

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

Meysam Tekiyehband¹, Mostafa Ghobaei-Arani^{2*}, Ali Shahidinejad³

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, Service Delay.

JCDSA, Vol. 3, No. 2, Summer 2025
 Received: 2025-05-09

Online ISSN: 2981-1295
 Accepted: 2025-07-08

Journal Homepage: <https://sanad.iau.ir/en/Journal/jcdsa>
 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: 10.82526/jcdsa.2025.1206281

COPYRIGHTS



©2025 by the authors. Published by the Islamic Azad University Shiraz Branch. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 International (CC BY 4.0)
<https://creativecommons.org/licenses/by/4.0>

* 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.





رویکردی کارآمد برای تامین پویای سرویس های اینترنت اشیا روی زیرساخت رایانش مه

میثم تکیه بند^۱، مصطفی قبائی آرانی^{۲*}، علی شهیدی نژاد^۳

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: پیشرفت های اخیر در فناوری اینترنت اشیا منجر به گسترش کاربردهای آن در حوزه های مختلفی مانند ساختمان ها، شهرها، مراقبت های بهداشتی و غیره شده است. رایانش مه به عنوان یک پلتفرم توزیع شده در لبه شبکه، امکان اجرای برنامه های کاربردی اینترنت اشیا را به صورت بلادرنگ فراهم می کند. به دلیل پویایی محیط مه و تغییرات مداوم در رفتار برنامه های اینترنت اشیا، یکی از چالش های اصلی، تامین بهینه و پویای سرویس های اینترنت اشیا بر روی منابع موجود در محیط مه است. این مطالعه با هدف ارائه یک مکانیزم تامین سرویس بهینه و پویا، از تکنیک های یادگیری تقویتی مانند اتوماتای یادگیر استفاده کرده است. چارچوب پیشنهادی این مطالعه شامل یک معماری سه لایه (اشیا، مه و ابر) و یک مولفه مدیر تامین سرویس پویا است که از سه زیرمولفه مانیتورینگ درخواست های سرویس و وضعیت گره های مه، تحلیل گر بار کاری و ارائه دهنده سرویس، تشکیل شده و وظیفه تامین سرویس های مورد نیاز در لایه مه را بر عهده دارد. این مکانیزم در سه سناریوی مختلف ارزیابی شده است که شامل بررسی جریان ترافیک واقعی، مقایسه با راه حل های بهینه و تاثیر آستانه تاخیر سرویس می شود. نتایج شبیه سازی نشان می دهد که روش پیشنهادی در مقایسه با سایر مکانیزم ها، تاخیر سرویس، هزینه و نقض تاخیر را به طور موثری کاهش می دهد.

واژه های کلیدی: رایانش مه، تامین پویای سرویس، اتوماتای یادگیر، برنامه های کاربردی اینترنت اشیا، تاخیر سرویس.

DOI: 10.82526/jcdsa.2025.1206281

نوع مقاله: پژوهشی

تاریخ چاپ مقاله: ۱۴۰۴/۰۶/۳۱

تاریخ پذیرش مقاله: ۱۴۰۴/۰۴/۱۷

تاریخ ارسال مقاله: ۱۴۰۴/۰۲/۱۹

۱- مقدمه

استفاده از پهنای بلند و هزینه ها را کاهش داده و آگاهی از موقعیت مکانی را فراهم کند و برای برنامه های کاربردی حساس به تاخیر به خصوص در مناطق نوظهور مانند اینترنت اشیا، اینترنت انرژی^۳، شهر هوشمند، پزشکی هوشمند، صنعت نسل ۴^۴ و جریان کلان داده ها^۵، کیفیت سرویس^۶ (QoS) را تقویت کند [۶، ۵]. یک برنامه کاربردی اینترنت اشیا ممکن است از چندین سرویس تشکیل شده باشد که می توانند در مکان های مختلفی اجرا شوند. به عنوان مثال یک برنامه کاربردی ممکن است سرویس هایی مثل تایید هویت، فایروال، حافظه نهان و رمزنگاری داشته باشد. برخی از سرویس های برنامه کاربردی، حساس به تاخیر هستند و آستانه تحمل تاخیر سختی دارند و ممکن است نیاز باشد تا نزدیک تر به کاربران یا منابع داده (به عنوان مثال سرویس حافظه نهان) اجرا شوند (برای مثال در لبه شبکه تا در دستگاه های رایانش مه). از طرف دیگر، برخی سرویس های برنامه های

بسیاری از برنامه های کاربردی اینترنت اشیا مانند واقعیت افزوده، اتومبیل های متصل و خودران، هواپیماهای بدون سرنشین، رباتیک صنعتی، نظارت و تولید در زمان واقعی به الزامات تاخیر سختگیرانه ای نیاز دارند. در بعضی موارد این الزامات به زیر ۱۰ میلی ثانیه می رسد [۱]. این برنامه های کاربردی نمی توانند تاخیر زیاد و غیرقابل پیش بینی ابر را در هنگام استقرار منابع ابری دور از محل تولید داده های برنامه کاربردی تحمل کنند. اخیراً رایانش مه [۲]، رایانش لبه [۳] و رایانش لبه موبایل^۲ [۴] پیشنهاد شده اند تا با قراردادن نزدیک تر منابع محاسبه، ذخیره سازی و شبکه به کاربران، تاخیر را کم کرده و پهنای بلند شبکه های اینترنت اشیا را کاهش دهند. مه می تواند تاخیر، میزان

* نویسنده مسئول

² Mobile Edge Computing (MEC)

³ Internet Of Energy

⁴ Industry 4.0

⁵ BigData Stream

⁶ Quality of Service



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 قرار گرفته و با روش‌های مرجع مقایسه می‌شود. در نهایت، در بخش آخر نتیجه‌گیری مطرح می‌گردد.



توزیع شده هستند، مساله استقرار سرویس‌های برنامه‌های کاربردی اینترنت اشیاء یک مساله چالش برانگیز بوده و ویژگی‌های مختلف دستگاه‌های اینترنت اشیاء و انتظارات مختلف کاربران نیز چالش‌های این مساله را بیشتر می‌کند. اجرای ایستای مساله تامین سرویس‌های 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 و ابر پویا می‌باشد و در طول زمان متغیر است لذا تامین سرویس‌های 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 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 تاخیر 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-Analyze-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 بر این، آنها یک چارچوب آگاه از تحرک الهام‌گرفته از مدل محاسبه خودکار برای مهاجرت ظروف به منابع مه تولید کرده‌اند. مدهوسودهان و همکاران

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 اجرا شدن 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 مشخص شده 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 فراهم کردن دسترس‌پذیری بالا با هزینه استقرار کم، در لایه‌ی ابر مستقر شوند یا اینکه برای فراهم کردن 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 مشخص شده توسط کاربران یا برنامه‌های کاربردی می‌باشد. این چالش به مساله تامین سرویس مه معروف است و هدف آن پیدا کردن بهترین نگاشت ممکن بین سرویس‌های اینترنت اشیاء و گره‌های مه با توجه به الزامات مشخص شده 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 ماهیتاً سلسله مراتبی بوده و منابع آن ناهمگن و

در [۹]، یک الگوریتم الهام گرفته از شبکه‌های عصبی مصنوعی^۱ مبتنی بر الگوریتم مورچه‌شب‌پره^۲ 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 توزیع شده پیشنهاد می‌کنند که با استفاده از ظرفیت حداکثری مه‌ها در یک شبکه‌ی مه‌ناهمگن، فضای جستجو را کاهش می‌دهد. در این روش، توپولوژی مه به یک اتوماتای یادگیر توزیع شده 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 اشیا، دو تابع هزینه تک‌هدفی برای انرژی و تأخیر، و یک تابع تک‌هدفی دیگر با در نظر گرفتن 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، پهبادهای 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 پهباد را پیشنهاد می‌دهند که امکان استقرار پویای گره‌های IoT و دروازه‌های وب را از طریق پهبادهای فراهم می‌کند. این راه‌حل با توزیع هوشمند دستگاه‌ها براساس نیاز کاربر و ارثه‌ی سرویس ابری توزیع شده، هزینه‌های راه‌اندازی را کاهش می‌دهد. تحلیل اقتصادی و پیاده‌سازی عملی آن‌ها در دو سناریوی کشاورزی هوشمند و پایش آلودگی هوا نشان می‌دهد که IoTaaS نه تنها صرفه‌جویی هزینه‌ای دارد، بلکه بهره‌وری استفاده از

دستگاه‌های اینترنت اشیا را افزایش می‌دهد. کوبن 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 در نظر گرفتن مهلت زمانی سرویس‌های اینترنت اشیا، 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 اشیا برای به حداقل رساندن نقض SLO در حالی که الزامات تأخیر آن‌ها را برآورده می‌کند، انتخاب می‌کند. همچنین، آن‌ها اثربخشی راه‌حل خود را با استفاده از دو برنامه اینترنت اشیا تلفن همراه 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. اینبایا 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 ارائه شده آن‌ها از مراحل متوالی بهینه‌سازی سودمندی با الگوریتم کرم‌های شب‌تاب، بهینه‌سازی بلادرنگ توان عملیاتی با تکنیک‌های کاهش گرادیان تصادفی و تبرید شبیه‌سازی شده، 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 فراابتکاری 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 اشیا مبتنی

² Antlion Algorithm

¹ Artificial Neural Network



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 حداقل رساندن هزینه استقرار و در عین حال برآوردن سهم بار بین منابع مه استفاده می‌کند. علاوه بر این، 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 به حداقل رساندن 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 بر این، 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]، اوینگ 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 بر این، آنها مجموعه‌ای از آزمایشات گسترده را برای نشان دادن استحکام راه‌حل پیشنهادی خود از نظر هزینه مهاجرت طراحی کرده‌اند و آن را با استراتژی‌های آفلاین و آنلاین مقایسه کرده‌اند.

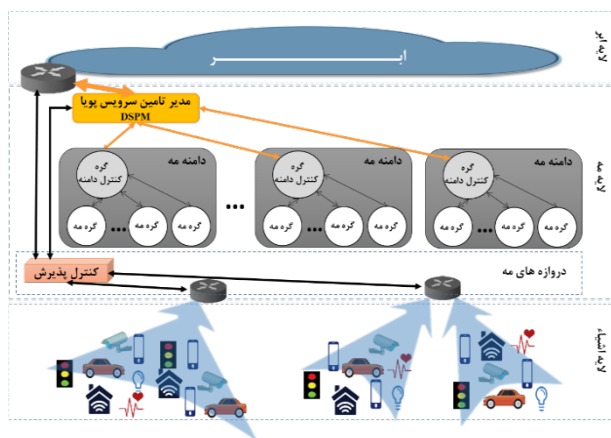
کیو و همکارانش در [۲۱]، ارائه سرویس انعطاف‌پذیر برای خرابی‌های نامشخص سرویس در محیط رایانش لبه را مورد بحث قرار داده‌اند. در واقع، مساله آن‌ها مکانیزم جایابی سرویس را برای حداکثرسازی مطلوبیت موردانتظار کلی با استقرار سرویس‌های اینترنت اشیاء در گره‌های لبه تعیین می‌کند. آن‌ها مساله خود را در قالب مساله بهینه‌سازی حداقل-حداکثر^۱ 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^۲ برای دستیابی به عملکرد بهتر استفاده کرده‌اند. علاوه بر این، 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 تأخیر در مقایسه با روش‌های موجود جایابی سرویس بهتر عمل می‌کند. دانگ و همکارانش [۲۳]، چارچوبی را برای به حداقل رساندن 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 بر این، 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 بر این، 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 بر این، 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

^۱ min-max

^۲ non-dominated sorting genetic algorithm 3

حسگرهای لایه پایینی جمع‌آوری شده و به لایه میانی (لایه می) فرستاده می‌شود. در لایه می، درخواست‌های دریافت‌شده از لایه می پایین‌تر به صورت هوشمند بررسی شده و درخواست‌هایی که به صورت 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 لایه بالاتر که همان لایه ابر است، ارسال می‌شوند. لایه سوم یا لایه 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)، وقتی که درخواست‌ها توسط دستگاه‌های اینترنت اشیا به نقطه دسترسی^۱ 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. در صورتی که درخواستی باید در ابر پاسخ داده شود، مؤلفه کنترل پذیرش^۲ این درخواست را به دروازه ابر^۳ (CG) ارسال می‌کند. وقتی که 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 مؤلفه مدیر تامین پویای سرویس^۴ (DSPM) فرستاده می‌شود. این مؤلفه که مؤلفه 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. وقتی که 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

ارائه سرویس مه برای به حداقل رساندن هزینه کل، جابجایی سرویس‌های مه به صورت ایستا انجام می‌شود. اما از آنجا که الگوی استفاده از منابع در برخی از برنامه‌های اینترنت اشیا 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 مورد نیاز است. بنابراین، در این پژوهش، ما از تکنیک‌های پیش‌بینی (به عنوان مثال سری‌های زمانی) برای پیش‌بینی بار کاری اینترنت اشیا با تئوری اتوماتای یادگیر برای استقرار سرویس‌های اینترنت اشیا در گره‌های مه یا سرورهای ابری برای به حداقل رساندن هزینه کل و تأخیر و برآوردن الزامات کیفیت سرویس سرویس‌های اینترنت اشیا استفاده کرده‌ایم. با توجه به بررسی و خلاصه مکانیزم‌های پویا برای جابجایی سرویس، مقایسه آنها از نظر ابزار ارزیابی، تکنیک‌های مورد استفاده، 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 مساله‌ای 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 پایه اتوماتای یادگیر، برای حل این مساله معرفی و در نهایت شبیه‌سازی آن انجام شده و به تجزیه و تحلیل نتایج حاصل از شبیه‌سازی پرداخته شده است. برای ایجاد سناریوها و تجزیه و تحلیل 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 اشیا، مه و ابر در نظر می‌گیریم. در پایین‌ترین لایه لایه اینترنت اشیا نام دارد تمامی حسگرها و اشیا هوشمند نظیر تبلت، موبایل و ... قرار دارد. در لایه می دامنه‌های مه قرار دارند. این لایه، بین دو لایه بالایی (لایه ابر) و لایه پایینی (لایه می) دستگاه‌های اینترنت اشیا قرار دارد. اطلاعات مختلف توسط

³ Cloud Gateway

⁴ Dynamic Service Provisioning Manager

¹ Access Point

² Admission Control



جدول (۱): مقایسه مکانیزم‌های مختلف ارائه سرویس پویا

منبع	روش مورد استفاده	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 مورد استفاده، 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 انعطاف پذیری	متمرکز	Fixed sensors, drones and mobile robots
[۱۲]	Formal method	شبیه‌سازی (iFogSim)	هزینه و بهره‌وری استفاده از دستگاه‌های اینترنت اشیا	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 پیشیمانی	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 پردازش شده، 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 اشیا) 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



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 توزیع شده 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 ایجاد شده‌اند، دسترسی داشته باشند. به عبارت دیگر، رایانش مه اجازه محاسبات و خدمات را در لبه شبکه فراهم می‌کند. در نتیجه، بسیاری از پیشرفت‌ها شامل بهبود کیفیت سرویس (QoS) (از لحاظ تأخیر کمتر و بازدهی بالاتر)، بهبود مقیاس‌پذیری، بهره‌وری منابع و احراز هویت و مجوزسنجی ساده‌تر دستگاه‌های اینترنت اشیا را فراهم می‌کند. در اکثر موارد، رایانش مه می‌تواند، درخواست‌های دستگاه‌های اینترنت اشیا را بدون ارسال به سرورهای ابر، در لبه شبکه، انجام دهد. درخواست‌های کاربران ابتدا از لایه‌ی اشیا 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، مؤلفه کنترل پذیرش و مؤلفه 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 وارد شده 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 حال اجرا 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 بعد از آنکه نوبتش فرا رسیده 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 مؤلفه 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 پایگاه داده خود ذخیره کرده و در مواقع لزوم آن‌ها را در اختیار گره DSPM قرار می‌دهد. این مؤلفه درخواست‌هایی که از سوی DSPM ارسال شده است را به گره مه

پردازش و ذخیره‌سازی مقدار زیادی از اطلاعات می‌باشد. این لایه شامل سرورهای فیزیکی بسیاری می‌باشد. در چارچوب پیشنهادی مطابق شکل (۱)، وقتی که درخواست‌ها توسط دستگاه‌های اینترنت اشیا به نقطه دسترسی^۱ 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. در صورتی که درخواستی باید در ابر پاسخ داده شود، مؤلفه کنترل پذیرش^۲ این درخواست را به دروازه‌ی ابر^۳ (CG) ارسال می‌کند. وقتی که 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 مؤلفه‌ی مدیر تامین پویای سرویس^۴ (DSPM) فرستاده می‌شود. این مؤلفه که مؤلفه‌ی 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. وقتی که 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^۵ متفاوت در نظر می‌گیریم که هر دامنه مه به مجموعه‌ای از دستگاه‌های اینترنت اشیا سرویس‌دهی می‌کند. هر دامنه مه شامل مجموعه‌ای از چند گره مه^۶ و یک گره کنترل دامنه^۷ 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 اشیا 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 مؤلفه‌های اصلی چارچوب پیشنهادی می‌پردازیم:

۳-۱-۱- لایه اشیا

لایه‌ی اول یا پایین‌ترین لایه در معماری پیشنهادی شامل دستگاه‌های اینترنت اشیا 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 مؤلفه

⁵ Fog Domain

⁶ Fog Node

⁷ Control Node

¹ Access Point

² Admission Control

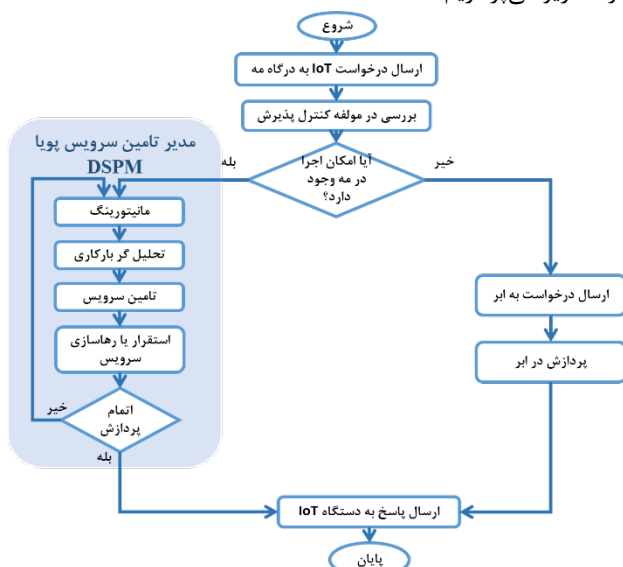
³ Cloud Gateway

⁴ Dynamic Service Provisioning Manager

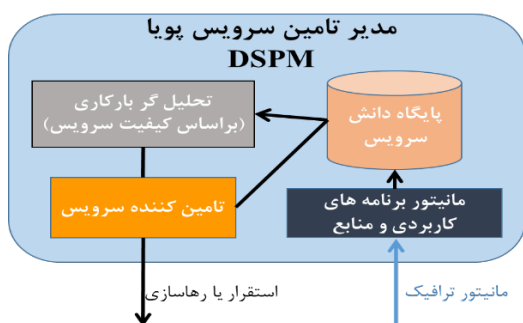


پذیرش مشخص نموده که باید در لایه‌ی مه اجرا شوند، بصورت پویا تامین کند. برای این کار به اطلاعات مربوط به دامنه‌های مه و گره‌های موجود در آن‌ها نیاز دارد. مولفه DSPM خود یکی از گره‌های مه موجود در لایه‌ی مه است. اما از آنجا که کار اصلی تصمیم‌گیری در رابطه با تامین سرویس‌های موردنیاز به عهده این مولفه می‌باشد 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 مولفه‌ی 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

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 مولفه DSPM می‌دهد. DSPM تصمیم می‌گیرد که چه تعداد منابع برای هر درخواست در نظر گرفته شود و در نهایت نتیجه تصمیم‌گیری را به گره کنترل دامنه مه ارسال می‌کند و این مولفه براساس اطلاعاتی که از مولفه DSPM به دست آورده است تصمیم می‌گیرد که چه تعداد از منابع را خاموش و چه تعداد را روشن کند تا در این صورت منابع بتوانند به صورت کارآمد مورد استفاده قرار گیرند. در صورتی که منابع مه تحت کنترل گره کنترل دامنه مه، تماماً مشغول باشند یا از دسترس خارج شده 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^۱ به مولفه کنترل پذیرش ارسال می‌شوند و این مولفه تصمیم می‌گیرد که هر درخواست بر اساس احتیاجاتش، به لایه‌ی مه فرستاد شود و یا مستقیماً به لایه‌ی ابر منتقل گردد. در صورتی که درخواستی بلادرنگ بوده و نسبت به تاخیر زمانی حساس باشد، باید وارد لایه‌ی مه شود، در نتیجه این مولفه آن درخواست را به مولفه‌ی DSPM در لایه‌ی مه می‌فرستد و در صورتی که باید به لایه‌ی ابر منتقل شود، این مولفه 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 بیشتری 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، در نتیجه سرویس، به یک گره با منابع بیشتر ارسال می‌شود. اما چنانچه سرویس موردنظر به منابع کمتری نیاز داشته باشد، ممکن است با در اختیار 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

^۱ Fog Gateway

سناریوی اول: این سناریو زمانی اتفاق می افتد که درخواست های رسیده به مؤلفه کنترل پذیرش دارای شرایطی هستند که باید در ابر پردازش شوند (مثلاً محدودیت زمانی برای پاسخ ندارند و یا نیازمند حافظه ای با حجم ذخیره سازی بالا می باشند و...) در این صورت درخواست مستقیماً وارد ابر می شود (مسیر سمت راست در شکل (۲)).

سناریوی دوم: این سناریو زمانی اتفاق می افتد 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 می دهد. در این مسیر مؤلفه DSPM قرار دارد که عنصر اصلی چارچوب پیشنهادی ما می باشد که در ادامه به صورت مفصل به آن پرداخته خواهد شد. شکل (۳) اجزاء مؤلفه DSPM را نشان می دهد. این مؤلفه از چندین واحد به شرح ذیل تشکیل شده است:

مؤلفه مانیتورینگ منبع و بینامه کاربردی:^۱ این واحد اطلاعات مربوط به منابع مه و سرویس های درخواست شده توسط دستگاه های اینترنت اشیا را جمع آوری می کند و از دو زیر مؤلفه مانیتور منبع و مانیتور سرویس های درخواست شده توسط کاربران تشکیل شده است. مؤلفه مانیتور منبع، به گره های کنترل دامنه در دامنه های مه مراجعه کرده و از آنجا اطلاعات مربوط به منابع مختلف مه (مانند: تعداد منابع روشن، تعداد منابع خاموش و ظرفیت هر منبع و ...) را دریافت می کند. به عبارت دیگر این مؤلفه مسئول جمع آوری اطلاعات در مورد منابع مه می باشد. مؤلفه مانیتور سرویس های درخواستی کاربر، مسئول جمع آوری اطلاعات در مورد سرویس های درخواستی کاربران (دستگاه های اینترنت اشیا) و احتیاجات آنها می باشد. اطلاعات مانیتور شده توسط این دو زیرمؤلفه، جمع آوری و یکپارچه شده و سپس برای استفاده توسط سایر واحدهای DSPM در پایگاه دانش سرویس^۲ ذخیره می شود.

پایگاه دانش سرویس: در این واحد، اطلاعات تمامی سرویس های درخواستی رسیده از طرف دستگاه های اینترنت اشیا و نیز وضعیت منابع موجود در مه، توسط واحد مانیتورینگ ذخیره می شود. مدیریت دانش این پایگاه به عهده واحد تامین سرویس می باشد. پس از هر بار تخصیص یا بازپس گیری منبع و همچنین پس از هر استقرار یا رهاسازی سرویس در یک گره مه، اطلاعات این پایگاه دانش توسط واحد تامین سرویس به روزرسانی و مدیریت می شود.

تحلیل گر بارکاری:^۲ این واحد با پردازش داده های جمع آوری شده از مؤلفه مانیتور در پایگاه دانش، اطلاعاتی در مورد وضعیت بهره وری فعلی سیستم و پیش بینی هایی از نیازهای آتی، به دست می آورد. چون درخواست های رسیده از دستگاه های اینترنت اشیا پویا هستند و متناسب با زمان تغییر می کنند، در این قسمت بر اساس تاریخچه درخواست های اینترنت اشیا، وضعیت آینده سیستم پیش بینی می شود. در این مرحله از تکنیک های سری های زمانی جهت پیش بینی وضعیت سیستم استفاده می شود. سپس نتایج به دست آمده در این

واحد در داخل پایگاه دانش سرویس قرار می گیرد تا این اطلاعات برای واحدهای بعدی مورد استفاده قرار گیرد.

مؤلفه تامین سرویس:^۴ پس از مشخص شدن وضعیت آینده سیستم توسط واحد تحلیل گر بارکاری، واحد تامین سرویس، بر اساس اطلاعات تحلیل شده در پایگاه دانش سرویس، برنامه ریزی می کند که کدام سرویس بر روی کدام گره مه مستقر، یا از روی کدام گره مه رهاسازی شود. واحد تامین سرویس، تصمیم گیرنده و هسته اصلی DSPM می باشد. در این قسمت با توجه به تحلیل انجام شده در مؤلفه تحلیل گر بارکاری و با استفاده از یک اتوماتای یادگیر تصمیم گرفته می شود که چه عملی انجام شود (سرویس بر روی گره مه مستقر شود یا از روی آن رهاسازی شود). در نهایت تصمیم گرفته شده در پایگاه دانش سرویس ذخیره می شود. 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 گره های مه مورد نظر در دامنه خود اعمال می کند.

۳-۱-۲- لایه ای ابر

بالا ترین لایه در چارچوب پیشنهادی، لایه ای ابر است که شامل سرورهای قوی و مراکز داده با قابلیت ذخیره سازی بالا می باشد. درخواست های اینترنت اشیا که حساس به تاخیر نیستند و همچنین درخواست هایی که به پردازش و ذخیره سازی بالایی نیاز دارند، پس از ورود به مؤلفه کنترل پذیرش، به این لایه ارسال می شوند (طبق سناریو اول که قبلاً ذکر شد). این درخواست ها ابتدا مستقیماً از مؤلفه کنترل پذیرش به دروازه ای ابر منتقل می شوند و سپس وارد ابر می شوند. بسته به اینکه کدام سرور این نوع درخواست را اجرا می کند، درخواست به صف اختصاص داده شده آن سرور می رود و در نوبت خود به سرور مربوطه وارد می شود. در چارچوب پیشنهادی، مدیریت درخواست های ارسال شده به دروازه ای لایه ای ابر، تماماً بر عهده لایه ای ابر بوده (از این جهت که درخواست به چه منبعی ارسال شود یا بر روی چه سروری مستقر شود) و نتایج به دست آمده، از طریق دروازه ای لایه ای ابر برای ارسال به دستگاه اینترنت اشیا، به مؤلفه کنترل پذیرش در لایه ای دوم ارسال می شود.

۳-۲- الگوریتم پیشنهادی

در این بخش، رویکرد تامین سرویس پویای پیشنهادی را با جزئیات شرح می دهیم. الگوریتم پیشنهادی ما منطبق بر مؤلفه DSPM است و در فاصله های زمانی مختلف (τ) تکرار می شود. در این الگوریتم با توجه به اجزای مؤلفه DSPM، به ترتیب عملیات مانیتورینگ، تحلیل حجم کاری و ارائه سرویس در بخش های Monitoring (خط ۳)، Workload_Analyzer (خط ۱۴) و Service_Provisioner (خط ۲۱) انجام می شود. در هر بار تکرار الگوریتم، درخواست های ورودی از

³ Workload Analyzer

⁴ Service Provisioner

¹ Application and Resource Monitor

² Service Knowledge Base



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 $Info$) 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 $data_msg$) 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 $data_info$) 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 پیش‌بینی 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 داده‌های 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 $ARIMA$ ^۱ 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 $ARIMA$ مقادیری 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 خودگردانده)، 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 $traffic_data_Array$ 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.

^۱ Autoregressive integrated moving average

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 الگوریتم 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^۷ 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 می‌شود. 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 می‌یابد، 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:

اتوماتای یادگیر [۲۷،۲۸،۲۹]، یکی از روش‌های یادگیری تقویتی^۱ 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 حالت‌های خود را تغییر دهد و در هر حالت نیز عمل^۲ (اقدام) خاص خود را اجرا نماید. اتوماتای یادگیر^۳ (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 به صورت 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 به صورت 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 اندازه‌ی آن، احتمال هر عملی کاهش یا افزایش می‌یابد. این روند تکرار می‌شود و به مرور زمان اتوماتای یادگیر کامل‌تر می‌شود و می‌تواند 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: 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: 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

⁵ Environment

⁶ Learning Algorithm

⁷ Violation Percent

¹ Reinforcement Learning

² State

³ Action

⁴ Learning Automata



$$T_{DSPM_Proc} = T_{Monitoring} + T_{Workload_Analyzer} + T_{Service_Provisioner} \quad (1)$$

که در آن $T_{Monitoring}$ پیچیدگی بخش مانیتورینگ (خطوط ۳ تا ۱۳ الگوریتم پیشنهادی)، $T_{Workload_Analyzer}$ پیچیدگی بخش تحلیل گر بار کاری (خطوط ۱۴ تا ۲۰ الگوریتم پیشنهادی) و $T_{Service_Provisioner}$ پیچیدگی بخش ارائه دهنده سرویس (خطوط ۲۱ تا ۳۷ الگوریتم پیشنهادی) می باشد. برای راحتی، فرض می کنیم که $|A|$ برابر با تعداد سرویس های اینترنت اشیا و $|F|$ برابر با تعداد گره های مه است. در بخش مانیتورینگ ابتدا برای هر سرویس اینترنت اشیا، دستوراتی اجرا می شوند که احتیاجات آن سرویس را مشخص می کند و سپس برای هر گره مه، دستوراتی جهت تعیین ظرفیت های آن گره اجرا می شود. بنابراین پیچیدگی بخش مانیتورینگ طبق (۲) بدست می آید:

$$T_{Monitoring} = O(|A|) + O(|F|) \quad (2)$$

بخش تحلیل گر بار کاری، حجم بار کاری آینده را با استفاده از تابع $ARIMA$ برای هر سرویس اینترنت اشیا در هر گره مه پیش بینی می کند. بنابراین، پیچیدگی بخش تحلیل گر بار کاری طبق رابطه (۳) (توجه داشته باشید که T_{ARIMA} یک مقدار ثابت است، زیرا تابع $ARIMA$ در راهکار پیشنهادی ما همواره تعداد معینی از مقادیر را پیش بینی می کند) به دست می آید:

$$T_{Workload_Analyzer} = O(|A| \times |F| \times T_{ARIMA}) \\ = O(|A| \times |F|) \quad (3)$$

پیچیدگی ارائه دهنده سرویس را می توان با (۴) بیان کرد. مقدار ثابت ۳۰۰ همان تعداد تکرار مراحل یادگیری در اتوماتای یادگیر است.

$$T_{Service_Provisioner} = O(300 \times |A| \times |F|) \\ = O(|A| \times |F|) \quad (4)$$

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

$$T_{DSPM_Proc} = O(|A|) + O(|F|) + O(|A| \times |F|) + O(|A| \times |F|) \quad (5)$$

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

$$T_{DSPM_Proc} \approx O(|A| \times |F|) \quad (6)$$

۴- ارزیابی و عملکرد نتایج

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

۴-۱- توابع هدف

در این بخش توابع هدف مورد ارزیابی در آزمایش ها را شرح می دهیم: **هزینه:** اولین پارامتر ارزیابی ما هزینه می باشد. منظور از هزینه هزینه سرویس دهی به یک درخواست دستگاه اینترنت اشیا می باشد که شامل هزینه منابع مورد استفاده، هزینه ارتباطی و هزینه جریمه می شود. هزینه منابع مورد استفاده خود شامل هزینه

منابع ابر، هزینه منابع مه، هزینه منابع مورد استفاده در مولفه کنترل پذیرش و هزینه منابع مورد استفاده در مولفه DSPM می باشد که هر یک از این موارد شامل هزینه CPU، حافظه اصلی و فضای ذخیره سازی مورد نیاز برای اجرای سرویس در یک سرور ابری یا در یک گره مه به همراه هزینه پردازش، حافظه و فضای ذخیره سازی مورد نیاز در مولفه کنترل پذیرش و مولفه DSPM می باشد. هزینه ارتباطی شامل هزینه ارتباط بین گره های موجود در لایه مه و سرورهای لایه ابر، هزینه ارتباط بین گره ها در لایه مه با همدیگر و هزینه ارتباط بین مولفه DSPM و گره های مه می باشد. هزینه جریمه برابر با هزینه ای است که مولفه DSPM می بایست در صورت وقوع نقض تأخیر متحمل پرداخت آن شود. در آزمایش های انجام شده ما مقدار متوسط پارامتر هزینه را محاسبه می کنیم.

تأخیر سرویس: مدت زمان بین ارسال یک درخواست سرویس توسط یک دستگاه اینترنت اشیا و دریافت یک پاسخ برای آن تأخیر سرویس نامیده می شود. متوسط تأخیر سرویس برای یک سرویس، پارامتر دوم در ارزیابی های ما می باشد. این پارامتر شامل میانگین تأخیر انتشار بین دستگاه های اینترنت اشیا و یک گره مه، میانگین تأخیر انتشار میان یک گره مه و یک سرور ابری، زمان انتظار سرویس درخواست شده در گره مه (شامل تأخیر مربوط به انتظار در صف انتظار گره مه و تأخیر پردازش سرویس در آن گره مه)، زمان انتظار سرویس درخواست شده در سرور ابری (شامل تأخیر مربوط به انتظار در صف انتظار سرور ابری و تأخیر پردازش سرویس در آن سرور ابر)، میانگین تأخیر انتقال یک درخواست از بین یک دستگاه اینترنت اشیا به یک گره در لایه مه، میانگین تأخیر انتقال یک درخواست بین یک گره در لایه مه و یک سرور در لایه ابر می باشد.

نقض تأخیر: پارامتر سوم مورد ارزیابی در آزمایش های ما نقض تأخیر می باشد. نقض تأخیر نیز یکی از معیارهای مهم برای اندازه گیری کیفیت خدمات است که در واقع برابر با درصد درخواست هایی است که در مهلت مقرر به آن ها پاسخ داده نمی شود.

۴-۲- تنظیمات شبیه سازی

آزمایشات ما در محیط جاوا توسعه یافته و در آن ها از مجموعه ابزار iFogsim برای مدل سازی موجودیت های محیط مه [۳۲] (به عنوان مثال، سرورهای ابری، گره های مه و دستگاه های اینترنت اشیا) استفاده شده است. علاوه بر این، ما تجزیه و تحلیل تجربی خود را بر روی رایانه ای با پردازنده Intel Core i3، 4 گیگابایت RAM، 500 گیگابایت فضای ذخیره سازی بر روی دیسک و در محیط Windows 64bit انجام داده ایم. ما از ردیابی ترافیک واقعی به نام MAWI Working Group برای دستیابی به نتایج واقعی تر در سناریوهای مختلف استفاده کرده ایم [۳۳].



Algorithm : DSPM Proc

```

1: Begin
2: for each (Time interval  $\tau$ ) do
    //Monitoring
3:   for each (IoT Service  $s$ ) do
4:     Receive  $req_{msg}$  for  $s$ 
        // The Extraction() function extracts
         $Deadline, Processing\_required,$ 
         $Memory\_required, Storage\_required$ 
        and  $Bandwidth\_required$  from inside the message
5:      $Sdata\_info = \text{Extraction}(req_{msg})$  //  $Sdata\_info$  is IoT service
        information record
6:     Update Service_Database with  $Sdata\_info$ 
7:   end for each
8:   for each (Fog node  $f$ ) do
9:     Connect to  $f\_node$ 
10:    Receive  $fdata\_msg$  from  $f\_node$ 
        // The Extraction() function extracts  $CPU\_Capacity,$ 
         $Memory\_Capacity$  and  $Storage\_Capacity$ 
        from inside the message
11:     $fdata\_info = \text{Extraction}(fdata\_msg)$  //  $fdata\_info$  is
        information record for fog node  $f$ 
12:    Update Service_Database with  $fdata\_info$ 
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:  $P_{Deploy}(t=1) = 1 -$ 
 $P_{Release}(t=1), P_{Release}(t=1) = \text{Initial-value}()$ 
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 ( $\alpha_i$ ) = Select one action with maximum
      probability; // Max( $P_{Deploy}, P_{Release}$ )
35:      Return Selected action ( $\alpha_i$ ) 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^۱ مصنوعی نیز استفاده می‌کنیم که یک بار کاری مصنوعی را براساس زنجیره مارکوف زمان-گسسته^۲ (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، از TCP و UDP برای انتقال پیام‌های خود استفاده می‌کنند [۳۴]. در این پژوهش، ما فرض می‌کنیم که جریمه نقض سرویس^۳، 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 سناریو 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 بر 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 در ثانیه^۴ (MIPS) است و ظرفیت پردازش هر سرور ابری بین ۱۶۰۰۰ تا ۲۶۰۰۰ میلیون دستورالعمل در ثانیه (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 واقعی افزوده موبایل^۵ (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

^۴ Million Instructions Per Second^۵ Mobile Augmented Reality^۱ traffic generator^۲ Discrete-Time Markov Chain^۳ Penalty Price

تعداد سرویس‌های اینترنت اشیا به ترتیب ۳، ۱۰ و ۴۰ تنظیم شده است. همچنین فاصله ردیابی ترافیک و فاصله زمانی t ، برابر با ۹۰۰ ثانیه است. شکل (۴ الف) و (۴ ب) به ترتیب نرخ ترافیک ورودی به گره‌های مه و میزان تأخیر 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 تأخیر سرویس را دارد. در مقایسه با روش All-Cloud، روش Min-Viol به دلیل تمرکز بر کاهش نقض تأخیر، تأخیر سرویس بسیار کمتری دارد. اما روش پیشنهادی ما کمترین تأخیر سرویس را به دلیل تکنیک پیش‌بینی ترافیک مورد استفاده برای ارائه سرویس‌های اینترنت اشیا به دست می‌آورد. میزان تأخیر در روش Fog-Static نیز بیشتر از روش‌های Min-Viol و روش پیشنهادی که به صورت پویا عمل می‌کنند، اما کمتر از روش All-Cloud است، زیرا در این روش جلیایی 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 اشیا، 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 تأخیر 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 گره‌های مه=۱۰ تعداد سرویس‌ها=۴۰ فاصله زمانی تغییر ترافیک=۹۰۰ ثانیه فاصله زمانی t =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 گره‌های مه=۱۰ تعداد سرویس‌ها=۲ فاصله زمانی تغییر ترافیک=۶۰ ثانیه فاصله زمانی t =۱۲۰ ثانیه
سناریوی سوم	بررسی تأثیر اندازه حدآستانه تأخیر سرویس	12:00 PM to 4:00 PM of 2023/04/12 برابر با یک ردیابی ترافیک ۴ ساعته	تعداد سرورهای ابری=۳ تعداد گره‌های مه=۱۰ تعداد سرویس‌ها=۲۰ فاصله زمانی تغییر ترافیک=۱۰ ثانیه فاصله زمانی t =۱۰ ثانیه

درخواست است [۳۵]. همچنین میزان فضای ذخیره‌سازی مورد نیاز برای این برنامه‌ها بین ۵۰ تا ۵۰۰ مگابایت و مقدار حافظه مورد نیاز 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 تأخیر 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- تحلیل نتایج

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 All-Cloud، Optimal، Fog-Static [۷] و Min-Viol [۱۲] مقایسه می‌کنیم. اولین روش Optimal نامیده می‌شود و یک راه حل بهینه است که بهترین روش جایایی برای سرویس‌های اینترنت اشیا در گره‌های مه را فراهم می‌کند. در این روش تمامی حالات ممکن برای جایایی 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 اشیا 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 اشیا را در نظر می‌گیرد. روش چهارم Fog-Static [۷] نامیده می‌شود، که یک رویکرد ایستا می‌باشد که تمام سرویس‌های اینترنت اشیا 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 برنامه‌های 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 سناریوی دوم، مکانیزم پیشنهادی با راه‌حل بهینه مورد ارزیابی قرار گرفته و سناریوی سوم به بررسی تأثیر اندازه حدآستانه تأخیر سرویس‌های اینترنت اشیا 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 ۱۲ و ۱۳ آوریل ۲۰۲۳) استفاده کردیم. علاوه بر این، تعداد سرورهای ابری، تعداد گره‌های لایه‌ی مه و

از آنجایی که روش Min-Viol یک روش جایابی پویا است و بر کاهش نقض تأخیر 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 تأخیر در الگوریتم‌های مذکور را نشان می‌دهد. بر اساس نتایج تجربی، مشخص شد که مکانیسم پیشنهادی کمترین میزان نقض تأخیر را (در حدود ۳ درصد) دارد، زیرا روش پیشنهادی ما مبتنی بر کاهش نقض تأخیر است. این حتی نسبت به روش Min-Viol که بر کاهش نقض تأخیر تمرکز دارد نیز کمتر است. همچنین بیشترین میزان نقض تأخیر مربوط به روش All-Cloud بوده و میزان نقض تأخیر 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

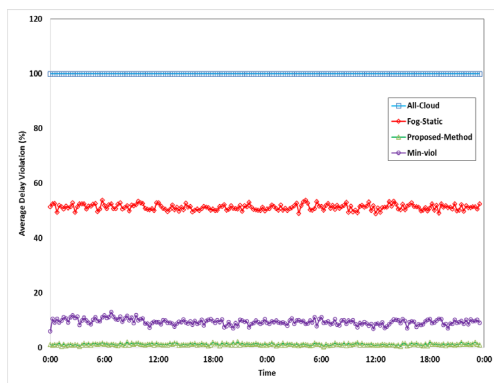
در این سناریو، ما مکانیزم پیشنهادی خود را در مقایسه با راه‌حل‌های بهینه و روش Min-Viol مورد ارزیابی قرار می‌دهیم. در راه‌حل بهینه، ما همه حالت‌های ممکن جهت استقرار سرویس‌های اینترنت اشیاء در گره‌های لایه‌ی مه و سرورهای لایه‌ی ابر را بررسی می‌کنیم تا بهترین راه‌حل را با کمترین هزینه بدست آوریم. ما در 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 بر این، 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 در نظر گرفته شده است. شکل (۵ الف) و (۵ ب) به ترتیب نرخ ترافیک ورودی به گره‌های مه و تأخیر سرویس را نشان می‌دهند. همان‌طور که مشاهده می‌شود، بیشترین و کمترین میزان تأخیر سرویس به ترتیب متعلق به روش‌های All-Cloud و Optimal است. میانگین تأخیر 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 تأخیر 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 تأخیر 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 تأخیر روش‌های پیشنهادی و پایه را نشان می‌دهد. همان‌طور که مشاهده می‌شود، بیشترین میزان نقض تأخیر 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 تأخیر 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.

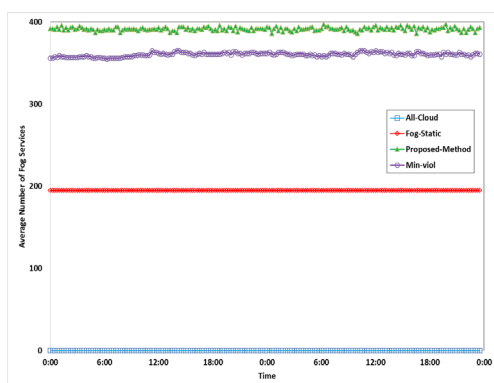
• **سناریوی سوم: بررسی تأثیر حد آستانه تأخیر سرویس**
این سناریو تأثیر اندازه حد آستانه تأخیر را در روش پیشنهادی ما مورد بحث قرار می‌دهد. ما از مجموعه داده ردیابی ترافیک در دنیای واقعی در تاریخ ۱۲/۰۴/۲۰۲۳ به مدت ۴ ساعت (۱۲:۰۰ بعدازظهر تا ۴ بعدازظهر) استفاده کرده‌ایم. علاوه بر این، تعداد سرورهای ابری، تعداد گره‌های لایه‌ی مه و تعداد سرویس‌های اینترنت اشیاء به ترتیب برابر با ۱، ۱۰ و ۲۰ تنظیم شده است. شکل (۶ الف) تأخیر سرویس را در روش‌های پیشنهادی و پایه نشان می‌دهد. با توجه به نتایج به دست آمده، با افزایش اندازه حد آستانه‌ی تأخیر مربوط به سرویس، الگوریتم‌ها اغلب سرویس‌های اینترنت اشیاء را به سرورهای ابری منتقل می‌کنند. بنابراین سرویس‌ها به دلیل عدم سرویس‌دهی در لایه‌ی مه با تأخیر زیاد انجام می‌شود. در نتیجه میانگین تأخیر سرویس نیز افزایش می‌یابد و در نهایت به نتایج All-Cloud می‌رسد. همچنین، کمترین میانگین تأخیر 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 تأخیر 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 تأخیر 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 ذکر شده عملکرد یکسانی خواهند داشت. شکل (۶ ج) نقض تأخیر تمام



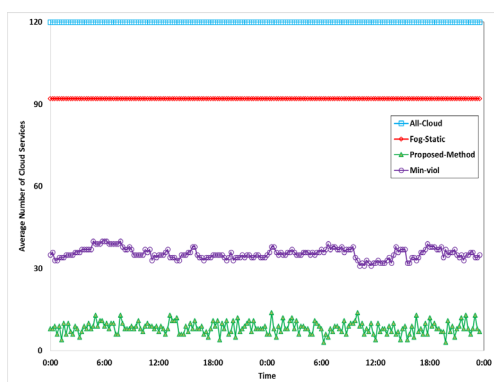
مکانیسم‌ها را نشان می‌دهد. نقض تاخیر روش پیشنهادی کمترین است، در حالی که رویکرد All Cloud بیشترین نقض تاخیر را دارد. در مقادیر کوچکتر حدآستانه‌ی تاخیر، کمترین نقض تاخیر مربوط به مکانیزم پیشنهادی است و در مقادیر بزرگتر حدآستانه تاخیر، همه روشها یکسان هستند. با افزایش حد آستانه تاخیر، نقض تاخیر در همه 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 اشیا 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.



(د) نقض تاخیر سرویس

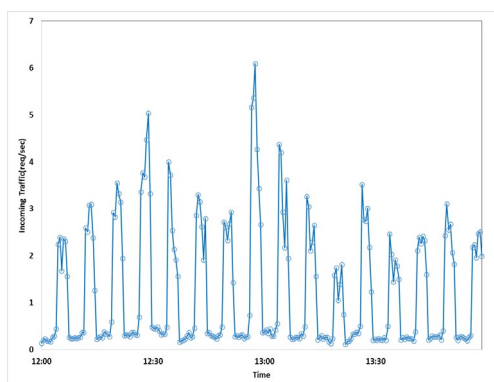


(ه) میانگین تعداد سرویس‌های مه

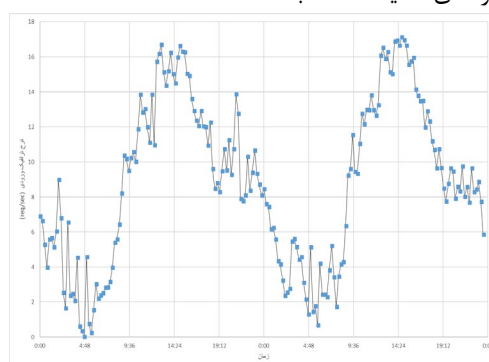


(و) میانگین تعداد سرویس‌های ابر

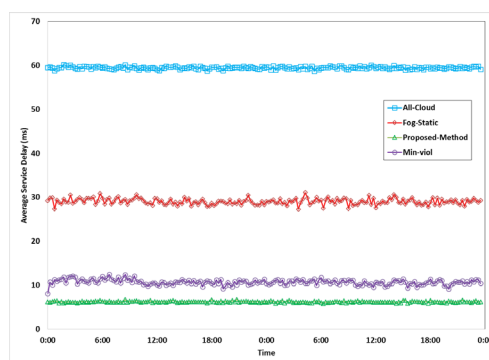
شکل (۴): نتایج شبیه سازی سناریوی اول



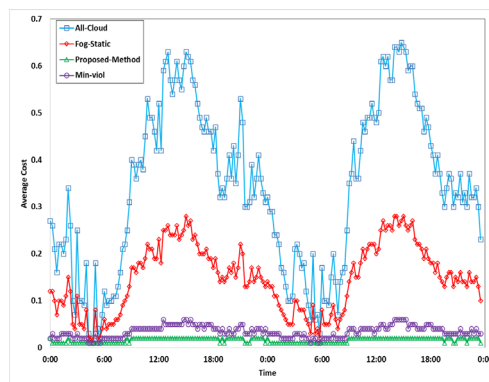
(الف) نرخ ترافیک ورودی



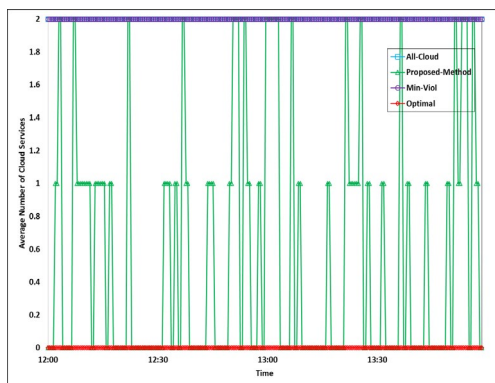
(الف) نرخ ترافیک ورودی



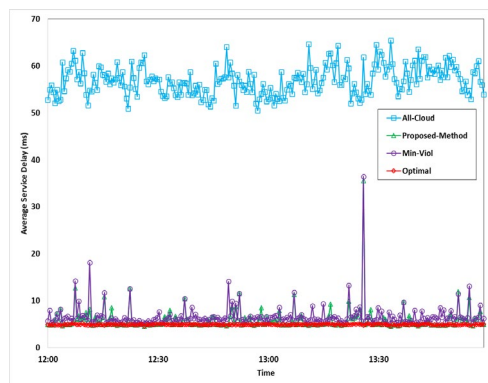
(ب) تاخیر سرویس



(ج) هزینه

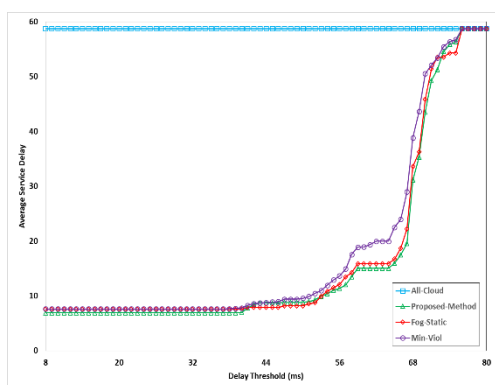


(و) میانگین تعداد سرویس‌های ابر

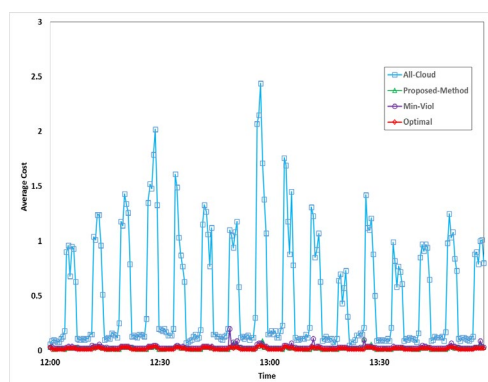


(ب) تاخیر سرویس

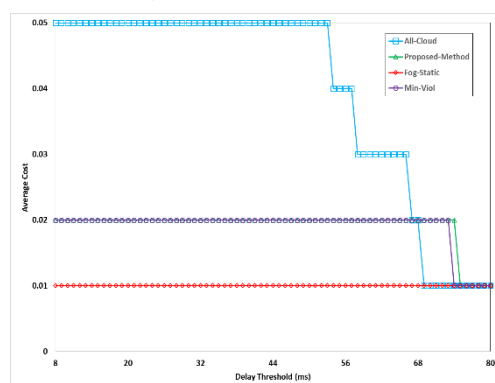
شکل (۵): نتایج شبیه سازی سناریوی دوم



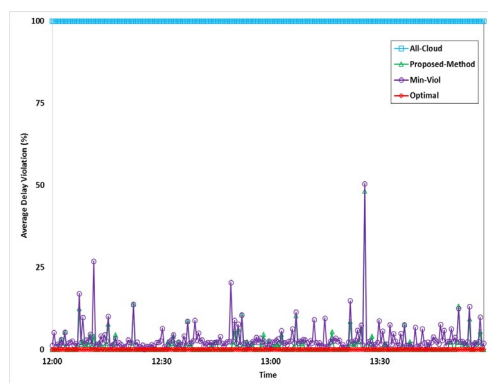
(الف) تاخیر سرویس



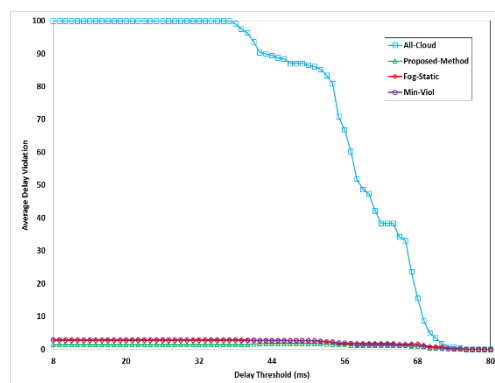
(ج) هزینه



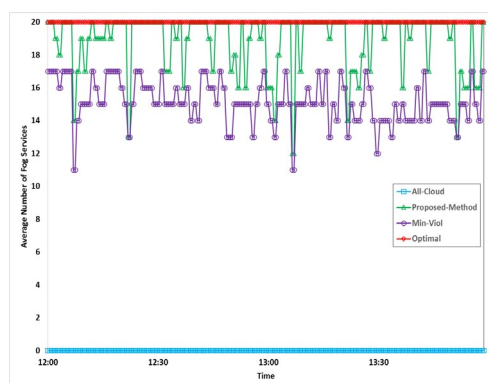
(ب) هزینه



(د) نقض تاخیر سرویس



(ج) نقض تاخیر سرویس

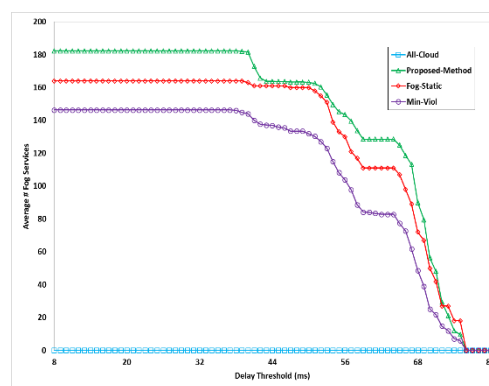


(ه) میانگین تعداد سرویس‌های مه

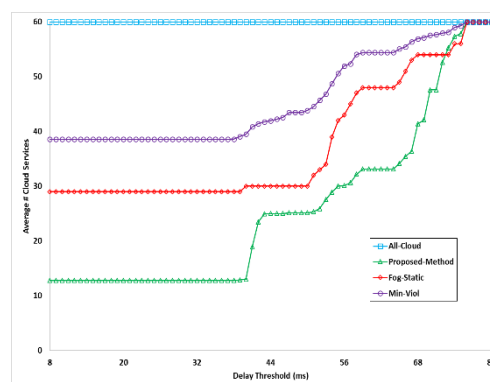


مرجع

- [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.



(د) میانگین تعداد سرویس‌های مه



(ه) میانگین تعداد سرویس‌های ابر

شکل (۶): نتایج شبیه سازی سناریوی سوم

۵- نتیجه گیری

با توجه به یافته‌های ارائه‌شده، رویکرد پیشنهادی که ترکیبی از روش‌های پیش‌بینی سری‌های زمانی و اتوماتای یادگیر است، توانسته است به‌طور مؤثری مسائل مربوط به تامین پویای سرویس در محیط مه را حل کند. این رویکرد نه تنها باعث بهینه‌سازی استفاده از منابع مه شده است، بلکه کیفیت سرویس‌دهی به کاربران را نیز به‌طور قابل توجهی بهبود بخشیده است. با به‌کارگیری تکنیک‌های پیش‌بینی بار کاری و اتوماتای یادگیر، تخصیص منابع به‌صورت هدفمند انجام شده و از هدررفت منابع جلوگیری می‌شود. همچنین، نتایج شبیه‌سازی نشان می‌دهد که این روش از نظر هزینه، کاهش تاخیر سرویس، نقض تاخیر و استقرار سرویس‌ها در مقایسه با روش‌های دیگر برتری دارد. علاوه بر این، استفاده از اتوماتای یادگیر به دلیل سادگی محاسبات و سرعت تصمیم‌گیری بالا، امکان تامین پویای سرویس را به‌صورت بهینه فراهم می‌کند. این امر منجر به افزایش رضایت کاربران، به‌ویژه در برنامه‌های کاربردی بلندمدت، می‌شود. با استقرار بیشتر سرویس‌ها در لایه‌ی مه و نزدیکی آن به دستگاه‌های اینترنت اشیا، زمان پاسخ‌دهی به‌طور قابل توجهی کاهش یافته و نیازهای برنامه‌های کاربردی بلندمدت به‌طور مؤثرتری برآورده می‌شود. در نهایت، این رویکرد پیشنهادی نه تنها کارایی سیستم را افزایش داده است، بلکه سطح کیفیت خدمات را نیز ارتقاء بخشیده است.

- [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.
- [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].

