Iraqi Journal of Information and Communications Technology(IJICT)
Vol. 8, Issue 1, April 2025
www.ijict.edu.iq
ISSN: 2222-758X
e-ISSN: 2789-7362

SOFTWARE DEFINED RESOURCE ALLOCATION IN MOBILE EDGE COMPUTING

Fatimah A. Rawdhan [1], **Emad H. Al-Hemiary** [2]
[1] Department of Computer Engineering, College of Engineering, Mustansiriyah University,Baghdad, Iraq
[2] Department of Computer Networks Engineering, College of Information Engineering, Al-Nahrain University, Jadriya, Baghdad, Iraq
fatimah.azeez@uomustansiriyah.edu.iq[1], emad@coie-nahrain.edu.iq[2]
Corresponding Author: **Emad H. Al-Hemiary**
Received:19/07/2024; Revised:23/10/2024; Accepted:17/11/2024
DOI:10.31987/ijict.8.1.294

***Abstract*- In the environment of multiple edges, an unbalanced distribution of offloaded tasks can result in a lack of edge resources, which in turn which in turn leads to lower performance. On the other hand, fast decisions regarding edge selection are crucial for efficient performance. Therefore, this paper suggests an edge-edge network based on software-defined networks to manage resources and tasks at the mobile edge of computing in the Internet of Things (IoT) environment. The proposed technique in this paper introduces an effective method for making selections concerning collaborative offloading tasks within edge computing environments based on Software-Defined Networks (SDN), which will decide where to offload and process tasks on the optimal Mobile Edge Computing (MEC) server among five MEC servers based on currently available resources when tasks need processing during a specific time using SDN controller that view the status of all network. The rank of the feasible MEC server is based on the presently available CPU frequency of the MEC server relative to the required computing resources for the task. To calculate the final height score of the MEC server, this work used Min-Max normalization and a high score for the MEC server from these servers that were considered optimal for offloading tasks. This paper aims to maintain the total latency as little as much as possible.**

keywords: **Edge network, SDN, Schedule task, Offloading algorithm, Computation offloading.**

I. INTRODUCTION

The era of 5G has brought about a remarkable surge in mobile device traffic and subscriptions. Ericsson's most recent projection spanning 2020 to 2026 anticipates a rise in worldwide mobile subscriptions between (7.9- 8.8) billion, while mobile data traffic worldwide is predicted to double [1].

The emergence of the Internet of Things (IoT) in the last few years is indeed revolutionary. IoT is a term that denotes a system of interrelated computing devices and mechanical and digital machines that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. The IoT can cause an industrial revolution, including real-time monitoring and automation decision-making that affects many industries like healthcare, transportation, and manufacturing [2]. The growing expansion of data, coupled with the inadequate computational power of mobile devices, is now a major barrier that could impact this technological advancement. On the other hand, Cloud computing still proves indispensable. But all is not lost; here comes Mobile Edge Computing (MEC) to address these very limitations and challenges [3], [4]. MEC is a novel approach that brings Cloud computing capabilities in close proximity to mobile devices within the realm of 5G networks. Edge computing is an emerging model where mobile devices can delegate their tasks to servers located at the edge of the network. It has several advantages, such as low latency, high bandwidth, and computing agility, which are vital components for real-time IoT applications. With MEC support, IoT

ISSN: 2222-758X
e-ISSN: 2789-7362

devices can cut down energy usage, ensure faster response time, and improve operational efficiency [1].

Edge computing significantly cuts down on bandwidth needs by managing data close to where it is created, minimizing the amount that needs to be sent to the centralized Cloud. The massive amount of data produced by edge-based IoT applications will only increase, necessitating efficient big data management [5]. The development of edge computing is distributed in nature, which significantly involves placing computer resources near the end users, thereby revolutionizing and making data processing more optimal. This tactical placement fulfills an essential function: reducing the possibility of transmission delays when data must travel long distances to get to typical data centers [6]. MEC stands out from the Cloud due to its proximity to information sources, lower latency, and location awareness. It facilitates big data capture, lower latency, and local service development, making it ideal for computation-intensive applications [7].

To fill this gap, this paper proposed a “Task Management Model” at the MEC architecture for IoT that aims to allocate resources in the mobile edge computing server to serve task execution faster and to receive the task by IoT devices to manage the task and the resource efficiently.

This paper is orgnaized as follows. Section II provides an overview of related studies in SDN-based edge computing for IoT applications. Section III presents the proposed model, which leverages SDN and edge computing to improve the scalability and performance of IoT systems. Section IV shows the results obtained from evaluating the performance, which has proven to be efficient in demonstrating the proposed approach. In conclusion, Section VI concludes the paper.

II. RELATED WORKS

Many ideas and methods have been proposed to deal with the difficulties of resource allocation and control in the area of MEC. However, efficiently using resources is vital for the success of executing tasks in edge computing.

The study in [8] elaborates on a number of major resource allocation strategies: Resource Sharing, Load-Balancing Algorithm, Adaptive Resource Allocation, Optimization, and Fairness-Objective Mode. These aim to enhance overall performance as well as Quality of Service (QoS) for different applications that have traffic patterns.

In [9], Sarah R. Al-Hafidh and Emad H. Al-Hemiary brought forth an innovative method in their study of a "Simplified Distributed Ledger for Task Offloading in Edge Networks". The system they suggest uses a distributed ledger to track offloaded data in edge networks resourcefully. It ensures new blocks are validated efficiently through a consensus-based voting process. Moreover, the offloading decision-making mechanism in this model does not depend on the processing time itself; it’s determined by a fixed threshold value, which keeps things interestingly unconventional and distinct from other approaches in this field. Their study demonstrated that the ledger implementation did not notably affect response times while successfully monitoring offloaded tasks.

In [10], resource allocation and energy management in smart buildings are handled by the REED system through task offloading to IoT resource-rich devices or edge nodes - a Device-to-Device (D2D) edge computing architecture that minimizes both Energy consumption and delay. The model employs the Deep Deterministic Policy Gradient (DDPG) algorithm, which generates near-optimal solutions.

In [11], the study works on collaborative service placement, task scheduling, and resource allocation schemes in edge-cloud cooperation networks. It aims to minimize task processing delay while ensuring long-term task queuing stability.

ISSN: 2222-758X
e-ISSN: 2789-7362

By transforming the optimization problem and designing iterative algorithms, the paper demonstrates superior performance compared to existing schemes through extensive simulations.

In [12], the study proposed a novel approach to scheduling - one that dynamically adapts itself to the Virtual Machines (VM) selection, determined by resource vector compatibility (RAM, CPU, Storage, Bandwidth) rather than the processor to distribute requests to VM in an efficient manner.

In [13], the study proposed task scheduling and resource allocation mechanisms in MEC for health monitoring systems, which focus on managing emergency conditions, reducing processing time and latency, and optimizing resource utilization to ensure timely and effective responses to critical situations.

In [14], the study explores offloading techniques and algorithms to minimize task execution delay, reduce energy consumption, and enhance the Quality of User Experiences (QoE) in mobile edge computing environments for optimizing computation offloading in mobile edge computing; this includes deep deterministic policy gradient and double deep Q network, reinforcement learning, and genetic algorithm.

In [15], the authors developed an adaptive offloading algorithm to load balancing and optimize task allocation in a Collaborative Edge Computing (CEC) system. They used a queueing model and dynamically adjusted offloading threshold values, demonstrating its effectiveness in minimizing mean response times and balancing loads that are created efficiently, emphasizing the importance of adaptive algorithms in CEC systems.

Drawing from the introduction and literature analysis in the previous section, it is evident that mobile IoT devices have a limited capacity for batteries, further limiting resource-intensive apps and multimedia services. The computation-offloading approach is used to lessen these restrictions and increase battery life.

III. SYSTEM MODEL

The system comprises M terminal devices, N edge computing server, and one SDN Controller (refer to Fig. 1). Each terminal device generates tasks that follow an exponential distribution. Five MEC servers in the network gather and send the task (data) to the network; these MEC servers are linked to a switch network device. The traffic from the switch to the access point is arranged by the OpenFlow switch, which is directly connected to the SDN controller. Data forwarding to the network gadgets that can be connected is managed with the aid of the controller. This technique permits network scalability since the SDN controller is flexible and configurable. Because the statistics and manipulation planes are separated, the controller has full visibility over the community and the devices that are once related to it, enabling it to govern and control the network. The parameters used in this work are defined as follows. The task T_i has four tuple, presented as $T_i = \{S_{\text{in}},\ S_{\text{out}},\ C_i,\ D_i\}$, where S_{in} and S_{out} are the input and output size of the task, C_{i} denotes its CPU cycles needed for processing, and D_{i} presents the deadline for the complete execution of the task. The computation capacity of each server can be given as $F = \{f_{\text{mec1}}, f_{\text{mec2}}, \ldots\ldots, f_{\text{mecN}}\}$.These resources can be allocated to IoT devices. When the task is offloaded, the total end-to-end delay time comprises four parts: transmission time (upload and download task), computing time, queuing time, and offloading time from the MEC server to the offloaded MEC server.

ISSN: 2222-758X
e-ISSN: 2789-7362

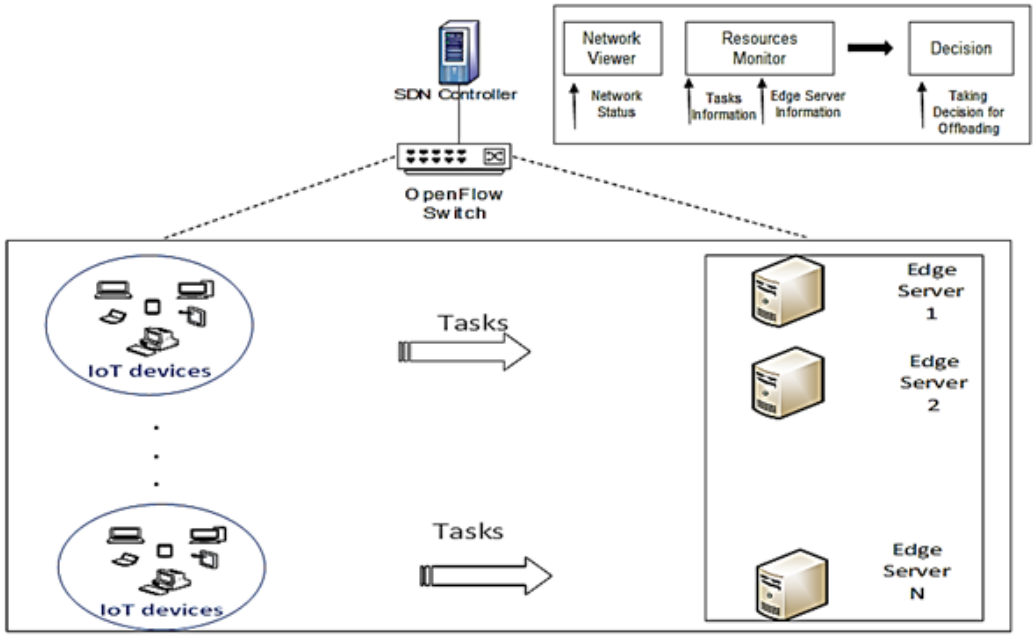

Figure 1: The system model of task scheduling and offloading in mobile edge networks with SDN controller.

A. *Delay in System Model*

1) Transmission Delay: The transmission data rate of the uplink is when tasks are sent from IoT devices to the MEC server over wireless networks. MEC server transmission time delay includs: the compute task's up-link transmission time and the task's download-link transmission time. This can be defined as below:

$$T_i^{\text{up}} = \frac{S_{\text{in}}}{R} \tag{1}$$

$$T_i^{\text{dn}} = \frac{S_{\text{out}}}{R} \tag{2}$$

where $S_{\text{in}}, S_{\text{out}}$ are the input and output size of the task as previously defined, and R is the transmission data rate for uplink and downlink.

2) Computation Delay (Processing Delay): Compute delay is an important component of mobile edge computing that greatly impacts application performance and efficiency. The computation tasks are executed at the local device or edge server computing, depending on available resources.

$$T_s^{\text{exe}} = \frac{C_s}{F_s^{\text{CPU}}} \tag{3}$$

Where C_s requested computation resources from the device and F_s^{CPU} it denoted the computation capacity of edge servers.

ISSN: 2222-758X
e-ISSN: 2789-7362

3) Waiting Delay: All other tasks can be calculated by taking the time to find the queue delay. This study assume there is a queuing delay for each task arrives at the MEC at a separate time. Thus, the waiting time can be determined by:

$$T_i^{\text{wait}} = \frac{\sum_1^m C_s}{F_s^{\text{CPU}}} \tag{4}$$

Where m is the number of tasks in the queue of edge server N.

4) End-to-end Delay: By adding the above equations, the end-to-end delay of the task can be expressed as:

$$T_i^{\text{end\_end}} = T_i^{\text{up}} + T_i^{\text{wait}} + T_s^{\text{exe}} + T_i^{\text{dn}} \tag{5}$$

Task details about the input data, workload requirements (such as CPU cycles), and delay constraints are the primary components of the task offloading requests. MEC servers update the SDN subsystem regularly with their status reports. Computational resources, task queues, and latency information are typically included in the status. A controller module can collect, manage, and evaluate this data to support decision-making. When an IoT device chooses to transfer a task, it forwards it to the controller, who determines the best time to schedule it.

B. Selection of Best MEC Server

The methods used to select the best MEC server are important to minimize the time of execution task; this study suppose multiple parameters when choosing it: CPU frequency free, minimum cost, nearby MEC server, and less task in Buffer. The method used to select the best mobile edge computing is Multi-Criteria Decision-Making (MCDM); a common method used to normalize the criteria and then combine them using a weighted sum. Algorithm 1 explains the Offloading and MEC Selection.

A particular moment's approach involves computing the ratio of available resources to resources used by each server. For the first time, the task send to the server. Afterward, each server calculates the queue waiting percentage based on its current load. These values are then processed using individual results, normalized, and combined to determine the final decision.

It is meant to rank the nodes based on their current computational resources, denoted by F_{cpu} and each network resource (latency), and it is carried out by find the Best Offloading Node (BON) method. The method initially determines the unique scores of each computational resource to apply this strategy. Specifically, the following formula is used to determine the resource's score:

$$R_k = \frac{R_{\text{ava}}}{R_{\text{req}}} \tag{6}$$

where R_{ava} is the value of available resource k in Edge nodes and R_{req} is the value of the requested resource k in the offloading task. It is crucial to remember that the resources for which smaller values are preferable must be computed in reverse. Equation (6) implies that the higher a resource's score, the better. To find the ultimate resource score of an edge node, each resource score is first standardized using min-max normalization to the same scale:

$$\eta_k = \frac{R_k - R_{\text{min}}}{R_{\text{max}} - R_{\text{min}}} \tag{7}$$

ISSN: 2222-758X
e-ISSN: 2789-7362

Algorithm 1: Offloading and MEC Selection

Input: Task set with parameters T_i, edge nodes E with information, X
Output: Total execution time
Sort all tasks using the First In First Out (FIFO) rule queue
for each MEC $n \in N$ **do**
  Calculate the score for each MEC using Eq. (6)
end for
for each task i in T **do**
  Determine task requirements and resource needs
  if $R_{\text{req}} < R_{\text{ava}}$
    $X = 0$ (MEC server 1 computing)
    Calculate $T_i^{\text{end\_end}}$ using Eq. (5)
  else
    $X = 1$
    Select the edge node with the highest score and set $X_{ij} = 1$
    Calculate $T_i^{\text{end\_end}}$ using Eq. (5)
end for

IV. SIMULATION AND RESULTS

The simulation of this work is based on the parameters shown in Table I. All simulations are carried out using a personal computer Lenovo IdeaPad 5 15IAL7, processor with 12th Gen Intel® Core™ i7-1255U × 12, 16 GB RAM, and 1 TB storage space. The simulation of the model was performed using the Python programming language. The work environment chosen was the most recent version of Pycharm. Ubuntu 22.04.4 LTS is used as an operating system.

TABLE I
Simulation Parameters

Parameter	Notation	Value
N Task	Number of tasks	(100-500)
Tn	Task input	(400, 1000) KB
To	Task output	(50, 300) KB
W	Required computing resources	(200, 1500) Megacycles
D	Deadline	(1, 5) ms
f_{mec}	CPU frequency of MEC server	6-10 GHz

Fig. 2 shows the relation between system latency and the amount of tasks. The graph shows that as the number of jobs processed increases, the overall system latency also goes up. This observation implies that workload volume and latency are directly correlated, meaning that the processing of tasks may be delayed due to a higher task load. The increased latency as tasks indicates a possible bottleneck in the task processing pipeline, where the system might find it difficult to manage the increasing workload effectively.

ISSN: 2222-758X
e-ISSN: 2789-7362

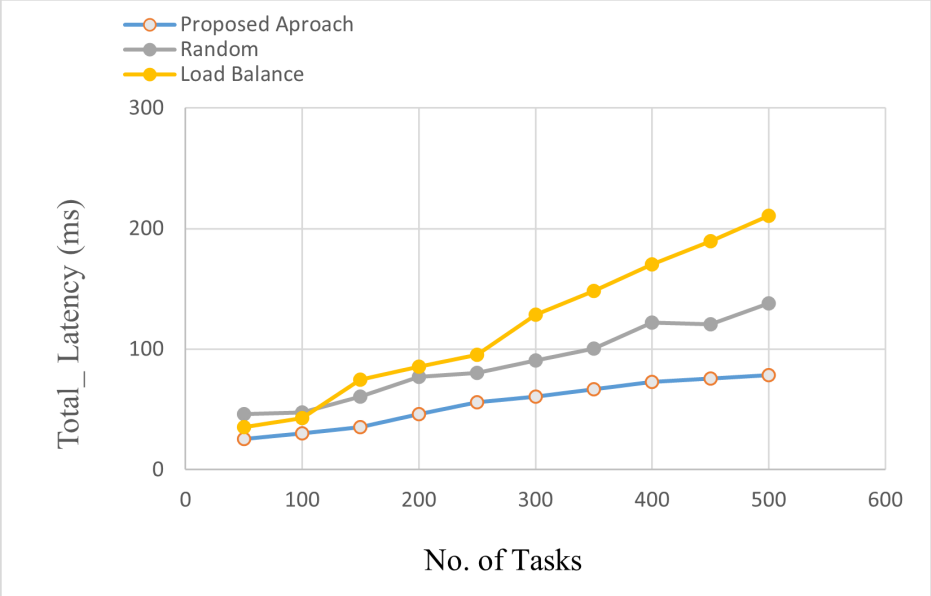

Figure 2: No. of tasks vs. latency.

Fig. 3 shows the relationship between the system's latency and processing capacity. The graph demonstrates how the system's latency reduces as processing capacity rises. This relationship highlights the crucial role of sufficient resources in enabling timely task execution inside the system by showing that more computing capacity leads to lower latency. Considering that latency and computing capability are reciprocally related, the holdups in processing tasks could be reduced by a system with adequate resources.

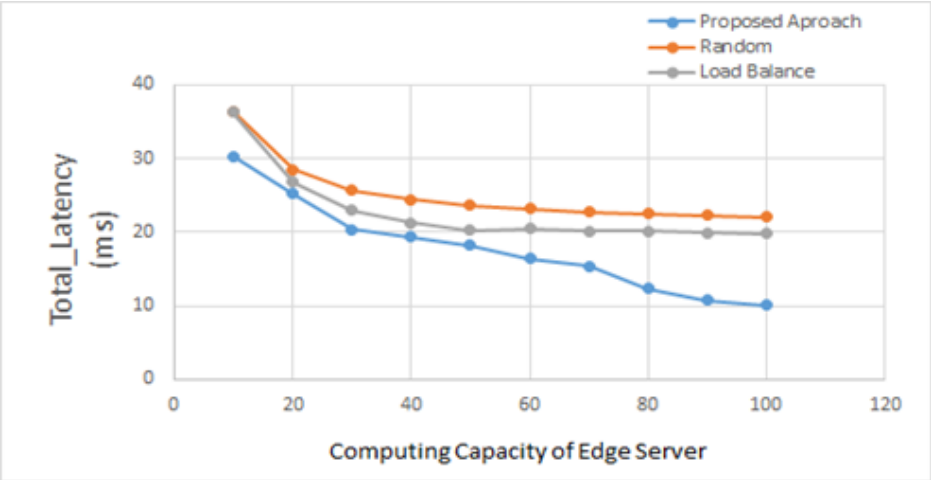

Figure 3: Computing capacity vs. latency.

ISSN: 2222-758X
e-ISSN: 2789-7362

The MEC servers have a direct impact on the latency depicted in Fig. 4. As shown in the graph, the system experiences a lower delay with an increase in the number of MEC servers. This correlation highlights how the enhancement of task processing latency is attained through the escalation of server infrastructure via additional MEC servers. The indication that latency reduces with the increment in server numbers implies that task offloading to multiple servers can be beneficial in improving effectiveness while dealing with a heavier workload.

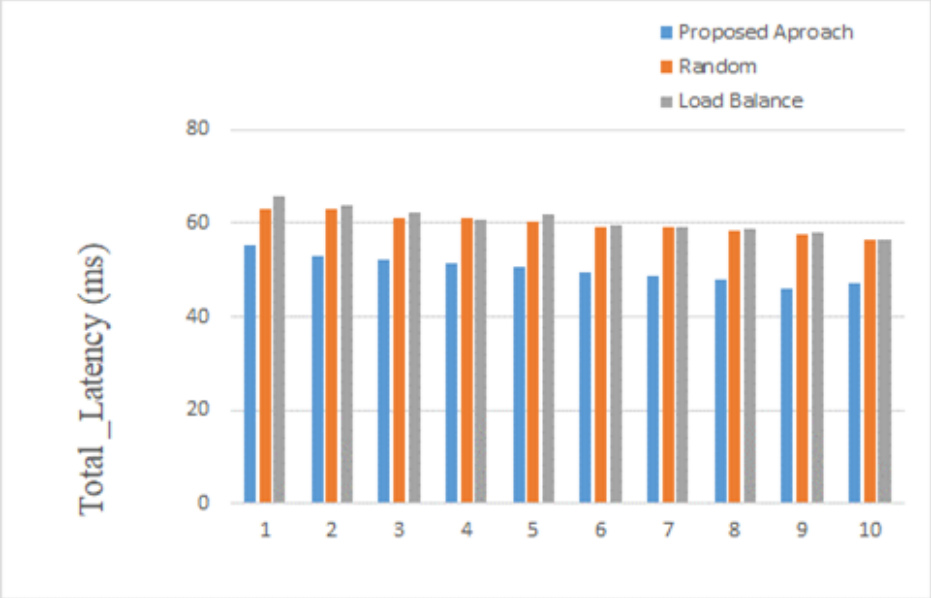

Figure 4: No. of MEC server vs. latency.

V. DISCUSSION

This paper introduces a system paradigm involving a controller, N edge computing servers, and M terminal devices. The task parameters, as defined by the model, include the size of input and output, CPU cycles for processing, and task deadlines. This work illustrates the importance of models in describing practical situations real practical situations on the one hand and then this concept of being able to study performance along with scalability of edge computing systems using these models on the other. Moreover, the discussion delves into the results of evaluating the proposed methodology's effectiveness and how policies on resource allocation plus system models are related to latency in a complex manner - including job completion times and overall system performance. Similarly, debates also entail the effects that varying processing capacities, coupled with different numbers of MEC servers as well as diverse job allocation techniques, have on system performance. The results suggest that the efficiency of a system may significantly rise, and task execution latency can be minimized when resource allocation and task scheduling methods are optimized. By being able to allocate resources from multiple edge servers smartly, the MEC system demonstrates enhanced scalability and increased performance - both

essential in meeting the needs of IoT applications. The paper further points out the importance of these findings that lead towards improving the success of an edge computing system, which more inspection and enhancement efforts in this field could later augment.

VI. CONCLUSION

The convergence of SDN with MEC: SDN improves IoT application efficiency. It does this by facilitating Mobile Service Control (MSC) in mobile edge. This proposal ensures quick task completion and proper resource management at the mobile edge through centrally controlled capabilities - coupled with real-time determination of optimal task offloading choices. The approach thus meets the critical requirements for effective large-scale IoT deployment oversight: low latency, high bandwidth, and compute agility demand by the environment. This study demonstrates how SDN-based MEC can revolutionize IoT landscapes to pave the way for more developments towards robust and flexible network architectures.

FUNDING

None.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers for their valuable contribution in the publication of this paper.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] L. A. Haibeh, M. C. E. Yagoub, and A. Jarray, "A survey on mobile edge computing infrastructure: Design, resource management, and optimization approaches," IEEE Access, vol. 10, pp. 27591–27610, 2022.
[2] O. Ali, M. K. Ishak, M. K. L. Bhatti, I. Khan, and K.-I. Kim, "A comprehensive review of internet of things: Technology stack, middlewares, and fog/edge computing interface," Sensors, vol. 22, no. 3, 2022.
[3] H. Hao, J. Zhang, and Q. Gu, "Optimal iot service offloading with uncertainty in sdn-based mobile edge computing," Mobile Networks and Applications, pp. 1–10, 2021.
[4] J. Guo, C. Li, and Y. Luo, "Resource management and switch migration in sdn-based multi-access edge computing environments," The Journal of Supercomputing, vol. 78, 09 2022.
[5] Zhang, W. E., Sheng, Q. Z., Mahmood, A., Tran, D. H., Zaib, M., Hamad, S. A., Aljubairy, A., Alhazmi, A. a. F., Sagar, S., Ma, C. (2020). The 10 Research Topics in the Internet of Things. https://doi.org/10.1109/cic50333.2020.00015.
[6] Pourrashidi Shahrbabaki, Pouria (2023) SDN-enabled Workload Offloading Schemes for IoT Video Analytics Applications. Master's thesis, Concordia University.
[7] Alghamdi, I. (2021, January 1). Computation offloading in mobile edge computing: an optimal stopping theory approach. https://theses.gla.ac.uk/82506/
[8] F. P. -C. Lin and Z. Tsai, "Hierarchical Edge-Cloud SDN Controller System With Optimal Adaptive Resource Allocation for Load-Balancing," in IEEE Systems Journal, vol. 14, no. 1, pp. 265-276, March 2020, doi: 10.1109/JSYST.2019.2894689.
[9] S. and E. Al-Hemiary, " Simplified Distributed Ledger for Task Offloading in Edge Network," Iraqi Journal of Information and Communication Technology, vol. 6, pp. 58–67, 08 2023.
[10] Ibrar, M., Erbad, A., Abegaz, M., Akbar, A., Houchati, M., Corchado, J. M. (2023). REED: Enhanced Resource Allocation and Energy Management in SDN-Enabled Edge Computing-Based Smart Buildings. https://doi.org/10.1109/iwcmc58020.2023.10182384.
[11] Fan, W., Zhao, L., Liu, X., Su, Y., Li, S., Wu, F., Liu, Y. (2024). Collaborative Service Placement, Task Scheduling, and Resource Allocation for Task Offloading with Edge-Cloud Cooperation. IEEE Transactions on Mobile Computing, 23(1), 238–256. https://doi.org/10.1109/tmc.2022.3219261.
[12] Ebadifard, F., Babamir, S. M. (2020). Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Cluster Computing, 24(2), 1075–1101. https://doi.org/10.1007/s10586-020-03177-0.
[13] Sharif, Z., Jung, L. T., Ayaz, M., Yahya, M., Pitafi, S. (2023). Priority-based task scheduling and resource allocation in edge computing for health monitoring system. Journal of King Saud University. Computer and Information Sciences/Mağalat Ğamat Al-malīk Saud: Ùlm Al-ḥasib Wa Al-malumat, 35(2), 544–559. https://doi.org/10.1016/j.jksuci.2023.01.001.
[14] Liao, L., Lai, Y., Yang, F., Zeng, W. (2023). Online computation offloading with double reinforcement learning algorithm in mobile edge computing. Journal of Parallel and Distributed Computing, 171, 28–39. https://doi.org/10.1016/j.jpdc.2022.09.006.
[15] O. Al-Tuhafi and E. Al-Hemiary, "Edge-to-cloud adaptive offloading for next-generation services," Iraqi Journal of Information and Communication Technology, vol. 6, pp. 58–67, 08 2023.

