدل‌سازی و شبیه‌سازی محیط‌های رایانش مه را برای ارزیابی مدیریت منابع و برنامه‌های زمان‌بندی در میان منابع لبه و ابر در چندین سناریو، براساس تأثیرات آن‌ها در تأخیر، مصرف انرژی، تراکم شبکه و هزینه‌های عملیاتی، امکان‌پذیر می‌سازد. همچنین معیارهای عملکرد را اندازه‌گیری کرده و دستگاه‌های لبه، مراکز داده، سنسورها، لینک‌های شبکه، جریان داده‌ها و برنامه‌های جریان پردازش را شبیه‌سازی می‌کند.

همان طور که در شکل (۱) نشان داده شده است ما معماری سه لایه‌ای را برای اشیاء، مه و ابر در نظر می‌گیریم. در پایین‌ترین لایه که لایه‌ی اینترنت اشیاء نام دارد تمامی حسگرها و اشیاء هوشمند نظیر تبلت، موبایل و ... قرار دارد. در لایه‌ی مه دامنه‌های مه قرار دارند. این لایه، بین دو لایه‌ی بالایی (لایه‌ی ابر) و لایه‌ی پایینی (لایه‌ی دستگاه‌های اینترنت اشیاء) قرار دارد. اطلاعات مختلف توسط حسگرهای لایه پایینی جمع‌آوری شده و به لایه‌ی میانی (لایه‌ی مه) فرستاده می‌شود. در لایه‌ی مه، درخواست‌های دریافت‌شده از لایه‌ی پایین‌تر به‌صورت هوشمند بررسی شده و درخواست‌هایی که به‌صورت بلادرنگ هستند و دارای زمان پاسخ کمی می‌باشند و همچنین آن‌هایی که به پردازش، محاسبه و ذخیره‌سازی کمی نیاز داشته باشند، در همین لایه پردازش می‌شوند و درخواست‌های باقی‌مانده به لایه‌ی بالاتر که همان لایه‌ی ابر است، ارسال می‌شوند. لایه‌ی سوم یا لایه‌ی ابر (بالاترین لایه) قادر به پردازش و ذخیره‌سازی مقدار زیادی از اطلاعات می‌باشد. این لایه شامل سرورهای فیزیکی بسیاری می‌باشد. در چارچوب پیشنهادی مطابق شکل (۱)، وقتی‌که درخواست‌ها توسط دستگاه‌های اینترنت اشیاء به نقطه دسترسی[1] می‌رسند. در مرحله بعد وارد مؤلفه کنترل پذیرش می‌شوند. این مؤلفه تشخیص می‌دهد که درخواست در خود مه پردازش شود یا به ابر منتقل گردد. درصورتی‌که درخواستی باید در ابر پاسخ داده شود، مؤلفه کنترل پذیرش[2] این درخواست را به دروازه‌ی ابر[3] (CG) ارسال می‌کند. وقتی‌که درخواستی وارد ابر شد در آن قسمت نیز بسته به نوع درخواست و غیره تعیین می‌شود که وارد کدام سرور شود و بر این اساس در داخل صف‌های مخصوص آن سرور قرار می‌گیرد و در زمان مناسب به آن پاسخ داده می‌شود. اما چنانچه درخواستی می‌بایست در مه پاسخ داده شود (مانند درخواست‌های رسیده از سیستم‌های بلادرنگ)، به مولفه‌ی مدیر تامین پویای سرویس[4] (DSPM) فرستاده می‌شود. این مولفه که مولفه‌ی اصلی چارچوب پیشنهادی ما می‌باشد، باید تشخیص دهد که برای این درخواست رسیده چه منابعی لازم است و کدام گره یا گره‌های مه برای رسیدگی به این درخواست می‌بایست انتخاب شود، بطوریکه معیارهای کیفیت سرویس برای این درخواست رعایت شود. وقتی‌که درخواستی توسط مؤلفه DSPM در اختیار گره کنترل مه در یک دامنه مه قرار گرفت، در مرحله بعد مؤلفه گره کنترل مه این درخواست را در اختیار گره یا گره‌های مه مشخص شده در داخل دامنه مه قرار می‌دهد.



شکل (۱): چارچوب پیشنهادی رویکرد ارائه شده

[1] Access Point
[2] Admision Control
[3] Cloud Gateway
[4] Dynamic Service Provisioning Manager

جدول (۱): مقایسه مکانیزم‌های مختلف ارائه سرویس پویا

منبع	روش مورداستفاده	ابزار ارزیابی	معیارهای ارزیابی	استراتژی توسعه	مطالعه موردی
[۷]	Heuristic-based	شبیه‌سازی (Java)	نقض تاخیر، هزینه، تاخیر سرویس، تعداد سرویس‌های مه	متمرکز	Smart traffic application
[۸]	MAPE + GA	شبیه‌سازی (iFogSim+ IBM CPLEX optimizer)	میزان شبکه مورداستفاده، تاخیر، هزینه اجرا، صحت	غیرمتمرکز	Smart mobile application
[۹]	Artificial Neural Network	شبیه‌سازی (PureEdgeSim)	توان مصرفی، بهره‌وری cpu، بهره‌وری شبکه و متوسط زمان انتظار درخواست	متمرکز	Health IoT application
[۱۰]	Distributed Learning Automata	شبیه‌سازی (iFogSim)	میزان مصرف انرژی، تاخیر سرویس	غیرمتمرکز	IoT application
[۱۱]	Heuristic-based	شبیه‌سازی (MATLAB)	هزینه و انعطاف‌پذیری	متمرکز	Fixed sensors, drones and mobile robots
[۱۲]	Formal method	شبیه‌سازی (iFogSim)	هزینه و بهره‌وری استفاده از دستگاه‌های اینترنت اشیاء	متمرکز	Smart road application
[۱۳]	Genetic algorithm	شبیه‌سازی (FogNetSim++)	تخلف از مهلت زمانی، تلفات منابع و هزینه سرویس	غیرمتمرکز	IoT application
[۱۴]	Heuristic-based	نمونه اولیه (Tensorflow)	زمان اجرا، تاخیر شبکه، صحت، مصرف انرژی	متمرکز	Smart mobile application
[۱۵]	PSO-RL	شبیه‌سازی (MATLAB)	توان عملیاتی، بهره‌وری منابع و تأخیر انتها به انتها	متمرکز	MEC application
[۱۶]	PSO + Heuristic-based	شبیه‌سازی (IBM CPLEX optimizer)	نرخ موفقیت، زمان اجرا	متمرکز	Smart road application
[۱۷]	Greedy heuristic-based	نمونه اولیه (Calvin)	نرخ پذیرش، میزان استفاده از CPU، هزینه	متمرکز	IoT application
[۱۸]	Heuristic-based	شبیه‌سازی (iFogSim)	مصرف انرژی، هزینه، نرخ جایابی، تاخیر، میزان استفاده از مه، میزان استفاده از شبکه	غیرمتمرکز	ITS application
[۱۹]	hybrid fault-tolerance algorithm	شبیه‌سازی (iFogSim)	قابلیت اطمینان خدمات مه، کارایی انرژی و بهره‌وری منابع	غیرمتمرکز	IoT application
[۲۰]	Online learning	شبیه‌سازی (ONE simulator)	نرخ کارایی، هزینه، نرخ پشیمانی	متمرکز	Smart mobile application
[۲۱]	Heuristic-based	شبیه‌سازی (MATLAB)	سود کلی	متمرکز	IoT application
[۲۲]	NSGA-III	شبیه‌سازی (MATLAB)	تاخیر، تعادل بار	متمرکز	IoV application
[۲۳]	PSO	شبیه‌سازی (MATLAB)	تاخیر سرویس، حجم کار پردازش‌شده، زمان جایابی،نرخ کاهش تاخیر	غیرمتمرکز	IoT application
[۲۴]	Markov theory	شبیه‌سازی (MATLAB)	زمان اجرا، سود	متمرکز	IoV application
[۲۵]	Heuristic-based	شبیه‌سازی (MATLAB)	تعداد پرس و جوهای تصمیم، مصرف انرژی	متمرکز	IoT application
[۲۶]	FAHP	نمونه اولیه (Openstack)	زمان جایابی، هزینه جایابی	غیرمتمرکز	IoT application
روش پیشنهادی ما	LA	شبیه‌سازی (iFogSim)	تاخیر سرویس، هزینه، نقض تاخیر	متمرکز	IoT application

۳-۱- چهارچوب پیشنهادی

موبایل و ... قرار دارد. در لایه‌ی مه دامنه‌های مه قرار دارند. این لایه، بین دو لایه‌ی بالایی (لایه‌ی ابر) و لایه‌ی پایینی (لایه‌ی دستگاه‌های اینترنت اشیاء) قرار دارد. اطلاعات مختلف توسط حسگرهای لایه پایینی جمع‌آوری شده و به لایه‌ی میانی (لایه‌ی مه) فرستاده می‌شود. در لایه‌ی مه، درخواست‌های دریافت‌شده از لایه‌ی پایین‌تر به‌صورت هوشمند بررسی شده و درخواست‌هایی که به‌صورت بلادرنگ هستند و دارای زمان پاسخ کمی می‌باشند و همچنین آن‌هایی که به پردازش، محاسبه و ذخیره‌سازی کمی نیاز داشته باشند، در همین لایه پردازش می‌شوند و درخواست‌های باقی‌مانده به لایه‌ی بالاتر که همان لایه‌ی ابر است، ارسال می‌شوند. لایه‌ی سوم یا لایه‌ی ابر (بالاترین لایه) قادر به

پردازش و ذخیره‌سازی مقدار زیادی از اطلاعات می‌باشد. این لایه شامل سرورهای فیزیکی بسیاری می‌باشد. در چارچوب پیشنهادی مطابق شکل (۱)، وقتی‌که درخواست‌ها توسط دستگاه‌های اینترنت اشیاء به نقطه دسترسی[1] می‌رسند. در مرحله بعد وارد مؤلفه کنترل پذیرش می‌شوند. این مؤلفه تشخیص می‌دهد که درخواست در خود مه پردازش شود یا به ابر منتقل گردد. درصورتی‌که درخواستی باید در ابر پاسخ داده شود، مؤلفه کنترل پذیرش[2] این درخواست را به دروازه‌ی ابر[3] (CG) ارسال می‌کند. وقتی‌که درخواستی وارد ابر شد در آن قسمت نیز بسته به نوع درخواست و غیره تعیین می‌شود که وارد کدام سرور شود و بر این اساس در داخل صف‌های مخصوص آن سرور قرار می‌گیرد و در زمان مناسب به آن پاسخ داده می‌شود. اما چنانچه درخواستی می‌بایست در مه پاسخ داده شود (مانند درخواست‌های رسیده از سیستم‌های بلادرنگ)، به مولفه‌ی مدیر تامین پویای سرویس[4] (DSPM) فرستاده می‌شود. این مولفه که مولفه‌ی اصلی چارچوب پیشنهادی ما می‌باشد، باید تشخیص دهد که برای این درخواست رسیده چه منابعی لازم است و کدام گره یا گره‌های مه برای رسیدگی به این درخواست می‌بایست انتخاب شود، بطوریکه معیارهای کیفیت سرویس برای این درخواست رعایت شود. وقتی‌که درخواستی توسط مؤلفه DSPM در اختیار گره کنترل مه در یک دامنه مه قرار گرفت، در مرحله بعد مؤلفه گره کنترل مه این درخواست را در اختیار گره یا گره‌های مه مشخص شده در داخل دامنه مه قرار می‌دهد.

لایه‌ی مه را شامل N دامنه مه[5] متفاوت در نظر می‌گیریم که هر دامنه مه به مجموعه‌ای از دستگاه‌های اینترنت اشیاء سرویس‌دهی می‌کند. هر دامنه مه شامل مجموعه‌ای از چند گره مه[6] و یک گره کنترل دامنه[7] است. هر گره مه قابلیت پردازش درخواست‌های ارجاع شده به آن را دارد. گره DSPM، وظیفه تعیین استراتژی نحوه توزیع و استقرار درخواست‌های رسیده از سوی دستگاه‌های اینترنت اشیاء بین دامنه‌های مه را بر عهده دارد. در گره DSPM، اطلاعات مربوط به گره‌های کنترل هر دامنه مه در یک پایگاه دانش وجود دارد. در ادامه به تشریح مولفه‌های اصلی چارچوب پیشنهادی می‌پردازیم:

۳-۱-۱- لایه اشیا

لایه‌ی اول یا پایین‌ترین لایه در معماری پیشنهادی شامل دستگاه‌های اینترنت اشیاء می‌باشد. این دستگاه‌های هوشمند و نیمه‌هوشمند مانند انواع حسگرها، لپ‌تاپ‌ها، موبایل‌ها، لوازم خانگی، ساعت‌های هوشمند، دوربین‌ها و ... از طریق شبکه‌های ناهمگون به هم متصل‌اند. داده‌های موردنیاز برای پردازش یا ذخیره‌سازی توسط این دستگاه‌ها از محیط اطراف جمع‌آوری شده و از این لایه به لایه‌ی بالاتر (لایه‌ی مه) ارسال می‌شوند. این داده‌ها ابتدا به دروازه‌های مه رفته و سپس به مولفه کنترل پذیرش منتقل می‌شوند؛ که در قسمت‌های بعد روند کار این مؤلفه شرح داده خواهد شد.

۳-۱-۲- لایه‌ی مه

لایه‌ی دوم در معماری پیشنهادی، لایه‌ی مه است که از یک الگوی معماری محاسباتی توزیع‌شده پیروی می‌کند و شامل دستگاه‌های لبه شبکه در لایه‌های سلسله مراتبی مختلف با درجه‌های مختلفی از محاسبات و قابلیت ذخیره‌سازی، می‌باشد. رایانش مه این امکان را به کاربران می‌دهد تا به سرویس‌های موردنیاز خود، در نزدیکی مکانی که اطلاعات ایجادشده‌اند، دسترسی داشته باشند. به‌عبارت‌دیگر، رایانش مه اجازه محاسبات و خدمات را در لبه شبکه فراهم می‌کند. درنتیجه، بسیاری از پیشرفت‌ها شامل بهبود کیفیت سرویس (QoS) (ازلحاظ تأخیر کمتر و بازدهی بالاتر)، بهبود مقیاس‌پذیری، بهره‌وری منابع و احراز هویت و مجوزسنجی ساده‌تر دستگاه‌های اینترنت اشیاء را فراهم می‌کند. در اکثر موارد، رایانش مه می‌تواند، درخواست‌های دستگاه‌های اینترنت اشیاء را بدون ارسال به سرورهای ابر، در لبه شبکه، انجام دهد. درخواست‌های کاربران ابتدا از لایه‌ی اشیاء وارد این لایه شده و سپس برحسب نوع درخواست، تعیین می‌شود که آیا درخواست می‌بایست در لایه‌ی مه پاسخ داده شود یا به لایه‌ی بالاتر فرستاده شود. این لایه شامل N دامنه مه، مولفه کنترل پذیرش و مولفه DSPM می‌باشد؛ که در ادامه به تشریح هر یک می‌پردازیم:

دامنه مه: لایه‌ی مه از N دامنه مه تشکیل می‌شود. هر دامنه مه مجموعه‌ای از یک یا چند گره مه و یک گره کنترل دامنه مه است. هر گره مه دارای قابلیت پردازش و ذخیره‌سازی می‌باشد. از این رو می‌تواند درخواست‌های واردشده را پردازش کند و نتیجه را به گره کنترل دامنه مه برگرداند. همچنین هر گره مه، اطلاعاتی از آخرین وضعیت منابع خود و سرویس‌های در حال‌اجرا را به گره کنترل دامنه مه خود ارسال می‌کند. گره کنترل دامنه مه، درخواست سرویس رسیده از DSPM را دریافت کرده و آن را به گره یا گره‌های زیرمجموعه‌ی خود جهت اجرا، ارسال می‌کند. البته قبل از ورود به منبع داخل گره مه، در صف آن منبع قرار می‌گیرد و بعدازآنکه نوبتش فرا رسید وارد منبع شده و به آن پاسخ داده می‌شود. در راس هر دامنه مه، یک گره کنترل دامنه مه قرار دارد که وظیفه اصلی آن، مدیریت مجموعه‌ی گره‌های مه آن دامنه است. این مؤلفه خود نیز یک گره مه قوی است که وظیفه مدیریت منابع و همچنین توزیع منابع برای درخواست‌های رسیده از مولفه DSPM بین گره‌های مه تحت مدیریت خود را بر عهده دارد. همچنین گره کنترل دامنه اطلاعات لازم در مورد آخرین وضعیت گره‌های مه مستقر در دامنه را در پایگاه‌داده خود ذخیره کرده و در مواقع لزوم آن‌ها را در اختیار گره DSPM قرار می‌دهد. این مولفه درخواست‌هایی که از سوی DSPM ارسال شده است را به گره مه

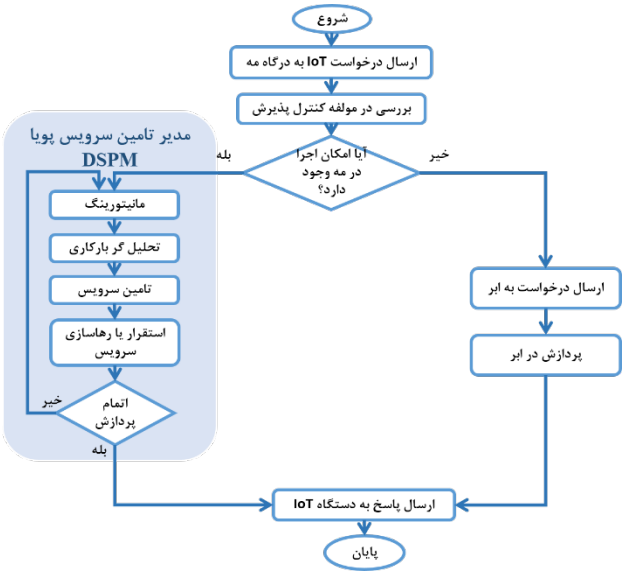
[1] Access Point
[2] Admision Control
[3] Cloud Gateway
[4] Dynamic Service Provisioning Manager
[5] Fog Domain
[6] Fog Node
[7] Control Node

مربوطه ارسال می‌کند. گره DSPM براساس اطلاعات دریافتی از گره کنترل دامنه و پایگاه دانش سرویس، مشخص می‌کند که کدام درخواست برای پردازش به کدام دامنه ارسال شود. گره کنترل دامنه مه، به این شکل عمل می‌کند که منابع مه موجود در دامنه تحت مدیریت خود را مانیتور می‌کند و اطلاعات این منابع (مثلاً تعداد منابع خاموش، تعداد منابع روشن، ظرفیت منابع و...) را به مؤلفه DSPM می‌دهد. DSPM تصمیم می‌گیرد که چه تعداد منابع برای هر درخواست در نظر گرفته شود و درنهایت نتیجه تصمیم‌گیری را به گره کنترل دامنه مه ارسال می‌کند و این مؤلفه بر اساس اطلاعاتی که از مؤلفه DSPM به دست آورده است تصمیم می‌گیرد که چه تعداد از منابع را خاموش و چه تعداد را روشن کند تا در این‌صورت منابع بتوانند به‌صورت کارآمد مورداستفاده قرار گیرند. درصورتی‌که منابع مه تحت کنترل گره کنترل دامنه مه، تماماً مشغول باشند یا از دسترس خارج‌شده باشند و یا از نظر ظرفیت، توانایی پاسخگویی به درخواستی را نداشته باشند، درخواست به گره‌های کنترل دامنه مه دیگری که در نزدیکی آن است منتقل می‌شود. هر گره مه شامل منابعی است که قابلیت اجرای سرویس‌های اینترنت اشیاء را دارند. این منابع شامل CPU، RAM و Storage می‌باشد که در هر گره مه ممکن است ظرفیت متفاوتی داشته باشند.

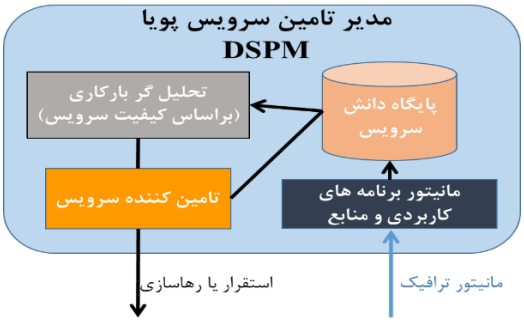
مولفه کنترل پذیرش: درخواست‌های دستگاه‌های اینترنت اشیاء در ابتدا از طریق درگاه مه[1] به مولفه کنترل پذیرش ارسال می‌شوند و این مؤلفه تصمیم می‌گیرد که هر درخواست بر اساس احتیاجاتش، به لایه‌ی مه فرستاد شود و یا مستقیماً به لایه‌ی ابر منتقل گردد. درصورتی‌که درخواستی بلادرنگ بوده و نسبت به تاخیر زمانی حساس باشد، باید وارد لایه‌ی مه شود، در نتیجه این مؤلفه آن درخواست را به مؤلفه‌ی DSPM در لایه‌ی مه می‌فرستد و درصورتی‌که باید به لایه‌ی ابر منتقل شود، این مؤلفه درخواست را به دروازه‌ی ابر ارسال می‌کند.

مولفه مدیر تامین سرویس پویا: از آن‌جا که ماهیت اجرای سرویس‌های برنامه‌های اینترنت اشیاء، پویا می‌باشد و ممکن است در حین اجرای یک سرویس در داخل یک گره مه، به یکباره سرویس موردنظر به منابع بیشتر یا کمتری نیاز داشته باشد، لذا ممکن است اجرای سرویس در گره مه متوقف شده و به گره‌ای دیگر محول شود، زیرا چنانچه سرویس موردنظر به منابع بیشتری نیاز داشته باشد، گره مه ممکن است دیگر قادر به پاسخ‌گویی نباشد و سرویس موردنظر نتواند در مهلت معین اجرا شود، درنتیجه سرویس، به یک گره با منابع بیشتر ارسال می‌شود. اما چنانچه سرویس موردنظر به منابع کمتری نیاز داشته باشد، ممکن است با دراختیار گرفتن منابع گره مه، مانع اجرای سایر سرویس‌های مه شود و این سرویس‌ها در مهلت معین اجرا نشوند. در چنین شرایطی سرویس موردنظر از گره مه قبلی خود آزاد می‌شود تا منابع آن گره مه رها شده و در اختیار سایر سرویس‌ها قرار گیرد و سرویس موردنظر نیز در یک گره مه با منابع کمتر مستقر می‌شود. این مولفه وظیفه دارد تا سرویس‌هایی را که مولفه کنترل پذیرش مشخص نموده که باید در لایه‌ی مه اجرا شوند، بصورت پویا تامین کند. برای این کار به اطلاعات مربوط به دامنه‌های مه و گره‌های موجود در آن‌ها نیاز دارد. مولفه DSPM خود یکی از گره‌های مه موجود در لایه‌ی مه است. اما از آنجا که کار اصلی تصمیم‌گیری در رابطه با تامین سرویس‌های موردنیاز به عهده این مولفه می‌باشد و برای این کار می‌بایست محاسباتی انجام شده و پایگاه دانشی از داده‌های محیط مه نگهداری شود، لذا گره‌ای می‌بایست برای این مولفه انتخاب شود که قابلیت‌های لازم را دارا باشد. در نتیجه یکی از گره‌های قوی موجود در لایه‌ی مه به عنوان مولفه DSPM انتخاب می‌شود.

شکل (۲) سناریوی چارچوب پیشنهادی را نشان می‌دهد. در این سناریو درخواست‌های برنامه‌های کاربردی اینترنت اشیاء از طریق درگاه مه به مولفه کنترل پذیرش ارسال می‌شود. این مولفه درخواست‌های رسیده را بررسی می‌کند. درخواست‌های بلادرنگ، پس از تعیین میزان منبع موردنیاز و گره یا گره‌های مقصد، برای سرویس‌دهی و اجرا در همان لایه‌ی مه به گره DSPM ارسال می‌شوند و سایر درخواست‌هایی که حساس به تاخیر نیستند، برای پردازش و اجرا به لایه‌ی ابر فرستاده می‌شوند. بنابراین وقتی درخواستی به مؤلفه‌ی کنترل پذیرش وارد می‌شود دو سناریو در پیش رو خواهد داشت که در ادامه به شرح این دو سناریو می‌پردازیم.



شکل (۲): سناریوی چارچوب پیشنهادی



شکل (۳): اجزاء مولفه DSPM

[1] Fog Gateway

سناریوی اول: این سناریو زمانی اتفاق می‌افتد که درخواست‌های رسیده به مؤلفه کنترل پذیرش دارای شرایطی هستند که باید در ابر پردازش شوند (مثلاً محدودیت زمانی برای پاسخ ندارند و یا نیازمند حافظه‌ای با حجم ذخیره‌سازی بالا می‌باشند و...) در این صورت درخواست مستقیماً وارد ابر می‌شود (مسیرسمت راست در شکل (۲)).

سناریوی دوم: این سناریو زمانی اتفاق می‌افتد که درخواست سرویس، از طریق مؤلفه کنترل پذیرش باید به مه وارد شود و در آنجا اجرا شود. مسیر سمت چپ در شکل (۲) این سناریو را نشان می‌دهد. در این مسیر مؤلفه DSPM قرار دارد که عنصر اصلی چارچوب پیشنهادی ما می‌باشد که در ادامه به‌صورت مفصل به آن پرداخته خواهد شد. شکل (۳) اجزاء مولفه DSPM را نشان می‌دهد. این مولفه از چندین واحد به شرح ذیل تشکیل شده است:

مولفه مانیتورینگ منبع و برنامه کاربردی[1]: این واحد اطلاعات مربوط به منابع مه و سرویس‌های درخواست‌شده توسط دستگاه‌های اینترنت اشیاء را جمع‌آوری می‌کند و از دو زیر مؤلفه مانیتور منبع و مانیتور سرویس‌های درخواست‌شده توسط کاربران تشکیل شده است. مؤلفه مانیتور منبع، به گره‌های کنترل دامنه در دامنه‌های مه مراجعه کرده و ازآنجا اطلاعات مربوط به منابع مختلف مه (مانند: تعداد منابع روشن، تعداد منابع خاموش و ظرفیت هر منبع و ...) را دریافت می‌کند. به عبارت دیگراین مولفه مسئول جمع‌آوری اطلاعات در مورد منابع مه می‌باشد. مؤلفه مانیتور سرویس‌های درخواستی کاربر، مسئول جمع‌آوری اطلاعات در مورد سرویس‌های درخواستی کاربران (دستگاه‌های اینترنت اشیاء) و احتیاجات آن‌ها می‌باشد. اطلاعات مانیتورشده توسط این دو زیرمؤلفه، جمع‌آوری و یکپارچه‌شده و سپس برای استفاده توسط سایر واحدهای DSPM در پایگاه‌دانش سرویس[2] ذخیره می‌شود.

پایگاه دانش سرویس: در این واحد، اطلاعات تمامی سرویس‌های درخواستی رسیده از طرف دستگاه‌های اینترنت اشیاء و نیز وضعیت منابع موجود در مه، توسط واحد مانیتورینگ ذخیره می‌شود. مدیریت دانش این پایگاه به عهده واحد تامین سرویس می‌باشد. پس از هر بار تخصیص یا بازپس‌گیری منبع و همچنین پس از هر استقرار یا رهاسازی سرویس در یک گره مه، اطلاعات این پایگاه‌دانش توسط واحد تامین سرویس به‌روزرسانی و مدیریت می‌شود.

تحلیل‌گر بارکاری[3]: این واحد با پردازش داده‌های جمع‌آوری‌شده از مؤلفه مانیتور در پایگاه دانش، اطلاعاتی در مورد وضعیت بهره‌وری فعلی سیستم و پیش‌بینی‌هایی از نیازهای آتی، به دست می‌آورد. چون درخواست‌های رسیده از دستگاه‌های اینترنت اشیاء پویا هستند و متناسب با زمان تغییر می‌کنند، در این قسمت بر اساس تاریخچه درخواست‌های اینترنت اشیاء، وضعیت آینده سیستم پیش‌بینی می‌شود. در این مرحله از تکنیک‌های سری‌های زمانی جهت پیش‌بینی وضعیت سیستم استفاده می‌شود. سپس نتایج به‌دست‌آمده در این واحد در داخل پایگاه‌دانش سرویس قرار می‌گیرد تا این اطلاعات برای واحدهای بعدی مورداستفاده قرار گیرد.

مولفه‌ی تامین سرویس[4]: پس از مشخص‌شدن وضعیت آینده سیستم توسط واحد تحلیل‌گر بارکاری، واحد تامین سرویس، بر اساس اطلاعات تحلیل‌شده در پایگاه‌دانش سرویس، برنامه‌ریزی می‌کند که کدام سرویس بر روی کدام گره مه مستقر، یا از روی کدام گره مه رهاسازی شود. واحد تامین سرویس، تصمیم‌گیرنده و هسته اصلی DSPM می‌باشد. در این قسمت با توجه به تحلیل انجام شده در مؤلفه تحلیل‌گر بارکاری و با استفاده از یک اتوماتای یادگیر تصمیم گرفته می‌شود که چه عملی انجام شود (سرویس بر روی گره مه مستقر شود یا از روی آن رهاسازی شود). درنهایت تصمیم گرفته‌شده در پایگاه‌دانش سرویس ذخیره می‌شود. سپس واحد تامین سرویس اطلاعات لازم برای اجرای این تصمیم را به گره کنترل دامنه مه ارسال کرده و گره کنترل دامنه مه هم بر اساس این اطلاعات تنظیمات لازم را روی گره یا گره‌های مه موردنظر در دامنه خود اعمال می‌کند.

۳-۱-۲- لایه‌ی ابر

بالاترین لایه در چارچوب پیشنهادی، لایه‌ی ابر است که شامل سرورهای قوی و مراکز داده با قابلیت ذخیره‌سازی بالا می‌باشد. درخواست‌های اینترنت اشیاء که حساس به تاخیر نیستند و همچنین درخواست‌هایی که به پردازش و ذخیره‌سازی بالایی نیاز دارند، پس از ورود به مولفه‌ی کنترل پذیرش، به این لایه ارسال می‌شوند (طبق سناریو اول که قبلا ذکر شد). این درخواست‌ها ابتدا مستقیماً از مولفه‌ی کنترل پذیرش به دروازه‌ی ابر منتقل می‌شوند و سپس وارد ابر می‌شوند. بسته به اینکه کدام سرور این نوع درخواست را اجرا می‌کند، درخواست به صف اختصاص داده شده آن سرور می‌رود و در نوبت خود به سرور مربوطه وارد می‌شود. در چارچوب پیشنهادی، مدیریت درخواست‌های ارسال‌شده به دروازه‌ی لایه‌ی ابر، تماما بر عهده لایه‌ی ابر بوده (از این جهت که درخواست به چه منبعی ارسال شود یا بر روی چه سروری مستقر شود) و نتایج به‌دست‌آمده، از طریق دروازه‌ی لایه‌ی ابر برای ارسال به دستگاه اینترنت اشیاء، به مولفه‌ی کنترل پذیرش در لایه‌ی دوم ارسال می‌شود.

۳-۲- الگوریتم پیشنهادی

در این بخش، رویکرد تامین سرویس پویای پیشنهادی را با جزییات شرح می‌دهیم. الگوریتم پیشنهادی ما منطبق بر مولفه‌ی DSPM است و در فاصله‌های زمانی مختلف (τ) تکرار می‌شود. در این الگوریتم با توجه به اجزای مولفه‌ی DSPM، به ترتیب عملیات مانیتورینگ، تحلیل حجم کاری و ارائه سرویس در بخش‌های Monitoring (خط ۳)، Workload_Analyzer (خط ۱۴) و Service_Provisioner (خط ۲۱) انجام می‌شود. در هر بار تکرار الگوریتم، درخواست‌های ورودی از

[1] Application and Resource Monitor
[2] Service Knowledge Base
[3] Workload Analyzer
[4] Service Provisioner

کاربران اینترنت اشیاء (خطوط ۳ تا ۷) و اطلاعات گره‌های مه (خطوط ۸ تا ۱۳)کنترل و بررسی می‌شود. در بخش مانیتورینگ درخواست‌های ورودی، پیام درخواستی برای هر سرویس اینترنت اشیاء، بررسی می‌شود و نیازهای سرویس با استفاده از تابع ()Extraction، از داخل آن استخراج می‌شود. ابتدا، مهلت اجرای سرویس اینترنت اشیاء، میزان پردازش موردنیاز، میزان حافظه‌ی اصلی موردنیاز، میزان فضای ذخیره‌سازی موردنیاز و میزان پهنای باند موردنیاز برای ارائه سرویس‌های اینترنت اشیاء جمع‌آوری می‌شوند. در نهایت، این الزامات سرویس در قالب یک رکورد اطلاعاتی (یعنی s_{info}) در پایگاه دانش سرویس ذخیره می‌شود. در بخش مانیتورینگ منابع گره‌های مه، با هر گره مه ارتباط برقرار می‌شود و گره مه موردنظر، وضعیت خود را در قالب یک پیام اعلام می‌کند. این پیام (یعنی f_{data_msg}) حاوی اطلاعاتی مانند ظرفیت گره مه است که می‌توان با استفاده از تابع ()Extraction این اطلاعات را استخراج کرد. ظرفیت CPU، ظرفیت حافظه اصلی و ظرفیت ذخیره‌سازی برای گره مه f از پیام استخراج شده و در نهایت، مشخصات گره در قالب یک رکورد اطلاعاتی (یعنی f_{data_info}) در پایگاه دانش سرویس ذخیره می‌شود.

در بخش تحلیل‌گر بار کاری (خطوط ۱۴ تا ۲۰) از داده‌های جمع‌آوری شده در بخش مانیتورینگ برای پیش‌بینی ترافیک ورودی به گره‌های لایه‌ی مه و وضعیت سیستم با توجه به سابقه درخواست سرویس‌های اینترنت اشیاء استفاده می‌شود. در این بخش، ما از تحلیل سری‌های زمانی برای تخمین نیازهای آینده گره‌های مه به سرویس‌های اینترنت اشیاء استفاده می‌کنیم. دنباله‌ای از اطلاعات و داده‌ها که در یک بازه‌ی زمانی مشخص و مربوط به یک موضوع جمع‌آوری شده‌اند، یک سری زمانی را می‌سازند. داده‌های تشکیل‌دهنده سری زمانی نمایانگر تغییرات پدیده‌ی مورد بررسی در طول زمان است؛ بنابراین می‌توانیم داده‌های سری زمانی را، داده‌هایی وابسته به زمان بدانیم. داده‌ها در یک سری زمانی اگر دارای فواصل مناسب باشند برای پیش‌بینی روند داده‌های آینده بسیار مفید است. از اهداف اصلی سری‌های زمانی می‌توان ابتدا به توضیح و تفسیر پدیده‌ها و در نهایت به پیش‌بینی پدیده‌ها اشاره کرد. با تحلیل یک سری زمانی ابتدا اطلاعاتی از روند یک پدیده می‌توان بدست آورد و اینکه متغیرها در هر شرایطی چه وضعیتی دارند و بعد از تفسیر یک پدیده می‌توان با توجه به شرایط موجود آینده پدیده را نیز پیش‌بینی کرد. تحلیل سری‌های زمانی به شناسایی هر الگوی منظمی در مشاهدات گذشته، با هدف پیش بینی برای دوره‌های آینده انجام می‌شود. تجزیه و تحلیل سری‌های زمانی کاربردهای بسیاری در حوزه‌های تجاری، اقتصادی، اجتماعی، زیست محیطی و صنعتی دارد. با تکیه بر اطلاعات حاصل از تحلیل داده های سری و وابسته به زمان می‌توانیم سیستمی پویاتر و بستری مناسب‌تر جهت تامین پویای سرویس در محیط مه فراهم کنیم. هدف استفاده از تحلیل سری زمانی در مدل‌سازی از روی داده‌های موجود برای پیش‌بینی مقادیر آن سری زمانی در آینده است.

تحلیل سری‌های زمانی به ایجاد بینشی در مورد رفتار آینده سری زمانی و آن حوزه مربوطه کمک می‌کند. این دنباله داده‌های معنادار کاربردهای فراوانی از جمله، بررسی نوسانات حجم بار کاری در گره‌های مه، تشخیص پیک‌های بار کاری، پیش‌بینی حجم بار کاری آینده و برنامه‌ریزی برای مشکلات احتمالی آینده می‌تواند داشته باشد.

تامین سرویس در سطح زیرساخت مه به صورت فوری و آنی صورت نمی‌گیرد و یک فاصله زمانی برای استقرار سرویس موردنظر برای استفاده، نیاز است. بنابراین، بایستی پیش‌بینی‌هایی برای تقاضاهای آتی برای تامین اتوماتیک سرویس به صورت پیشاپیش انجام شود. لذا ما در این بخش از الگوریتم پیشنهادی، جهت پیش‌بینی وضعیت بار کاری و تخمین نیازهای آینده سرویس‌های اینترنت اشیاء در گره‌های مه، از تحلیل سری زمانی و به ویژه از مدل ARIMA[1] استفاده می‌کنیم. طبق تجربیات گذشته روش ARIMA در اکثر مسائل پیش‌بینی بهتر جواب می‌دهد. علت آن هم اینست که ARIMA این قابلیت را دارد که در پیش‌بینی خود خطاهای پیش‌بینی‌های قبلی را هم در نظر بگیرد. ضمن اینکه استفاده از تحلیل سری زمانی و بویژه ARIMA سربار محاسباتی کمتری نسبت به روش‌های مشابه دارد، لذا برای استفاده در محیط‌های حساس به تاخیر نظیر محیط مه و سرویس‌های بلادرنگ مناسب است. در این بخش از الگوریتم (خطوط ۱۴ تا ۲۰)، ابتدا پارامترهای مدل ARIMA مقداردهی اولیه می‌شوند. پارامتر P مقدار خود واپس‌گرایانه برای قسمت فصلی، پارامتر D مقدار تفاوت قسمت فصلی، پارامتر Q مقدار میانگین متحرک برای قسمت فصلی، پارامتر m تعداد دوره‌ها در هر فصل،پارامتر p تعداد زمان‌های عقب‌مانده از مدل AR (مدل پیش‌بینی خودگرداننده)، پارامتر d درجه تفاوت، پارامتر q ترتیب مدل MA (مدل پیش‌بینی میانگین متحرک) و $prediction\_Size$ یک عدد صحیح است که نشان‌دهنده تعداد نقاط داده‌ای است که پس از مجموعه‌ی داده‌ها می‌بایست پیش‌بینی شوند. سپس، برای هر سرویس اینترنت اشیاء مستقر در هر گره مه، مدل ARIMA ترافیک ورودی آینده را از روی تاریخچه ترافیک با مقادیر موجود در آرایه آینده را از روی تاریخچه ترافیک با مقادیر موجود در آرایه $traffic\_data\_Array$ پیش‌بینی می‌کند. سرانجام، $Prediction\_Array$ که شامل مقادیر پیش‌بینی‌شده ترافیک ورودی برای سرویس‌های اینترنت اشیاء است به عنوان خروجی در بخش تحلیل‌گر بار کاری استفاده می‌شود. در بخش تامین‌کننده سرویس (خطوط ۲۱ تا ۳۷)، نتایج پیش‌بینی‌شده را از پایگاه‌دانش سرویس دریافت کرده و سپس برنامه‌ریزی می‌کند تا تصمیم مناسبی جهت استقرار یا آزادسازی سرویس اینترنت اشیاء گرفته شود. این مولفه، دامنه‌ی مه و گره (های) مه میزبان را برای استقرار یا آزادسازی سرویس اینترنت اشیاء مشخص می‌کند. برای این کار از یک روش مبتنی بر اتوماتای یادگیر جهت اتخاذ تصمیمات ارائه سرویس استفاده می‌شود.

[1] Autoregressive integrated moving average

اتوماتای یادگیر [۲۷,۲۸,۲۹]، یکی از روش‌های یادگیری تقویتی[1] و یک مدل تصادفی است که سعی دارد از طریق تعامل مکرر با محیط و بر حسب پاسخی که از محیط دریافت می‌کند، رفتار خود را بهینه‌سازی کند و احتمال مربوط به حالت[2]های خود را تغییر دهد و در هر حلت نیز عمل[3] (اقدام) خاص خود را اجرا نماید. اتوملتای یادگیر[4] (LA)یک تکنیک یادگیری در بسیاری از مسائل دنیای واقعی می‌باشد. از این تکنیک در مواقعی که اطلاعات کافی راجع به محیط وجود ندارد و همچنین در محیط‌های پویا، پیچیده و تصادفی با عدم قطعیت‌های زیاد مانند محیط‌های ابری، محیط‌های مه، شبکه‌های کامپیوتری و ... می‌توان استفاده کرد [۳۰,۳۱]. اتوماتای یادگیر از یک مجموعه‌ی محدود از حالات و اقدامات تشکیل شده و با دریافت بازخورد از محیط، سیاست‌های تصمیم‌گیری خود را به‌صورت تدریجی بهبود می‌بخشد. در هر اتوماتای یادگیر، تعداد محدودی از اعمال با احتمالات مختلف و مجموعه‌ای از ورودی‌ها وجود دارد. در هر مرحله، یک عمل بر اساس احتمالش به طور تصادفی از مجموعه‌ی متناهی اعمال انتخاب شده و در محیط اتوماتای یادگیر استفاده می‌شود. پس از استفاده از این عمل در محیط اتوماتای یادگیر، این عمل ارزیابی می‌شود و نتیجه‌ی آن به‌صورت سیگنال تقویتی به اتوماتای یادگیر ارسال می‌شود. جهت به‌روزرسانی وضعیت داخلی اتوماتای یادگیر و انتخاب عمل بعدی در مرحله بعد، از این سیگنال تقویتی ارسال‌شده توسط محیط استفاده می‌شود. این سیگنال می‌تواند سیگنال جریمه یا پاداش باشد و بسته به نوع و لندازه‌ی آن، احتمال هر عملی کاهش یا افزایش می‌یلبد. این روند تکرار می‌شود و به مرور زمان اتوماتای یادگیر کامل‌تر می‌شود و می‌تولند عمل بهتری را انتخاب کند تا محیط پاسخ‌های موردنظر را بدهد. اتوماتای یادگیر دارای سه جزء اصلی می‌باشد.

اتوماتا: واحد تصمیم‌گیرنده که شامل مجموعه‌ای از حالات داخلی و اقدامات ممکن است. این اتوماتا با استفاده از یک تابع احتمال، اقداماتی را انتخاب می‌کند.

محیط[5]**:** سیستمی خارجی که با اتوماتا تعامل دارد و بازخوردهایی (مانند جریمه یا پاداش) بر اساس اقدامات انجام‌شده ارائه می‌دهد. این محیط می‌تواند استاتیک، پویا یا تصادفی باشد.

الگوریتم یادگیری[6]**:** مکانیزمی که بازخورد دریافتی از محیط را تحلیل کرده و احتمال انتخاب هر اقدام را به‌روزرسانی می‌کند. این الگوریتم‌ها معمولاً بر پایه قواعد یادگیری عمل می‌کنند. از آنجا که در مساله تامین پویای سرویس مه نیز با یک محیط پویا و تصادفی در زیرساخت مه روبرو هستیم، از اتوماتای یادگیر در این بخش از الگوریتم خود استفاده نموده‌ایم. ممکن است دقت الگوریتم اتوماتای یادگیر به خوبی سایر روش‌های مرسوم نباشد اما از آنجا که اتوماتای یادگیر بار محاسباتی کمی دارد و به راحتی قابلیت این را دارد که به صورت نرم‌افزاری و سخت‌افزاری پیاده‌سازی شود، برای سرویس‌های حساس به تاخیر و محیط مه مناسب است. روش‌های دیگر نظیر شبکه عصبی و یادگیری عمیق بسیار زمان‌بر هستند لذا ما ترجیح دادیم که اجرای الگوریتم سریعتر انجام شود هر چند ممکن است دقت بالایی نداشته باشد. از طرف دیگر اتوماتای یادگیر در شرایطی که هیچ‌گونه اطلاعاتی از محیط در دسترس نیست به خوبی عمل می‌کند، که این در مورد محیط مه صدق می‌کند، در نتیجه ما در این بخش از الگوریتم خود از اتوماتای یادگیر جهت تصمیم‌گیری در خصوص استقرار یا آزادسازی سرویس‌ها استفاده می‌کنیم.

اعمال اتوماتای یادگیر در الگورتیم پیشنهادی، شامل تصمیمات استقرار و آزادسازی می‌باشد. در ابتدا، احتمال اولیه این اعمال تعیین می‌شود. در تنظیم اولیه پارامترها، با توجه به آزمایش‌های مکرر، ضریب پاداش $a=0.3$ و ضریب مجازات $b=0.7$ تنظیم می‌شود. علاوه بر این، آستانه‌ی درصد نقض[7] تاخیر $(VP_{threshold})$ یکی از ورودی‌های الگوریتم است و در الزامات کیفیت سرویس (QoS) هر سرویس اینترنت اشیاء مشخص می‌شود. در هر بازه زمانی τ، اتوماتای یادگیر ویژگی‌های محیط را از طریق تعامل و تکرار می‌آموزد و تصمیمات ارائه سرویس مناسب را برای اجرا در زیرساخت مه، مشخص می‌کند. در هر بازه زمانی، برای هر سرویس اینترنت اشیاء در هر گره مه، درصد نقض تأخیر (VP) برای انتخاب تصمیمات ارائه سرویس مناسب محاسبه می‌شود و با مقدار حد آستانه درصد نقض تاخیر $(VP_{threshold})$ مقایسه می شود. اگر مقدار VP از مقدار حد آستانه تجاوز کند، تصمیم Release، پاداش می‌گیرد و تصمیم Deploy، جریمه می‌شود. هنگامی که به تصمیم Release پاداش داده می‌شود، احتمال انتخاب تصمیم Release برای فاصله زمانی بعدی افزایش می‌یابد، در حالی که احتمال انتخاب تصمیم Deploy کاهش می‌یابد. در غیر این صورت، تصمیم Deploy پاداش دریافت می‌کند و تصمیم Release مجازات خواهد شد. اگر به تصمیم Deploy پاداش داده شود، احتمال تصمیم Deploy افزایش می یابد، در حالی که احتمال تصمیم Release کاهش می‌یابد. در نهایت، پس از ۳۰۰ بار تکرار (فاصله زمانی ۵ دقیقه برابر با ۳۰۰ ثانیه)، اتوماتای یادگیر، تصمیم بهینه (یعنی عملی با احتمال بیشتر) را برای فاصله زمانی بعدی مشخص می‌کند. در جدول (۲) الگوریتم پیشنهادی ما نمایش داده شده است.

۳-۳- پیچیدگی زمانی

پیچیدگی زمانی یک الگوریتم، به صورت تعداد مراحل موردنیاز برای حل یک نمونه مسئله با توجه به اندازه‌ی ورودی تعریف می‌شود. از آنجا که الگوریتم DSPM_Proc به عملیات اصلی انجام‌شده اشاره دارد، تعداد عملیات در این الگوریتم، مورد بررسی قرار می‌گیرد. عملیات نشان داده شده در این الگوریتم در هر بازه‌ی زمانی τ به صورت تکراری انجام می‌شود. در نتیجه، پیچیدگی زمانی الگوریتم را می‌توان با (۱) بیان کرد:

[1] Reinforcement Learning
[2] State
[3] Action
[4] Learning Automata
[5] Environment
[6] Learning Algorithm
[7] Violation Percent

$$T_{DSPM\_Proc} = T_{Monitoring} + T_{Workload\_Analyzer} + T_{Service\_Provisioner} \quad (۱)$$

که در آن $T_{Monitoring}$ پیچیدگی بخش مانیتورینگ (خطوط ۳ تا ۱۳ الگوریتم پیشنهادی)، $T_{Workload\_Analyzer}$ پیچیدگی بخش تحلیل‌گر بار کاری (خطوط ۱۴ تا ۲۰ الگوریتم پیشنهادی) و $T_{Service\_Provisioner}$ پیچیدگی بخش ارائه‌دهنده‌ی سرویس (خطوط ۲۱ تا ۳۷ الگوریتم پیشنهادی) می‌باشد. برای راحتی، فرض می‌کنیم که $|A|$ برابر با تعداد سرویس‌های اینترنت اشیا و $|F|$ برابر با تعداد گره‌های مه است. در بخش مانیتورینگ ابتدا برای هر سرویس اینترنت اشیاء، دستوراتی اجرا می‌شوند که احتیاجات آن سرویس را مشخص می‌کند و سپس برای هر گره مه، دستوراتی جهت تعیین ظرفیت‌های آن گره اجرا می‌شود. بنابراین پیچیدگی بخش مانیتورینگ طبق (۲) بدست می‌آید:

$$T_{Monitoring} = O(|A|) + O(|F|) \quad (۲)$$

بخش تحلیل‌گر بار کاری، حجم بار کاری آینده را با استفاده از تابع $ARIMA$ برای هر سرویس اینترنت اشیاء در هر گره مه پیش‌بینی می‌کند. بنابراین، پیچیدگی بخش تحلیل‌گر بار کاری طبق رابطه‌ی (۳) (توجه داشته باشید که T_{ARIMA} یک مقدار ثابت است، زیرا تابع $ARIMA$ در راهکار پیشنهادی ما همواره تعداد معینی از مقادیر را پیش‌بینی می‌کند) به دست می‌آید:

$$T_{Workload\_Analyzer} = O(|A| \times |F| \times T_{ARIMA}) = O(|A| \times |F|) \quad (۳)$$

پیچیدگی ارائه‌دهنده‌ی سرویس را می‌توان با (۴) بیان کرد. مقدار ثابت ۳۰۰ همان تعداد تکرار مراحل یادگیری در اتوماتای یادگیر است.

$$T_{Service\_Provisioner} = O(300 \times |A| \times |F|) = O(|A| \times |F|) \quad (۴)$$

با قراردادن مقادیر (۲-۴) در (۱) ، این رابطه بازسازی می‌شود:

$$T_{DSPM\_Proc} = O(|A|) + O(|F|) + O(|A| \times |F|) + O(|A| \times |F|) \quad (۵)$$

بنابراین، پیچیدگی کلی الگوریتم در هر بازه‌ی زمانی τ به شرح زیر است:

$$T_{DSPM\_Proc} \approx O(|A| \times |F|) \quad (۶)$$

۴- ارزیابی و عملکرد نتایج

در این بخش، به ارزیابی رویکرد پیشنهادی از طریق شبیه‌سازی با استفاده از ترافیک مصنوعی و واقعی می‌پردازیم. در ادامه توابع هدف و سپس تنظیمات محیط شبیه‌سازی را شرح داده و در نهایت، نتایج شبیه‌سازی مورد بررسی خواهد گرفت.

۴-۱- توابع هدف

در این بخش توابع هدف مورد ارزیابی در آزمایش‌ها را شرح می‌دهیم:

هزینه: اولین پارامتر ارزیابی ما هزینه می‌باشد. منظور از هزینه، هزینه‌ی سرویس‌دهی به یک درخواست دستگاه اینترنت اشیاء می‌باشد که شامل هزینه‌ی منابع مورداستفاده، هزینه‌ی ارتباطی و هزینه‌ی جریمه می‌شود. هزینه‌ی منابع مورداستفاده خود شامل هزینه‌ی منابع ابر، هزینه‌ی منابع مه، هزینه‌ی منابع مورداستفاده در مولفه‌ی کنترل پذیرش و هزینه‌ی منابع مورداستفاده در مولفه‌ی DSPM می‌باشد که هر یک از این موارد شامل هزینه‌ی CPU، حافظه‌ی اصلی و فضای ذخیره‌سازی موردنیاز برای اجرای سرویس در یک سرور ابری یا در یک گره مه به همراه هزینه‌ی پردازش، حافظه و فضای ذخیره‌سازی موردنیاز در مولفه‌ی کنترل پذیرش و مولفه‌ی DSPM می‌باشد. هزینه‌ی ارتباطی شامل هزینه‌ی ارتباط بین گره‌های موجود در لایه‌ی مه و سرورهای لایه‌ی ابر، هزینه‌ی ارتباط بین گره‌ها در لایه‌ی مه با همدیگر و هزینه‌ی ارتباط بین مولفه‌ی DSPM و گره‌های مه می‌باشد. هزینه‌ی جریمه برابر با هزینه‌ای است که مولفه‌ی DSPM می‌بایست درصورت وقوع نقض تاخیر متحمل پرداخت آن شود. در آزمایش‌های انجام‌شده ما مقدار متوسط پارامتر هزینه را محاسبه می‌کنیم.

تاخیر سرویس: مدت‌زمان بین ارسال یک درخواست سرویس توسط یک دستگاه اینترنت اشیاء و دریافت یک پاسخ برای آن تاخیر سرویس نامیده می‌شود. متوسط تاخیر سرویس برای یک سرویس، پارامتر دوم در ارزیابی‌های ما می‌باشد. این پارامتر شامل میانگین تاخیر انتشار بین دستگاه‌های اینترنت اشیاء و یک گره مه، میانگین تاخیر انتشار میان یک گره مه و یک سرور ابری، زمان انتظار سرویس درخواست‌شده در گره مه (شامل تاخیر مربوط به انتظار در صف انتظار گره مه و تاخیر پردازش سرویس در آن گره مه)، زمان انتظار سرویس درخواست شده در سرور ابری (شامل تاخیر مربوط به انتظار در صف انتظار سرور ابری و تاخیر پردازش سرویس در آن سرور ابر)، میانگین تاخیر انتقال یک درخواست از بین یک دستگاه اینترنت اشیاء به یک گره در لایه‌ی مه، میانگین تاخیر انتقال یک درخواست بین یک گره در لایه‌ی مه و یک سرور در لایه‌ی ابر می‌باشد.

نقض تاخیر: پارامتر سوم مورد ارزیابی در آزمایش‌های ما نقض تاخیر می‌باشد. نقض تاخیر نیز یکی از معیارهای مهم برای اندازه‌گیری کیفیت خدمات است که در واقع برابر با درصد درخواست‌هایی است که در مهلت مقرر به آن‌ها پاسخ داده نمی‌شود.

۴-۲- تنظیمات شبیه‌سازی

آزمایشات ما در محیط جاوا توسعه یافته و در آن‌ها از مجموعه ابزار iFogsim برای مدل‌سازی موجودیت‌های محیط مه [۳۲] (به عنوان مثال، سرورهای ابری، گره‌های مه و دستگاه‌های اینترنت اشیاء) استفاده شده است. علاوه براین، ما تجزیه و تحلیل تجربی خود را بر روی رایانه‌ای با پردازنده Intel Core i3، 4 گیگابایت RAM، 500 گیگابایت فضای ذخیره‌سازی بر روی دیسک و در محیط Windows 10-64bit انجام داده‌ایم. ما از ردیابی ترافیک واقعی به نام MAWI Working Group برای دستیابی به نتایج واقعی‌تر در سناریوهای مختلف استفاده کرده‌ایم [۳۳].

جدول (۲): الگوریتم پیشنهادی

A l g o r i t h m : D S P M _ P r o c

1:***Begin***
2:***for each*** (Time interval τ) ***do***
//Monitoring
3: ***for each*** (*IoT Service s*) ***do***
4: ***Receive*** req_{msg} ***for*** *s*
// The ***Extraction()*** function extracts *Deadline,Processing_required, Memory_required,Storage_reqired and Bandwidth_required* from inside the message
5: s_{data_info}= ***Extraction***(req_{msg}) // s_{data_info} is IoT service information record
6: ***Update Service_Database with*** s_{data_info}
7: ***end for each***
8: ***for each*** (*Fog node f*) ***do***
9: ***Connect to*** *f_node*
10: ***Receive*** f_{data_msg} ***from*** *f_node*
// The ***Extraction()*** function extracts *CPU_Capacity, Memory_Capacity and Storage_Capacity* from inside the message
11: f_{data_info}= ***Extraction***(f_{data_msg}) //f_{data_info} is information record for fog node *f*
12: ***Update Service Database with*** f_{data_info}
13: ***end for each***
//Workload_Analyzer
14: ***Initialize ARIMA parameters:*** *P=1, D=1, Q=1, m=0, p=3, d=0, q=3, prediction_Size=300;*
15: ***for each*** *(IoT Service s)* ***do***
16: ***for each*** *(Fog Node f)* ***do***
17: *traffic_data_array=read(traffic(s,f))* //incoming Traffic of service s to fog node *f* until now
18: *Prediction_Array=ARIMA(traffic_data_array, P, D, Q, m, p, d, q, prediction_Size)*
19: ***end for each***
20: ***end for each***
// Service_Provisioner
21: ***for each*** *(Fog Service s)* ***do***
22: ***for each*** *(Fog Node f)* ***do***
23: ***Initialize action probabilities:*** $P_{Deploy}(t=1)$ =1-$P_{Release}(t=1)$, $P_{Release}(t=1)$ =Initial-value()
24: ***Initialize parameter:*** a=0.3 ; b=0.7; $VP_{threshold}$;
25: Select one of the actions randomly (Deploy, Release)
26: ***while*** (t<300) ***do***
27: Calculate the *VP* in the time *t*
28: ***if*** $(VP(t)>VP_{threshold})$ ***then***
29: $P_{Release}(t+1) = P_{Release}(t)$+ a (1-$P_{Release}(t)$), $P_{Deploy}(t+1)$ = (1-a) $P_{Deploy}(t)$
30: ***else***
31: $P_{Release}(t+1)$ = (1-b) $P_{Release}(t)$, $P_{Deploy}(t+1)$ = (b/(r-1)) + (1-b) $P_{Deploy}(t)$
32: ***end if***
33: ***end while***
34: Selected action (α_i)=Select one action with maximum probability; //***Max***(P_{Deploy}, $P_{Release}$)
35: ***Return*** Selected action (α_i) as optimal service provisioning decision for fog service *s*
36: ***end for each***
37: ***end for each***
38:***end for each***
39:***End***

این آرشیو ترافیکی مربوط به ردیابی ترافیک در روزهای ۱۲ و ۱۳ آوریل ۲۰۲۳ می‌باشد. کارگروه MAWI این آرشیو از ردیابی‌های ترافیکی مربوط به نمونه‌های گرفته شده را در فواصل زمانی ۱۵ دقیقه‌ای، تهیه کرده است. در ترافیک واقعی MAWI، تعداد گره‌های مه محدود است. ما همچنین از یک مولد ترافیک[1] مصنوعی نیز استفاده می‌کنیم که یک بار کاری مصنوعی را براساس زنجیره مارکوف زمان-گسسته[2] (DTMC)، برای شبیه‌سازی دقیق‌تر ایجاد می‌کند. همچنین در ردیابی‌های ترافیکی خود، نوع بسته‌ها را بسته‌های TCP و UDP در نظر می‌گیریم زیرا پروتکل‌های رایج در اینترنت اشیاء، مانند mDNS، AMQP، MQTT و COAP، ازTCP وUDP برای انتقال پیام‌های خود استفاده می‌کنند [۳۴]. در این پژوهش، ما فرض می‌کنیم که جریمه نقض سرویس[3]، مطابق با یک عدد تصادفی بین ۱۰ تا ۲۰ در هر درصد در ثانیه برای سناریو ۱ و سناریو ۲ و بین ۱۰۰ تا ۲۰۰ در هر درصد در ثانیه برای سناریوی ۳ باشد. تأخیر انتشار بین دستگاه‌های اینترنت اشیاء و گره‌های مه بین ۱ تا ۲ میلی‌ثانیه و بین گره‌های مه و سرورهای ابری از ۱۵ تا ۳۵ میلی‌ثانیه در نظر گرفته می‌شود. رسانه انتقال بین دستگاه‌های اینترنت اشیاء و گره‌های مه، WiFi در نظر گرفته شده و پهنای باند این لینک ۱ گیگابیت بر ثانیه فرض می‌شود. پهنای باند لینک ارتباطی بین سرورهای ابری و گره‌های مه بین ۱۰ تا ۱۰۰ گیگابیت در ثانیه و بین گره‌های مه با یکدیگر بین ۵۴ مگابیت بر ثانیه و ۱ گیگابیت بر ثانیه در نظر گرفته می‌شود. همچنین فاصله بین سرورهای ابری و گره‌های مه بین ۶ تا ۱۰ هاپ و پهنای باند بین مولفه‌ی DSPM و گره‌های مه، ۱۰ گیگابیت بر ثانیه در نظر گرفته شده است. سطح کیفیت سرویس برای سرویس‌های مختلف به‌صورت یک عدد متغیر از ۹۰٪ تا ۹۹.۹۹۹٪ (که یک عدد تصادفی است) در نظر گرفته شده است. در شبیه‌سازی ما، ظرفیت پردازش هر گره مه بین ۸۰۰ تا ۱۳۰۰ میلیون دستورالعمل درثانیه[4] (MIPS) است و ظرفیت پردازش هر سرور ابری بین ۱۶۰۰۰ تا ۲۶۰۰۰ میلیون دستورالعمل درثانیه (MIPS) در نظر گرفته شده است. ظرفیت حافظه گره‌های مه و سرورهای ابری به ترتیب ۸ گیگابایت و ۳۲ گیگابایت و ظرفیت ذخیره‌سازی گره‌های مه ($Stor^f$) بیش از ۲۵ گیگابایت و ظرفیت ذخیره‌سازی سرورهای ابری ۱۰ برابر گره‌های مه در نظر گرفته شده است. هزینه ارتباط بین مولفه‌ی DSPM و گره‌های مه، ۰.۵ در هر گیگابایت و بین گره‌های مه با یکدیگر، ۰.۲ در هر گیگابایت است.

ما برنامه‌های واقعیت افزوده موبایل[5](MAR) را به‌عنوان یکی از رایج‌ترین برنامه‌های کاربردی در حال اجرا بر روی دستگاه‌های اینترنت اشیاء در نظر می‌گیریم تا کاربرد و مزایای چارچوب پیشنهادی خود را نشان دهیم و مقادیر واقعی برای پارامترهای مختلف را در شرایط عملی به دست آوریم. مقدار موردنیاز پردازش برای سرویس‌های موجود در برنامه‌های کاربردی MAR بین ۵۰ تا ۲۰۰ میلیون دستورالعمل در هر

[1] traffic generator
[2] Discrete-Time Markov Chain
[3] Penalty Price
[4] Million Instructions Per Second
[5] Mobile Augmented Reality

درخواست است [۳۵]. همچنین میزان فضای ذخیره‌سازی موردنیاز برای این برنامه‌ها بین ۵۰ تا ۵۰۰ مگابایت و مقدار حافظه موردنیاز بین ۲ تا ۴۰۰ مگابایت است. در برنامه‌های MAR، متوسط اندازه درخواست یک سرویس و متوسط اندازه پاسخ داده شده به یک سرویس به ترتیب بین ۱۰ تا ۲۶ کیلوبایت و ۱۰ تا ۲۰ بایت است و درخواست‌ها دارای آستانه‌ی تاخیر ۱۰ میلی‌ثانیه هستند. همچنین هزینه پردازش در سرورهای ابری و گره‌های مه، ۰.۰۰۲ در هر میلیون دستورالعمل و هزینه ذخیره‌سازی در سرورهای ابری و گره‌های مه، ۰.۰۰۴ در هر گیگابایت در ثانیه تعیین شده است. فرض بر این است که هر گره مه دارای ۴ واحد پردازش و هر سرور ابری دارای ۸ واحد پردازش است.

۴-۳- تحلیل‌نتایج

در این بخش، به بررسی و تحلیل نتایج حاصل از آزمایش‌های خود می‌پردازیم. ما مجموعه‌ای از آزمایش‌ها را تحت سناریوهای مختلف انجام می‌دهیم تا اثربخشی مکانیزم پیشنهادی خود را ارزیابی کنیم. ما مکانیزم پیشنهادی خود را با چهار روش اساسی دیگر به نام‌های Optimal، All-Cloud، Min-Viol [۱۲] و Fog-Static [۷] مقایسه می‌کنیم. اولین روش Optimal نامیده می‌شود و یک راه حل بهینه است که بهترین روش جایابی برای سرویس‌های اینترنت اشیاء در گره‌های مه را فراهم می‌کند. در این روش تمامی حالات ممکن برای جایابی سرویس‌های مختلف بر روی گره‌های مه بررسی شده و از بین حالات مختلف، بهترین حالت ممکن انتخاب می‌شود. روش دوم All-Cloud نام دارد. این الگوریتم تمام سرویس‌های اینترنت اشیاء را جهت اجرا، به سرورهای ابری ارسال می‌کند و به هیچ عنوان از گره‌های مه استفاده نمی‌کند. روش سوم که Min-Viol [۷] نامیده می‌شود، رویکردی حریصانه است که در ارائه سرویس پویا، نقض تأخیر سرویس‌های اینترنت اشیاء را در نظر می‌گیرد. روش چهارم Fog-Static [۷] نامیده می‌شود، که یک رویکرد ایستا می‌باشد که تمام سرویس‌های اینترنت اشیاء را در گره‌های مه مستقر می‌کند و هیچ سرویسی را به سرورهای ابری ارسال نمی‌کند. در نهایت، مکانیزم پیشنهادی ما از تکنیک اتوماتای یادگیر جهت تعیین تصمیمات ارائه سرویس برای استقرار یا آزادسازی برنامه های اینترنت اشیا در زیرساخت مه ناهمگن و پویا استفاده می‌کند. با توجه به جدول (۳) ما از سه سناریوی مختلف جهت ارزیابی مکانیزم پیشنهادی خود استفاده می‌کنیم. در سناریوی اول به ارزیابی مکانیزم پیشنهادی تحت جریان ترافیک واقعی پرداخته شده است. در سناریوری دوم، مکانیزم پیشنهادی با راه‌حل بهینه مورد ارزیابی قرار گرفته و سناریوی سوم به بررسی تاثیر اندازه حدآستانه تاخیر سرویس‌های اینترنت اشیاء پرداخته است.

- **سناریوی اول: ارزیابی مکانیزم پیشنهادی تحت ردیابی ترافیک واقعی**

در این سناریو، ما از مجموعه داده ردیابی ترافیک در دنیای واقعی، مربوط به ۴۸ ساعت (در روزهای ۱۲و ۱۳ آوریل ۲۰۲۳) استفاده کردیم. علاوه براین، تعداد سرورهای ابری، تعداد گره‌های لایه‌ی مه و تعداد سرویس‌های اینترنت اشیا به ترتیب ۳، ۱۰ و ۴۰ تنظیم شده است. همچنین فاصله ردیابی ترافیک و فاصله زمانی τ، برابر با ۹۰۰ ثانیه است. شکل (۴ الف) و (۴ ب) به ترتیب نرخ ترافیک ورودی به گره‌های مه و میزان تاخیر سرویس را نشان می‌دهند. طبق نتایج بدست آمده، از آنجایی که در روش All-Cloud از گره‌های مه استفاده نمی‌شود و همه سرویس‌ها بر روی سرورهای ابری دور از دستگاه‌های اینترنت اشیا مستقر می‌شوند، لذا این روش بیشترین تاخیر سرویس را دارد. در مقایسه با روش All-Cloud، روش Min-Viol به دلیل تمرکز بر کاهش نقض تاخیر، تاخیر سرویس بسیار کمتری دارد. اما روش پیشنهادی ما کمترین تأخیر سرویس را به دلیل تکنیک پیش‌بینی ترافیک مورداستفاده برای ارائه سرویس‌های اینترنت اشیاء به دست می‌آورد. میزان تاخیر در روش Fog-Static نیز بیشتر از روش‌های Min-Viol و روش پیشنهادی که به صورت پویا عمل می کنند، اما کمتر از روش All-Cloud است، زیرا در این روش جایابی سرویس‌ها بر روی گره‌های مه به صورت استاتیک انجام می‌شود. شکل (۴ ج) میانگین هزینه را در روش پیشنهادی و روش‌های پایه نشان می‌دهد. با توجه به این شکل، روش All-Cloud به دلیل تأخیر زیاد و فاصله زیاد بین سرورهای ابری و دستگاه‌های اینترنت اشیاء، بیشترین هزینه را دارد. همچنین روش پیشنهادی در اکثر موارد کمترین هزینه را نسبت به سایر روش‌ها دارد، زیرا این روش سعی می‌کند با پیش بینی ترافیک ورودی، میزان تخلف تاخیر سرویس را به حداقل برساند و در نتیجه هزینه‌ها را کاهش دهد.

جدول (۳): سناریوها

سناریو	هدف	ترافیک	تنظیمات توپولوژی
سناریوی اول	ارزیابی مکانیزم پیشنهادی تحت ردیابی ترافیک واقعی	(2023/04/12-13) برابر با یک ردیابی ترافیک ۴۸ ساعته	تعداد سرورهای ابری=۳ تعداد گره های مه=۱۰ تعداد سرویس ها=۴۰ فاصله زمانی تغییر ترافیک=۹۰۰ ثانیه فاصله زمانی(τ)= ۹۰۰ ثانیه
سناریوی دوم	ارزیابی مکانیزم پیشنهادی در مقایسه با راه حل بهینه	(12:00 PM to 2:00 PM of 2023/04/12) برابر با یک ردیابی ترافیک ۲ ساعته	تعداد سرورهای ابری=۱ تعداد گره های مه=۱۰ تعداد سرویس ها=۲ فاصله زمانی تغییر ترافیک=۶۰ ثانیه فاصله زمانی(τ)= ۱۲۰ ثانیه
سناریوی سوم	بررسی تاثیر اندازه حدآستانه تاخیر سرویس	(12:00 PM to 4:00 PM of 2023/04/12) برابر با یک ردیابی ترافیک ۴ ساعته	تعداد سرورهای ابری=۳ تعداد گره های مه=۱۰ تعداد سرویس ها=۲۰ فاصله زمانی تغییر ترافیک=۱۰ ثانیه فاصله زمانی(τ)= ۱۰ ثانیه

از آنجایی که روش Min-Viol یک روش جایابی پویا است و بر کاهش نقض تاخیر سرویس مه تمرکز دارد، هزینه آن بسیار نزدیک به روش پیشنهادی است. هزینه روش Fog-Static بیشتر از روش پیشنهادی است، زیرا به دلیل ماهیت پویای خدمات مه، جایابی استاتیک نمی‌تواند با تقاضای متغیر ترافیک مطابقت داشته باشد.شکل (۴ د) درصد نقض تاخیر در الگوریتم‌های مذکور را نشان می‌دهد. بر اساس نتایج تجربی، مشخص شد که مکانیسم پیشنهادی کمترین میزان نقض تاخیر را (در حدود ۳ درصد) دارد، زیرا روش پیشنهادی ما مبتنی بر کاهش نقض تاخیر است. این حتی نسبت به روش Min-Viol که بر کاهش نقض تاخیر تمرکز دارد نیز کمتر است. همچنین بیشترین میزان نقض تاخیر مربوط به روش All-Cloud بوده و میزان نقض تاخیر در روش Fog-Static چیزی بین روش All-Cloud و Min-Viol است. شکل (۴ ه) و (۴ و) نشان‌دهنده میانگین تعداد سرویس‌های مستقر در لایه‌های مه و ابر است. نتایج نشان می‌دهد که در روش All-Cloud، همه سرویس‌های اینترنت اشیاء بر روی سرورهای ابری مستقر شده‌اند. اما، تعداد سرویس‌های اینترنت اشیاء که در روش پیشنهادی بر روی گره‌های مه مستقر شده‌اند بیشتر از روش Min-Viol است. همچنین، با گذشت زمان، تعداد سرویس‌های اینترنت اشیاء مستقر در گره‌های مه در روش Fog-Static تغییر نمی‌کند. دلیل این امر این است که روش Fog-Static سرویس‌های اینترنت اشیاء را به صورت ایستا جایابی می‌کند.

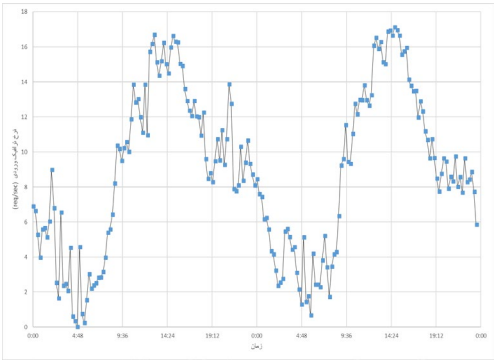
- **سناریوی دوم: ارزیابی مکانیزم پیشنهادی در مقایسه با راه‌حل بهینه**

در این سناریو، ما مکانیزم پیشنهادی خود را در مقایسه با راه‌حل‌های بهینه و روش Min-Viol مورد ارزیابی قرار می‌دهیم. در راه‌حل بهینه، ما همه حالت‌های ممکن جهت استقرار سرویس‌های اینترنت اشیاء در گره‌های لایه‌ی مه و سرورهای لایه‌ی ابر را بررسی می‌کنیم تا بهترین راه‌حل را با کمترین هزینه بدست آوریم. ما در این سناریو از مجموعه داده ردیابی ترافیک در دنیای واقعی مربوط به تاریخ ۱۲/۰۴/۲۰۲۳ به مدت ۲ ساعت (۱۲:۰۰ بعدازظهر تا ۲ بعدازظهر) استفاده کرده‌ایم. علاوه براین، تعداد سرورهای ابری، تعداد گره‌های لایه‌ی مه و تعداد سرویس‌های اینترنت اشیاء به ترتیب ۱، ۱۰ و ۴۰ تنظیم شده است. همچنین فاصله ردیابی ترافیک ۶۰ ثانیه و فاصله زمانی (τ)، برابر با ۱۲۰ ثانیه درنظر گرفته شده است. شکل (۵ الف) و (۵ ب) به ترتیب نرخ ترافیک ورودی به گره‌های مه و تاخیر سرویس را نشان می‌دهند. همان‌طور که مشاهده می‌شود، بیشترین و کمترین میزان تاخیر سرویس به ترتیب متعلق به روش‌های All-Cloud و Optimal است. میانگین تاخیر سرویس روش پیشنهادی نزدیک به روش بهینه و کمتر از روش Min-Viol است. به نظر می‌رسد که پیش‌بینی میزان ترافیک ورودی در روش پیشنهادی باعث جایابی مناسب سرویس‌های اینترنت اشیاء می‌شود، به طوری که میانگین تاخیر سرویس در این روش بسیار نزدیک به روش Optimal است. با توجه به نتایج به دست آمده در شکل (۵ ج)، روش All-Cloud بیشترین هزینه را دارد که عمدتاً به دلیل فاصله زیاد بین دستگاه‌های اینترنت اشیاء و سرورهای ابری است که باعث نقض تاخیر زیاد می‌شود. روش Optimal کمترین هزینه را دارد، زیرا بهترین راه‌حل را برای کاهش هزینه پیدا می‌کند، در حالی که روش پیشنهادی و Min-Viol بسیار نزدیک به روش Optimal هستند و روش پیشنهادی در اکثر موارد بهتر از Min-Viol عمل می‌کند. شکل (۵ د) درصد نقض تاخیر روش‌های پیشنهادی و پایه را نشان می‌دهد. همانطور که مشاهده می‌شود، بیشترین میزان نقض تاخیر متعلق به All-Cloud است، در حالی که روش پیشنهادی نسبت به روش Min-Viol عملکرد بهتری دارد و نزدیک‌ترین میزان به روش Optimal است. شکل (۵ ه) و (۵ و) میانگین تعداد سرویس‌های مستقر در لایه‌های مه و ابر را نشان می‌دهد. با توجه به نتایج، مشخص است که روش All-Cloud تمام سرویس‌های اینترنت اشیاء را بر روی سرورهای ابری مستقر می‌کند و روش Optimal آن‌ها را به طور کامل در گره‌های مه مستقر می‌کند. علاوه بر این، عملکرد روش پیشنهادی ما بهتر از Min-Viol و نزدیک به روش Optimal است. با توجه به اینکه در تنظیمات آزمایشی ما، جریمه تاخیر سرویس بالا در نظر گرفته شده است، بنابراین، روش پیشنهادی با استفاده از اطلاعات به دست آمده از پیش‌بینی ترافیک ورودی گره‌های مه، تمایل زیادی به استقرار خدمات بیشتری در گره‌های مه نسبت به سرورهای ابری دارد.

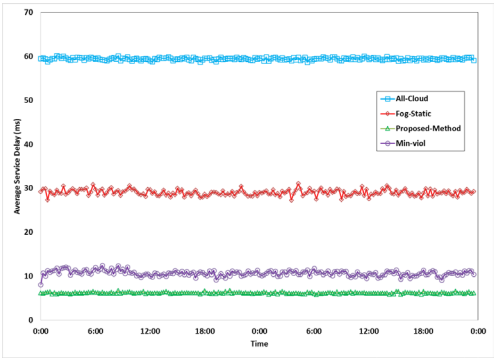
- **سناریوی سوم: بررسی تاثیر حدآستانه تاخیر سرویس**

این سناریو تاثیر اندازه حدآستانه تاخیر را در روش پیشنهادی ما مورد بحث قرار می‌دهد. ما از مجموعه داده ردیابی ترافیک در دنیای واقعی در تاریخ ۱۲/۰۴/۲۰۲۳ به مدت ۴ ساعت (۱۲:۰۰ بعدازظهر تا ۴ بعدازظهر) استفاده کرده‌ایم. علاوه براین، تعداد سرورهای ابری، تعداد گره‌های لایه‌ی مه و تعداد سرویس‌های اینترنت اشیاء به ترتیب برابر با ۳، ۱۰ و ۲۰ تنظیم شده است. شکل (۶ الف) تاخیر سرویس را در روش‌های پیشنهادی و پایه نشان می‌دهد. با توجه به نتایج به دست آمده، با افزایش اندازه حدآستانه‌ی تاخیر مربوط به سرویس، الگوریتم‌ها اغلب سرویس‌های اینترنت اشیاء را به سرورهای ابری منتقل می‌کنند. بنابراین سرویس‌ها به دلیل عدم سرویس‌دهی در لایه‌ی مه با تاخیر زیاد انجام می‌شود. در نتیجه میانگین تاخیر سرویس نیز افزایش می‌یابد و در نهایت به نتایج All-Cloud می‌رسد. همچنین، کمترین میانگین تاخیر سرویس مربوط به روش پیشنهادی است. از آنجایی که روش پیشنهادی ترافیک گره‌های مه را پیش‌بینی می‌کند، اگر امکان اجرای یک سرویس در مه وجود داشته باشد، آن را در لایه‌ی مه ارائه می‌کند. با توجه به نتایج به دست آمده در شکل (۶ ب) از نظر هزینه، هر چه حد آستانه‌ی تاخیر بزرگتر باشد، سرویس‌های بیشتری در لایه‌ی ابری مستقر شده و نقض سرویس کمتری رخ می‌دهد و در نتیجه هزینه کمتری به همراه دارد. همچنین در صورت وجود حد آستانه تاخیر کوچکتر، روش پیشنهادی هزینه قابل قبولی مشابه روش Min-Viol دارد. با این حال، در مورد آستانه تأخیر طولانی، حتی اگر هیچ سرویس اینترنت اشیاء در گره‌های مه مستقر نشده باشد، شامل هیچ تخلفی نمی‌شود. در نتیجه از نظر هزینه تمام روش‌های ذکرشده عملکرد یکسانی خواهند داشت. شکل (۶ ج) نقض تاخیر تمام

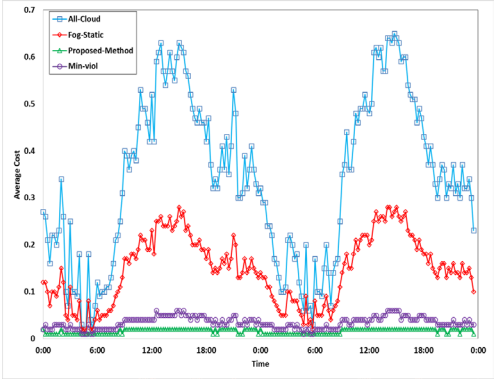
مکانیسم‌ها را نشان می‌دهد. نقض تاخیر روش پیشنهادی کمترین است، در حالی که رویکرد All Cloud بیشترین نقض تاخیر را دارد. در مقادیر کوچکتر حدآستانه‌ی تاخیر، کمترین نقض تاخیر مربوط به مکانیزم پیشنهادی است و در مقادیر بزرگتر حدآستانه تاخیر، همه روشها یکسان هستند. با افزایش حد آستانه تاخیر، نقض تاخیر در همه رویکردها کاهش می‌یابد، زیرا بیشتر سرویس‌ها به جای لایه‌ی مه به لایه‌ی ابری ارسال می‌شوند. شکل (۶ د) و (۶ ه) به ترتیب میانگین تعداد سرویس‌های مستقر در لایه‌های مه و ابر را نشان می‌دهد. با افزایش مقدار حد آستانه‌ی تاخیر سرویس، اکثر روش‌ها سرویس‌های اینترنت اشیاء را در سرورهای ابری مستقر می‌کنند و در نتیجه سرویس‌های کمتری بر روی گره‌های مه مستقر می‌شوند، در حالی که در حد آستانه‌های تاخیر کم، تعداد سرویس‌های مستقر در سرورهای ابری و گره‌های مه یک عدد ثابت است.



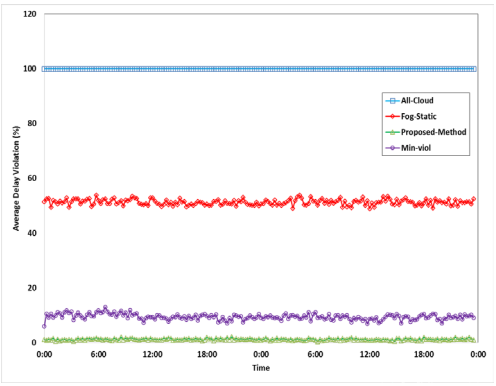
(الف) نرخ ترافیک ورودی



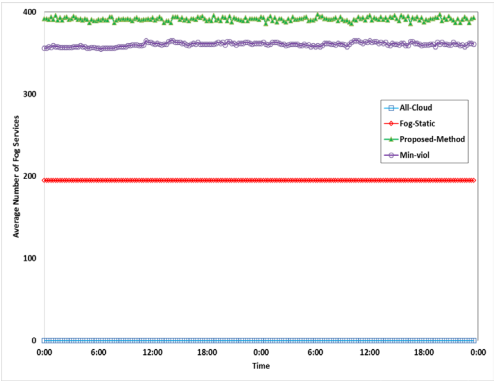
(ب) تاخیر سرویس



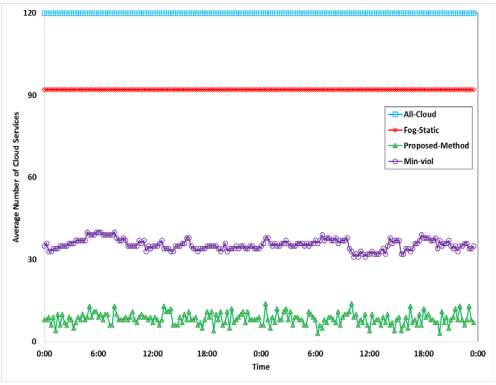
(ج) هزینه



(د) نقض تاخیر سرویس

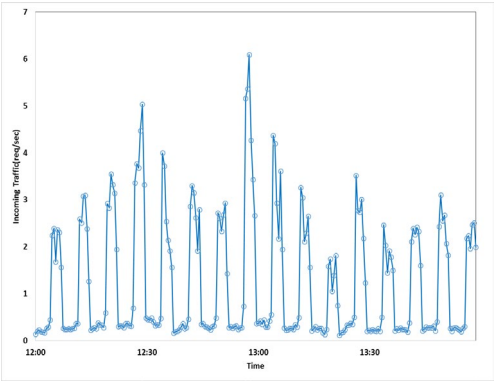


(ه) میانگین تعداد سرویس‌های مه

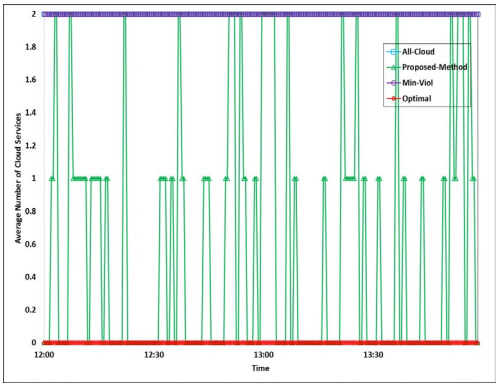


(و) میانگین تعداد سرویس‌های ابر

شکل (۴): نتایج شبیه سازی سناریوی اول

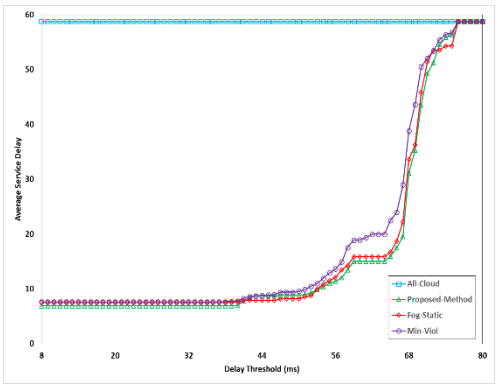


(الف) نرخ ترافیک ورودی

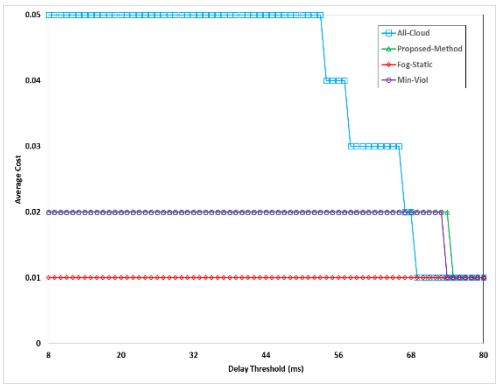


(و) میانگین تعداد سرویس‌های ابر

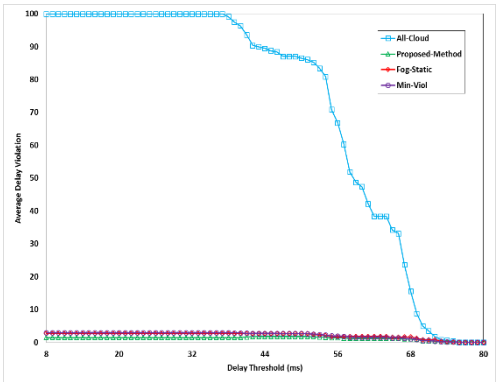
شکل (۵): نتایج شبیه سازی سناریوی دوم



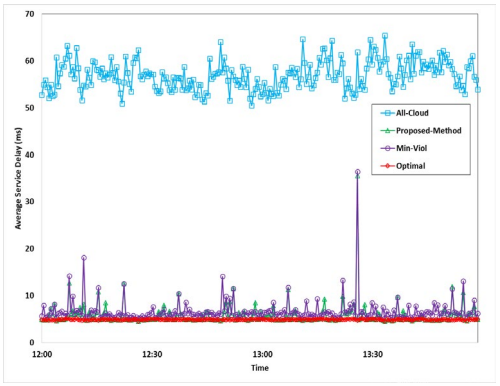
(الف) تاخیر سرویس



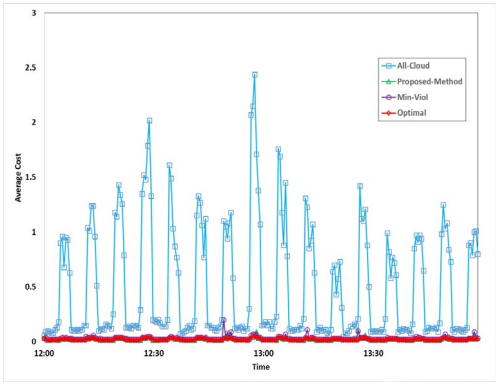
(ب) هزینه



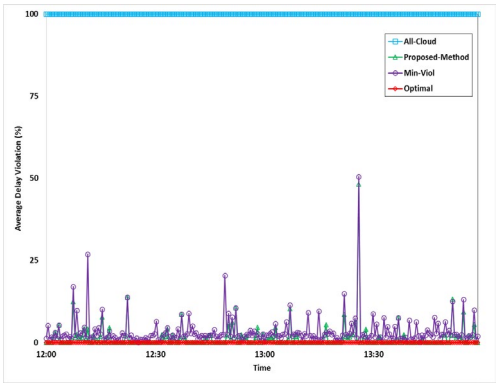
(ج) نقض تاخیر سرویس



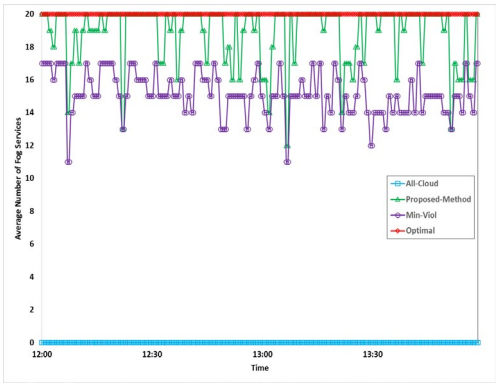
(ب) تاخیر سرویس



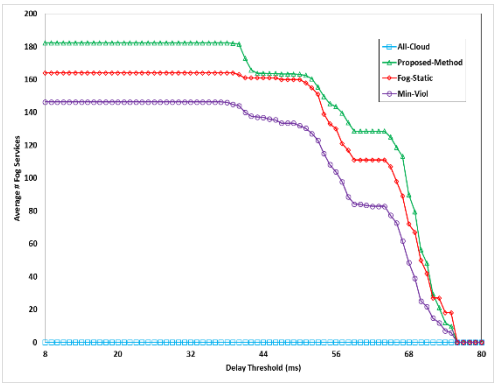
(ج) هزینه



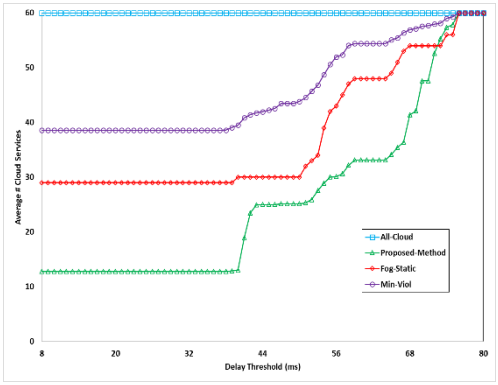
(د) نقض تاخیر سرویس



(ه) میانگین تعداد سرویس‌های مه



(د) میانگین تعداد سرویس‌های مه



(ه) میانگین تعداد سرویس‌های ابر

شکل (۶): نتایج شبیه سازی سناریوی سوم

۵- نتیجه‌گیری

با توجه به یافته‌های ارائه‌شـــده، رویکرد پیشـــنهادی که ترکیبی از روش‌های پیش‌بینی سـری‌های زمانی و اتوماتای یادگیر اسـت، توانسـته اسـت به‌طور مؤثری مسـائل مربوط به تامین پویای سـرویس در محیط مه را حل کند. این رویکرد نه تنها باعث بهینه‌سـازی اسـتفاده از منابع مه شـده اسـت، بلکه کیفیت سـرویس‌دهی به کاربران را نیز به‌طور قابل توجهی بهبود بخشـیده اسـت. با به‌کارگیری تکنیک‌های پیش‌بینی بار کاری و اتوماتای یادگیر، تخصیص منابع به‌صورت هدفمند انجام شده و از هدررفت منابع جلوگیری می‌شود. همچنین، نتایج شـبیه‌سـازی نشـان می‌دهد که این روش از نظر هزینه، کاهش تاخیر سـرویس، نقض تاخیر و اسـتقرار سـرویس‌ها در مقایسـه با روش‌های دیگر برتری دارد. علاوه بر این، اسـتفاده از اتوماتای یادگیر به دلیل سـادگی محاسـبات و سـرعت تصمیم‌گیری بالا، امکان تامین پویای سـرویس را به‌صورت بهینه فراهم می‌کند. این امر منجر به افزایش رضـایت کاربران، به‌ویژه در برنامه‌های کاربردی بلادرنگ، می‌شود. با اسـتقرار بیشـتر سـرویس‌ها در لایه‌ی مه و نزدیکی آن به دسـتگاه‌های اینترنت اشـیاء، زمان پاسـخ‌دهی به‌طور قابل توجهی کاهش یافته و نیازهای برنامه‌های کاربردی بلادرنگ به‌طور موثرتری برآورده می‌شـــود. در نهایت، این رویکرد پیشـــنهادی نه تنها کارایی سیسـتم را افزایش داده اسـت، بلکه سـطح کیفیت خدمات را نیز ارتقاء بخشیده است.

مرجع

[1] C. C. Byers, "Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled IoT networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 14-20, Aug. 2017, doi: 10.1109/MCOM.2017.1600885.

[2] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416-464, 1st Quart., 2018, doi: 10.1109/COMST.2017.2771153.

[3] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-oriented computing paradigms: A survey on architecture design and system management," *ACM Computing Surveys*, vol. 51, no. 2, Art. no. 39, Mar. 2019, doi: 10.1145/3154815.

[4] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657-1681, 3rd Quart., 2017, doi: 10.1109/COMST.2017.2705720.

[5] OpenFog Consortium Architecture Working Group, *OpenFog Reference Architecture for Fog Computing*, OPFRA001, Feb. 2017, p. 162.

[6] M. Iorga et al., "Fog computing conceptual model," Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, NIST Spec. Publ. 500-325, 2018, doi: 10.6028/NIST.SP.500-325.

[7] A. Yousefpour et al., "FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5080-5096, Jun. 2019, doi: 10.1109/JIOT.2019.2896311.

[8] J. P. Martin, A. Kandasamy, and K. Chandrasekaran, "Mobility aware autonomic approach for the migration of application modules in fog computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 5259-5278, Nov. 2020, doi: 10.1007/s12652-020-01854-x.

[9] H. S. Madhusudhan and P. Gupta, "Federated learning inspired Antlion based orchestration for Edge computing environment," *PLoS ONE*, vol. 19, no. 6, p. e0304067, Jun. 2024, doi: 10.1371/journal.pone.0304067.

[10] Y. Abofathi, B. Anari, and M. Masdari, "A learning automata based approach for module placement in fog computing environment," *Expert Systems with Applications*, vol. 237, p. 121607, 2024, doi: 10.1016/j.eswa.2023.121607.

[11] Y. Li et al., "AMPHI: Adaptive Mission-Aware Microservices Provisioning in Heterogeneous IoT Settings," in *2024 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2024, doi: 10.1109/SMARTCOMP61445.2024.00030.

[12] M. A. Hoque et al., "IoTaaS: Drone-based Internet of Things as a service framework for smart cities," in *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12425-12439, Jul. 2021, doi: 10.1109/JIOT.2021.3137362.

[13] M. Qin, M. Li, and R. O. Yahya, "Dynamic IoT service placement based on shared parallel architecture in fog-cloud computing," *Internet of Things*, vol. 23, p. 100856, 2023, doi: 10.1016/j.iot.2023.100856.

[14] S. Shekhar et al., "URMILA: Dynamically trading-off fog and edge resources for performance and mobility-aware IoT services," *Journal of Systems Architecture*, vol. 107, p. 101710, 2020, doi: 10.1016/j.sysarc.2020.101710.

[15] K. Abinaya, S. Dhanasekaran, and V. Vasudevan, "Optimizing VNF Service Provisioning in Mobile Edge Computing Networks Through PSO-RL," in *2024 Third International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, 2024, doi: 10.1109/INCOS59338.2024.10527497.

[16] A. Mseddi et al., "Joint container placement and task provisioning in dynamic fog computing," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10028-10040, Dec. 2019, doi: 10.1109/JIOT.2019.2935056.
[17] B. Donassolo et al., "Fog based framework for IoT service provisioning," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2019, doi: 10.1109/CCNC.2019.8651835.
[18] M.-Q. Tran et al., "Task placement on fog computing made efficient for IoT application provision," *Wireless Communications and Mobile Computing*, vol. 2019, Art. no. 6215454, 2019, doi: 10.1155/2019/6215454.
[19] J. Abawajy, S. Ghanavati, and D. Izadi, "Enhancing Dependability of Fog Computing Using Learning-based Task Scheduling," *IEEE Transactions on Dependable and Secure Computing*, early access, 2025, doi: 10.1109/TDSC.2025.3566395.
[20] T. Ouyang et al., "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Paris, France, 2019, pp. 1468-1476, doi: 10.1109/INFOCOM.2019.8737560.
[21] Qu, Yuben, et al. "Resilient service provisioning for edge computing." *IEEE Internet of Things Journal* (2021). https://doi.org/10.1109/JIOT.2021.3078620
[22] B. Shen et al., "Dynamic server placement in edge computing toward internet of vehicles," *Computer Communications*, vol. 178, pp. 114-123, 2021, doi: 10.1016/j.comcom.2021.07.021.
[23] H. Tran-Dang and D.-S. Kim, "FRATO: fog resource based adaptive task offloading for delay-minimizing IoT service provisioning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2491-2508, Oct. 2021, doi: 10.1109/TPDS.2021.3067654.
[24] C. Roy et al., "EdgeSafe: Dynamic Load Balancing Among Edge Nodes for Provisioning Safety-as-a-Service in Vehicular IoT Applications," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9320-9329, Sep. 2021, doi: 10.1109/TVT.2021.3097557.
[25] C. Roy et al., "DQ-Map: Dynamic Decision Query Mapping for Provisioning Safety-as-a-Service in IoT," *IEEE Internet of Things Journal*, early access, 2021, doi: 10.1109/JIOT.2021.3097535.
[26] E. G. Radhika and G. S. Sadasivam, "Budget optimized dynamic virtual machine provisioning in hybrid cloud using fuzzy analytic hierarchy process," *Expert Systems with Applications*, vol. 183, p. 115398, 2021, doi: 10.1016/j.eswa.2021.115398.
[27] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. North Chelmsford, MA, USA: Courier Corporation, 2012.
[28] K. Najim and A. S. Poznyak, *Learning Automata: Theory and Applications*. Oxford, U.K.: Elsevier, 2014.
[29] K.-S. Fu and T. J. Li, "Formulation of learning automata and automata games," *Information Sciences*, vol. 1, no. 3, pp. 237-256, 1969, doi: 10.1016/S0020-0255(69)80010-1.
[30] A. Rezvanian et al., *Recent Advances in Learning Automata* (vol. 754). Berlin, Germany: Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-72428-7
[31] M. Hasanzadeh-Mofrad and A. Rezvanian, "Learning automata clustering," *Journal of Computational Science*, vol. 24, pp. 379-388, 2018, doi: 10.1016/j.jocs.2017.09.008.
[32] R. Mahmud and R. Buyya, "Modelling and simulation of fog and edge computing environments using iFogSim toolkit," in *Fog and Edge Computing: Principles and Paradigms*, 1st ed., R. Buyya and R. N. Calheiros, Eds. Hoboken, NJ, USA: Wiley, 2019, pp. 1-35, doi: 10.1002/9781119525080.ch17.
[33] MAWI Working Group Traffic Archive, 2023. [Online]. Available: http://mawi.wide.ad.jp. Accessed: [Nov 2023].
[34] J. Tournier et al., "A survey of IoT protocols and their security issues through the lens of a generic IoT stack," *Internet of Things*, vol. 16, p. 100264, 2021, doi: 10.1016/j.iot.2020.100264.
[35] N. Didar and M. Brocanelli, "Joint AI task allocation and virtual object quality manipulation for improved MAR app performance," in *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*, 2024, pp. 1100-1110, doi: 10.1109/ICDCS60910.2024.00110.