Collaborative Task Offloading in Edge Computing Enabled Web 3.0

Mohammed Alkhathami

Information Systems Department, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia
E-mail: maalkhathami@imamu.edu.sa

Received 08 January 2024; Accepted 14 April 2024

Abstract

Web 3.0 is an evolved version of the Web that enables the integration of applications such as the Internet of Things (IoT) with the Web. It involves the storage of large data generated by different users and efficient computation of application and web-related tasks. With the help of edge nodes installed near the users, the computation load of Web 3.0 will be efficiently managed. Thus, efficient task offloading and computation become a key concern in edge computing-enabled Web 3.0. In this paper, a novel algorithm is proposed that solves the challenges of load imbalance at the edge nodes resulting in large queue sizes and increased task delays. The proposed technique identifies the edge nodes with a large network load and pairs them with a lightly loaded edge node that can handle some of their network load. The edge node pairing is based on the Gale–Shapley stable matching algorithm. The preference profile of edge nodes is developed based on factors such as task computation delay and task transmission delay. Once the pairing is done, the number of tasks is offloaded as per the computing capacity of the lightly loaded edge nodes. A detailed simulation-based performance evaluation of the proposed technique is presented showing a reduction in task delay by 20% and task deadline miss ratio by 68%.

Keywords: Web 3.0, task offloading, edge computing.

Journal of Web Engineering, Vol. 23_5, 681–698.
doi: 10.13052/jwe1540-9589.2354

1 Introduction

Web 3.0 is an upgrade of the current version of the web to make it compatible with new applications, data management, and analytic techniques [1–3]. Some of the key features of Web 3.0 are that it will support decentralized data sharing and intelligent processing of data. Moreover, Web 3.0 will also enhance security and reliability [4–6].

Web 3.0 can support many new applications such as the Internet of Things (IoT) that involve storing large amounts of data and computation of tasks [14, 15]. It will introduce decentralized data computation and management using Blockchain and edge computing. Many different types of sensors can interface with the web using this new version. Also, it will enable new communication protocols to facilitate IoT communications [7–9].

Web 3.0 will lead to a semantic web that provides more context and meaning to the current web. Moreover, it will lead to more secure data communications and data storage with the use of decentralized Blockchain-based identity management, access control, and encryption techniques [10–13].

Edge computing will be central to Web 3.0 as fast task computation and data storage is a key requirement of next-generation Internet-based applications [16–18]. Edge nodes will be installed near the end devices to meet the latency requirements of the Web 3.0-related tasks. However, edge computing-enabled task computation will bring challenges to Web 3.0. End users prefer to share their tasks with the nearest edge node due to communication-related constraints [19,20]. Thus, it is up to the edge layer to collaborate and manage the task computation load. The goal is to provide a balanced computation load to each edge server and reduce the task computation delay. The motivation of this work is to enable offloading of tasks for Web 3.0 using edge computing.

In this paper, the focus is on the problem of collaborative task computation by the Web 3.0-enabled edge nodes. The main contribution of the paper is as follows:

- An edge node classification algorithm is proposed to divide the nodes into two groups based on the number of tasks in the queue, expected task computation delay, and task deadline. The first group is the heavily loaded edge nodes and the other is lightly loaded edge nodes.
- To enable collaboration among the edge nodes and balance the computation load, edge nodes are paired to include a heavily loaded node and a lightly loaded node. The pairing problem is converted to a graph-matching problem and solved using the Gale–Shapley algorithm.

- The task offloading between a heavily loaded edge node and a lightly loaded edge node is performed based on the capacity of the light-loaded edge node.
- Detailed performance evaluation of the proposed technique as compared to a nearest task offloading technique and a matching-based task offloading technique is presented. Results highlight that the proposed technique achieves significant improvement in task delay and task deadline miss ratio.

The paper is organized as follows. Section 2 provides a brief overview of current work related to task offloading. Section 3 describes the system model. Section 4 explains the proposed technique in detail. Section 5 presents the performance evaluation of the proposed technique. Finally, Section 6 presents the conclusions.

2 Related Works

The work related to edge computing and task offloading has been going on for a few years; however, its application to Web 3.0 is relatively new and there are only a few related works. Table 1 provides a summary of the literature in this area. In [21], the focus of the work is on digital twin and edge computing-based Web 3.0. The goal of the work is to enable network virtualization and network digitization to meet the requirements of Web 3.0. A task offloading technique is proposed that uses base stations (BS) as edge servers. A Markov decision process (MDP) and a deep Q-learning-based technique are utilized to enable efficient task offloading. Simulation results show reduced task latency using the proposed technique.

The work in [22] proposes a task offloading mechanism for UAV and edge computing-based Web 3.0 networks. The goal of the work is to introduce reliable and secure computing for UAV networks. To enable this goal, a federated reinforcement learning (RL) based technique is proposed that improves the metric termed secure calculation capacity.

In [23], a Web 3.0-based Internet of Vehicles is considered. The key idea of the work is to enable efficient placement of edge servers which has a substantial impact on the task computation. The security of users is considered while making task-offloading decisions. Malicious users are identified using an anomaly detection algorithm. The proposed technique achieves improved task delay.

Table 1 Summary of related works

Reference	Network	Key Idea	Results
[21]	Web 3.0 Digital twin Edge computing	Network virtualization Network digitization Task offloading to BS Markov decision process Deep Q-learning	Reduced task delay
[22]	Web 3.0 UAVs Edge computing	Reliable computing Secure computing Federated RL	Secure calculation capacity
[23]	Web 3.0 Internet of Vehicles	Placement of edge servers Security consideration Anomaly detection	Improved task delay
[24]	IoT network Fog computing	Virtual resource units Stable matching Task deadline based allocation	Improved resource utilization
[25]	Vehicular network Edge computing	Greedy matching Vehicle mobility Vehicle connectivity Kuhn–Munkras algorithm	Improved task delay

The work in [24] is focused on IoT networks and fog computing. The proposal introduces the idea of using adaptive virtual resource units at the fog nodes. A matching-based technique is used to map IoT tasks and fog node resources. A task deadline is considered to utilize the virtual resource units. The work achieves improved resource allocation of the fog node capacity.

In [25], a vehicular network-based edge computing scenario is considered. A greedy matching technique is proposed that considers vehicle mobility and vehicle connectivity to allocate tasks at the edge nodes. A stable matching-based Kuhn–Munkras algorithm is used for task allocation. The work improves the delay of the tasks.

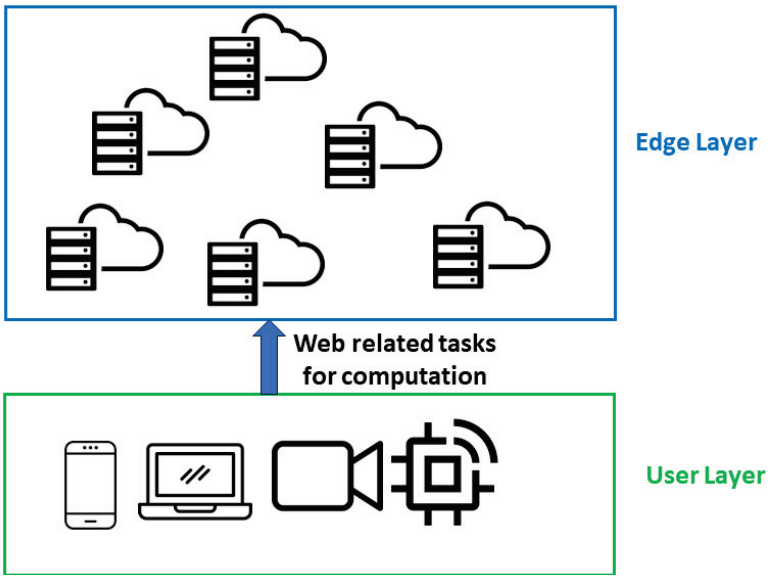

Figure 1 System model of edge computing enabled Web 3.0.

3 System Model

The considered system model in this paper is shown in Figure 1. There are two major layers in the system. The first is the user layer which generates data and tasks for computation whereas the second layer is the edge layer which comprises several edge servers that compute web-related tasks on behalf of users. The users transmit their tasks to the nearest edge server for computation purposes. The transmission between the user and the edge server is enabled using 5G communications. Similarly, tasks can be transmitted between edge nodes and this is also achieved using 5G communications.

Each edge server has a computation capacity which is measured in terms of two important metrics. The first metric is the number of bits per cycle (C) a CPU can process. The second metric is the speed of the CPU in terms of cycles per second (S). Thus, a task that is forwarded to the edge server takes T_{comp} time for computation given as follows:

$$T_{comp} = \frac{B}{S \times C} \tag{1}$$

where B is the size of the task in bits.

The incoming tasks at the edge server are placed in a queue for processing. Each edge server processes tasks one by one. This means that the task that is received earlier and placed at the top of the queue is processed first.

4 Proposed Technique

The goal of the proposed technique is to use collaboration among edge nodes to improve the task computing delay. To achieve this goal, a collaborative algorithm is proposed in this paper.

4.1 Edge Nodes Classification

Each edge node is classified as a heavy or light node depending on its computation capacity and the current size of its queue. Let Q be the number of tasks in the queue waiting for computation, then the computation time required for the last task in the queue can be given as:

$$T_{comp}^{last} = \sum_{n=1}^{Q+1} B_n \times \frac{1}{S \times C}. \tag{2}$$

Here $Q + 1$ is used to consider Q tasks in the queue and 1 task which is under computation. The size of each task B_n can vary.

The v^{th} edge node E_v is classified as light if it can compute all the tasks within the queue within the task deadline T_d.

$$E_v = \begin{cases} \text{Light}, & T_{comp}^{last} \leq T_d \\ \text{Heavy}, & \text{otherwise} \end{cases} \tag{3}$$

4.2 Collaborative Pairing using Stable Matching Algorithm

The proposed technique develops a collaboration mechanism among the edge nodes. The key idea is to pair each heavy edge node with a light edge node such that the latter can assist the former in computing tasks in its queue. To efficiently pair the edge nodes, the given problem is converted to a stable matching problem. The Gale–Shapley algorithm for stable matching is utilized to find a stable pairing among edge nodes, where each pair contains one node from a heavy and light group. The proposed algorithm runs in a centralized manner and one of the edge nodes acts as a server, thus maintaining data of all IoT nodes and edge nodes. The server edge node also

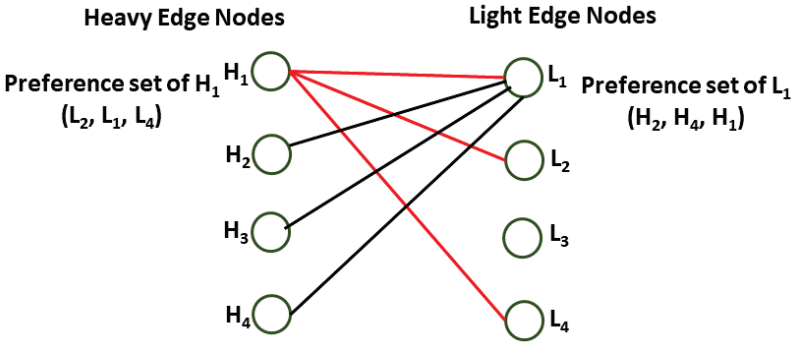

Figure 2 Preference profile of heavy and light edge nodes.

runs the classification algorithm as well as the task offloading technique. The algorithm is presented in Algorithm 1

4.2.1 Preference Profile

To apply the stable matching algorithm, the preference profile of each edge node in the heavy group towards every other edge node in the light group and vice versa is required, as shown in the bi-partite graph of Figure 2.

For the heavy edge nodes, the preference profile towards light edge nodes is calculated based on the following two factors:

- Transmission delay to transmit a task from the heavy edge node toward the light edge node.
- Total computation time to compute all the tasks in the queue (T_{comp}^{last}) of the light edge node.

The transmission delay is calculated using task size and data rate as follows:

$$T_{trans} = \frac{B}{R} \tag{4}$$

and the data rate for 5G wireless communications can be calculated from the Shannon equation.

Thus, the heavy edge nodes utilize the sum of Equations (2) and (4) to calculate their preference ranking towards light edge nodes. The reason for using this preference profile is that heavy edge nodes prefer to collaborate with light edge nodes that are closer to them, have a good wireless channel link, and are lightly loaded with computation tasks.

On the other hand, light edge nodes utilize a measure of load on the heavy edge node, i.e., T_{comp}^{last}, to evaluate the preferences. The reason is that a light

edge node will prefer to collaborate with a heavy edge node that has a lesser computational load so that it may receive fewer tasks for computation. This way the light edge node will save its processor utilization.

Algorithm 1: Proposed collaborative pairing technique of edge nodes

```
Classify edge nodes into heavy and light using equation 3
Find preference ranking of heavy edge nodes towards light edge nodes using
  equations 2 and 4
Find preference ranking of light edge nodes towards heavy edge nodes using
  equation 2
Initially, each heavy and light edge node is free
while a single heavy edge node v_h is free do
    v_l = highest ranked light edge node in the preference ranking of heavy edge
      node for which proposal has not been sent
    if v_l is free then
        collaboration between heavy edge node v_h and v_l is finalized
    end
    else
        if v_l prefers v_h to its current collaborative heavy edge node v'_h then
            collaboration between heavy edge node v_h and v_l is finalized
            Make v'_h free
        end
        else
            Proposal is not accepted and no change in status
        end
    end
end
Find computing space available in the light edge node using the difference of T_d and
  T_comp^last
Transmit the tasks to the light edge node
```

4.3 Stable Matching Algorithm

Once the preference profile is finalized, the stable matching algorithm is run by a central controller. As shown in Algorithm 1, each heavy edge node proposes to the light edge node which is in first rank in its preference profile. This process is continued till the time all edge nodes are matched and collaboration between them is finalized. It should be noted that during the algorithm, the light edge nodes may break their current collaboration in case they receive a proposal from a better heavy edge node. At the end of

Table 2 Details of simulation parameters

Simulation parameter	Value
Number of edge nodes	200
Number of user nodes	500–1500
Task size B	10–40 Mbits
Bits per cycle of CPU C	0.001
Speed of CPU S	5–10 GHz
Average queue size of edge node	2–8
Task deadline	50s
Data rate	20 Mbps

the algorithm, a one-to-one matching between heavy and light edge nodes is achieved and a stable pairing is obtained.

4.4 Task Transmission from Heavy to Light Edge Nodes

Finally, once the pairs are formed, the heavy edge node transmits some of its tasks to the light edge nodes. To calculate the number of tasks to be offloaded, the heavy edge node first calculates the tasks that cannot be computed within the deadline requirements. These tasks are marked for transmission to the collaborative light edge node. After this, the difference between T_d and T_{comp}^{last} of the light edge node is evaluated. This difference indicates how much computing space is available in the light edge node. Then, the heavy edge node transmits the tasks as per the computing space of the light edge node.

5 Results

The performance evaluation of the proposed technique is conducted in MATLAB software. The simulation parameters are listed in Table 2. The number of edge nodes is taken as 200. The number of user nodes is considered to be varying from 500–1500. Each user node generates one Web 3.0-related task for computation to the edge nodes. The task size B is varied from 10–40 Mbits. The bits per cycle of CPU C is taken as 0.001. The speed of CPU S is taken as 5–10 GHz. The average queue size of edge nodes is taken as 2–8 s. The task deadline is taken as 50 s. The data rate is taken as 20 Mbps.

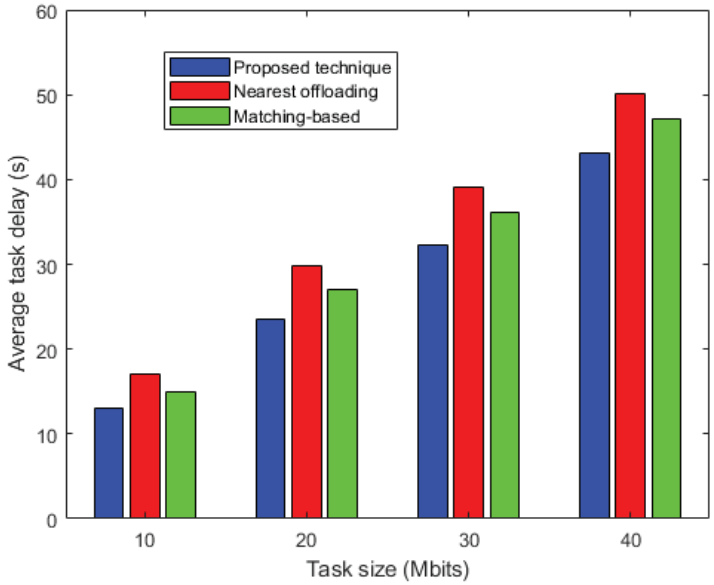

Figure 3 Task delay at different values of task size.

The performance of the proposed technique is compared against two other techniques from the literature:

- **Nearest offloading technique** in which user nodes transmit their task to the nearest edge node.
- **Matching-based offloading technique** in which user nodes transmit their task to the edge node considering channel quality, and available capacity of the edge node [24].

The performance evaluation is conducted based on two key metrics as follows:

- **Task delay** is the total time taken to transmit the task to the edge node and compute the task. It also includes the wait time required by the tasks in the queue of the edge node. Moreover, the proposed technique also includes the transmission time of the task from one edge node to the other.
- **Task deadline miss ratio** is the ratio of tasks that are not computed within their deadline, i.e., tasks whose task delay is greater than 50 s.

Unless otherwise stated, the following parameters are taken as default for the simulation. The task size B is equal to 30 Mbits. The speed of CPU S is taken as 5 GHz.

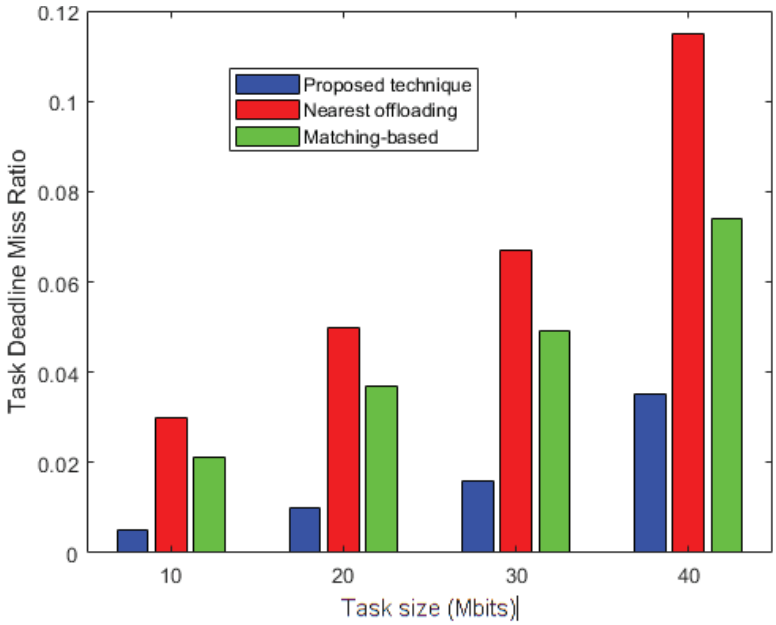

Figure 4 Task deadline miss ratio at different values of task size.

In Figure 3, the performance of the three techniques is presented at different task sizes. The results show that the proposed technique incurs the least amount of task delay at all task size values. As an example, at 30 Mbits, the proposed technique computes all tasks within 32 s. In comparison, the nearest task offloading technique requires 39.1 s and the matching-based task offloading needs 36 s to compute all the tasks. The reduction in task delay achieved by the proposed technique is due to the cooperative task offloading among the edge nodes which balances the network load. The edge nodes with a large task queue size offload their tasks to the lightly loaded edge nodes.

To investigate the impact of task sizes on the tasks that miss their deadlines, the plot is presented in Figure 4. It can be seen that the nearest task offloading can result in more than 11% of the tasks being computed beyond the deadline. On the other hand, matching-based task offloading can reduce this percentage to 7%. However, due to the absence of collaboration between edge nodes, this percentage is still very high. By using the proposed technique, this percentage can be reduced to 3%, thus resulting in an efficient task computation algorithm.

The task delay of edge nodes at different processor speeds is shown in Figure 5. Three different processor speeds of edge nodes are used varying

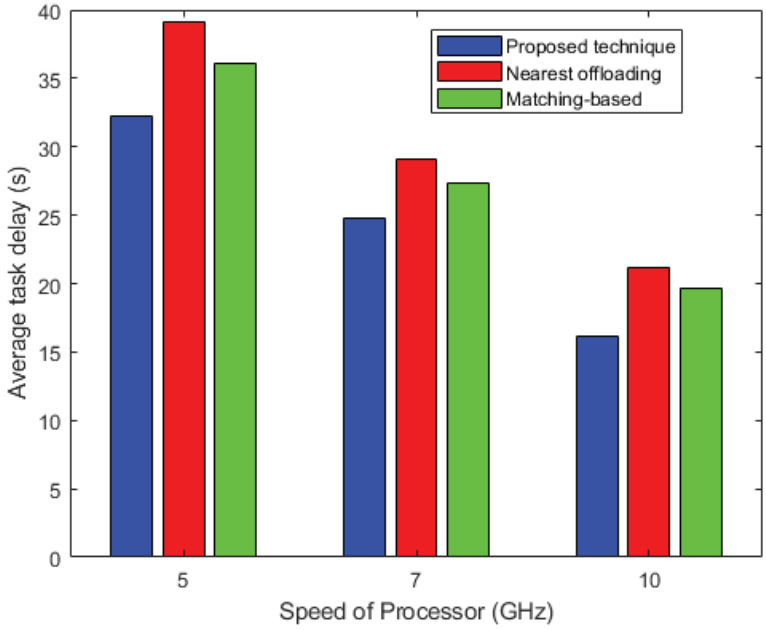

Figure 5 Task delay at different values of processor speed.

from 5–10 GHz. For all three techniques, the task delay is reduced with the increase in processor speeds. The proposed technique is the best-performing algorithm among the three due to its collaboration approach, thus resulting in task delay between 15.5 and 33 at different processor speeds. On the other hand, both other techniques result in a 3–7% increase in task delay.

The task deadline miss ratio at different values of processor speed is presented in Figure 6. The proposed technique results in 3–6% fewer number of tasks that lose their deadlines. In particular, at the processor speed of 5 GHz, the nearest task offloading causes 6.5% of the tasks to miss their deadline. This number is reduced to 4.8% by the matching-based task offloading.

To evaluate the impact of the queue size at the edge nodes, the result is presented in Figure 7. At low queue size values, the performance improvement of the proposed technique is only slightly better than the other techniques. This is because only a few tasks need shifting to the other collaborative edge nodes in this case. On the other hand, as the queue size grows, the performance gain of the proposed technique is much greater. For example, at a queue size of 7, the task delay is reduced by 7 s as compared to the other techniques.

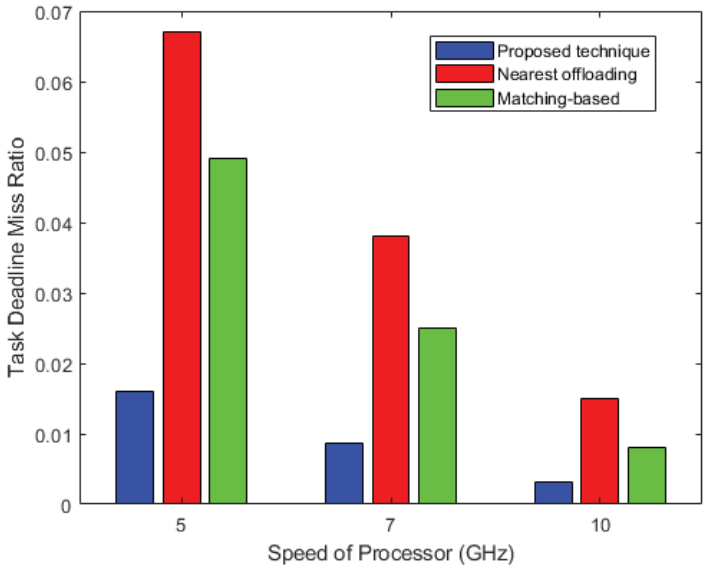

Figure 6 Task deadline miss ratio at different values of processor speed.

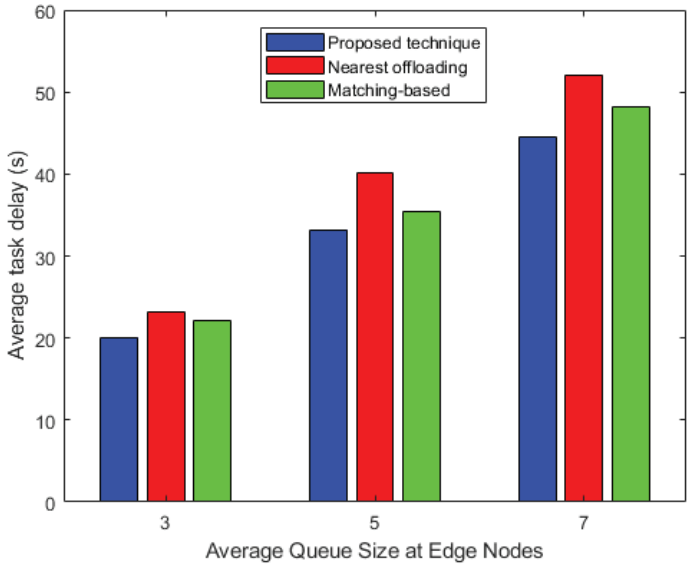

Figure 7 Task delay at different values of edge node queue size.

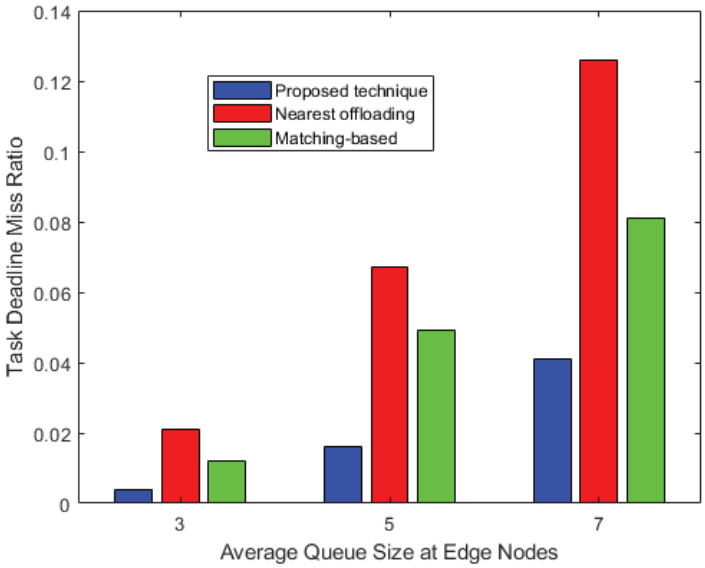

Figure 8 Task deadline miss ratio at different values of edge node queue size.

The task deadline miss ratio at different values of edge node queue size is shown in Figure 8. It can be seen that as the number of tasks in the queue of the edge node is increased, the task deadline miss ratio of the nearest offloading technique is significantly increased. The number reaches up to 12.5% at the queue size of 7. However, the proposed technique manages to keep the deadline miss ratio to less than 4% at the highest queue size of 7.

6 Conclusion

In this paper, a novel collaborative task offloading technique is proposed for edge computing-enabled Web 3.0 applications. The proposed technique classifies the edge nodes into heavy-loaded and light-loaded categories based on the network load and queue size. The work further develops a stable matching-based algorithm to find collaborative edge node pairs consisting of a heavy-loaded and light-loaded edge node. The tasks are offloaded from the heavily loaded edge nodes to their correspondingly allocated lightly loaded edge nodes. The performance evaluation of the proposed technique shows that it can reduce the task delay by 20% and task deadline miss ratio by 68%.

References

[1] Z. Yuan, Y. Tian, Z. Zhou, T. Li, S. Wang and J. Xiong, "Trustworthy Federated Learning Against Malicious Attacks in Web 3.0," in IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2024.3350365.

[2] Y. Chi, H. Duan, W. Cai, Z. J. Wang and V. C. M. Leung, "Knowledge Inference Over Web 3.0 for Intelligent Fault Diagnosis in Industrial Internet of Things," in IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2023.3344516.

[3] S. Yang, Y. Zhang, L. Cui, B. Deng, T. Chen and Q. Dong, "A Web 3.0-Based Trading Platform for Data Annotation Service With Optimal Pricing," in IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2023.3322817.

[4] Y. Lin et al., "A Unified Blockchain-Semantic Framework for Wireless Edge Intelligence Enabled Web 3.0," in IEEE Wireless Communications, doi: 10.1109/MWC.018.2200568.

[5] R. H. Kim, H. Song and G. S. Park, "Moving Real-Time Services to Web 3.0: Challenges and Opportunities," in IEEE Transactions on Services Computing, vol. 16, no. 6, pp. 4041–4059, Nov.–Dec. 2023, doi: 10.1109/TSC.2023.3307153.

[6] A. Antelmi, G. D'Ambrosio, A. Petta, L. Serra and C. Spagnuolo, "A Volunteer Computing Architecture for Computational Workflows on Decentralized Web," in IEEE Access, vol. 10, pp. 98993–99010, 2022, doi: 10.1109/ACCESS.2022.3207167.

[7] J. Chen, B. Qian, H. Zhou and D. Zhao, "A Decentralized Web 3.0 Platform for Manufacturing Customized Products," in IEEE Network, doi: 10.1109/MNET.2023.3318609.

[8] X. Zhang, G. Min, T. Li, Z. Ma, X. Cao and S. Wang, "AI and Blockchain Empowered Metaverse for Web 3.0: Vision, Architecture, and Future Directions," in IEEE Communications Magazine, vol. 61, no. 8, pp. 60–66, August 2023, doi: 10.1109/MCOM.004.2200473.

[9] Y. Lin et al., "A Blockchain-Based Semantic Exchange Framework for Web 3.0 Toward Participatory Economy," in IEEE Communications Magazine, vol. 61, no. 8, pp. 94–100, August 2023, doi: 10.1109/MCOM.003.2200817.

[10] W. Yang, X. Wang, Z. Guan, L. Wu, X. Du and M. Guizani, "SecureSL: A Privacy-preserving Vertical Cooperative Learning Scheme for Web

3.0," in IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2023.3332760.

[11] Y. Lin et al., "Blockchain-based Semantic Information Sharing and Pricing for Web 3.0," in IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2023.3345335.

[12] T. Wang, S. Zhang, Q. Yang and S. C. Liew, "Account Service Network: A Unified Decentralized Web 3.0 Portal with Credible Anonymity," in IEEE Network, doi: 10.1109/MNET.2023.3321090.

[13] B. Gong, C. Guo, Y. -J. Liu and Q. Wang, "Towards Secure Data Storage in Web 3.0: Ciphertext-Policy Attribute-Based Encryption," in IEEE Network, doi: 10.1109/MNET.2023.3317109.

[14] N. Kshetri, "Web 3.0 and the Metaverse Shaping Organizations' Brand and Product Strategies," in IT Professional, vol. 24, no. 2, pp. 11–15, 1 March–April 2022, doi: 10.1109/MITP.2022.3157206.

[15] D. M. Doe, J. Li, N. Dusit, Z. Gao, J. Li and Z. Han, "Promoting the Sustainability of Blockchain in Web 3.0 and the Metaverse Through Diversified Incentive Mechanism Design," in IEEE Open Journal of the Computer Society, vol. 4, pp. 171–184, 2023, doi: 10.1109/OJCS.2023.3260829.

[16] Y. Huang et al., "An Integrated Cloud-Edge-Device Adaptive Deep Learning Service for Cross-Platform Web," in IEEE Transactions on Mobile Computing, vol. 22, no. 4, pp. 1950-1967, 1 April 2023, doi: 10.1109/TMC.2021.3122279.

[17] P. Ren, L. Liu, X. Qiao and J. Chen, "Distributed Edge System Orchestration for Web-Based Mobile Augmented Reality Services," in IEEE Transactions on Services Computing, vol. 16, no. 3, pp. 1778–1792, 1 May-June 2023, doi: 10.1109/TSC.2022.3190375.

[18] Z. Xiang, Y. Zheng, Z. Zheng, S. Deng, M. Guo and S. Dustdar, "Cost-Effective Traffic Scheduling and Resource Allocation for Edge Service Provisioning," in IEEE/ACM Transactions on Networking, vol. 31, no. 6, pp. 2934–2949, Dec. 2023, doi: 10.1109/TNET.2023.3265002.

[19] Á. Santos, J. Bernardino and N. Correia, "Automated Application Deployment on Multi-Access Edge Computing: A Survey," in IEEE Access, vol. 11, pp. 89393–89408, 2023, doi: 10.1109/ACCESS.2023.3307023.

[20] B. C. Şenel, M. Mouchet, J. Cappos, T. Friedman, O. Fourmaux and R. McGeer, "Multitenant Containers as a Service (CaaS) for Clouds and Edge Clouds," in IEEE Access, vol. 11, pp. 144574–144601, 2023, doi: 10.1109/ACCESS.2023.3344486.

[21] Q. Wang, P. Wang, W. Sun and Y. Zhang, "Low-Latency Communications for Digital Twin Empowered Web 3.0," in IEEE Network, doi: 10.1109/MNET.2023.3319380.

[22] P. Consul, I. Budhiraja and D. Garg, "A Hybrid Secure Resource Allocation and Trajectory Optimization Approach for Mobile Edge Computing Using Federated Learning Based on WEB 3.0," in IEEE Transactions on Consumer Electronics, doi: 10.1109/TCE.2023.3339853.

[23] Z. Liu et al., "Secure Edge Server Placement with Non-Cooperative Game for Internet of Vehicles in Web 3.0," in IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2023.3321139.

[24] U. M. Malik, M. A. Javed, J. Frnda and J. Nedoma, "SMRETO: Stable Matching for Reliable and Efficient Task Offloading in Fog-Enabled IoT Networks," in IEEE Access, vol. 10, pp. 111579–111590, 2022, doi: 10.1109/ACCESS.2022.3215555.

[25] S. Tian, X. Deng, P. Chen, T. Pei, S. Oh, and W. Xue, A dynamic task offloading algorithm based on greedy matching in vehicle network. Ad Hoc Networks 2021, 123, 102639. doi: 10.1016/j.adhoc.2021.102639.

Biography

Mohammed Alkhathami is working as Associate Professor at Information Systems Department, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh Saudi Arabia. His research interests include communication systems, networks, security and computing.