Iraqi Journal of Information and Communications Technology(IJICT)
Vol. 6, Issue 2, August 2023

www.ijict.edu.iq

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

EDGE-TO-CLOUD ADAPTIVE OFFLOADING FOR NEXT-GENERATION SERVICES

Ola M. Al-Tuhafi [1], **Emad H. Al-Hemiary** [2]
[1,2] College of Information Engineering, Al-Nahrain University, Baghdad, Iraq
ula.mustafa@coie-nahrain.edu.iq [1], emad@coie-nahrain.edu.iq [2]
Corresponding Author: Emad H. Al-Hemiary
Received: 01/10/2022; Revised: 30/10/2022; Accepted: 06/12/2022
DOI:10.31987/ijict.6.2.230

***Abstract*- Due to the continuous growth of user traffic demands, there is a need to cope with the increased processing and storage requirements. The increased number of connected end devices causes a problem with carrying the generated loads efficiently. Edge and cloud computing and storage are real solutions to overcome the limitations of end devices on many prospective including computation capabilities, storage capabilities, and power consumption. Terminal devices offload their overflow tasks to the cloud for processing, analysis, and storage. This paper aims to improve computation offloading from edge nodes to the cloud in Internet of Things networks by making efficient decisions using an adaptive offloading algorithm. Offloading is controlled by a processing time offloading threshold value, which is determined automatically by edge nodes based on their traffic intensity and adaptively increased or decreased in loads. The proposed algorithm had been programmed and simulated; experimental evaluations show that the proposed adaptive offloading algorithm minimizes the edge mean response time by up to 58% and the cloud mean response time by up to 25% compared to the existing fixed, pre-defined offloading threshold value.**

keywords: Edge-Cloud network, computation offloading, adaptive offloading algorithm.

I. INTRODUCTION

Most Internet of Things (IoT) networks generally consist of three main layers, as shown in Fig. 1. The lowest layer is the end device layer, which contains sensors that are responsible for collecting data from the environment and actuators to perform the necessary actions. These devices send the gathered data to the upper layer for processing and storage. The middle layer is the edge layer, which may be servers, routers, access points, or any other suitable heterogeneous device. These devices are relatively limited in their processing and storage capabilities. This layer connects the end device layer with the uppermost layer, which is a cloud layer. The cloud layer is made up of several servers and data centers that are capable of super-processing, storage, and analysis [1]. The end devices usually communicate with their edge nodes through a local network, while the edge nodes communicate with the cloud through the Internet [2].

Edge and cloud computing are widely used in many applications, for example, healthcare centers, augmented reality (AR), traffic management systems, caching and processing for improved networking, and smart homes and cities [3], [4].

Offloading, on the other hand, is defined as the transfer of computations from the resource-limited node to the resource-rich node to improve application performance and power efficiency [5]. In an IoT network, offloading may happen from an end device to an edge node, from an edge node to another edge node, or from a cloud node. Offloading decisions can be either local, which are made regarding the condition of the current node, or global, which are made regarding the condition of the whole system. The global offloading decision can be further classified into active, proactive, and hybrid [6].

The main advantages of offloading are: reducing cost on the end device; increasing the end device‘s battery lifetime; supporting new applications that may require more computation and storage capabilities; improving overall performance;

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

and leading to greatercustomer satisfaction. Regarding security, offloading can increase security when user data is stored on the cloud, as this leads to less data transfer through the network; in another type of application, offloading may lead to more send and receive of data through the network, and that will decrease security [7].

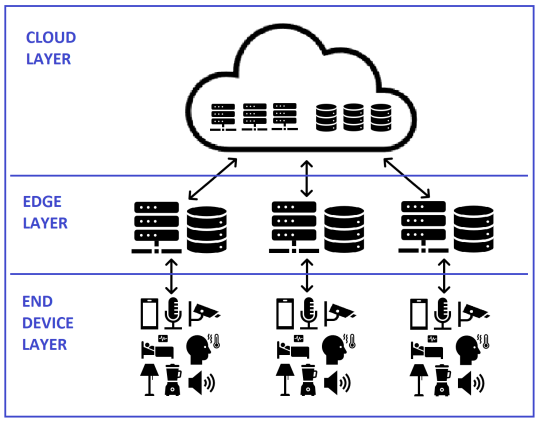

Figure 1: IoT System Architecture.

Response time is defined as the time difference between the moment when all data is collected for offloading decision-making and the moment when the consumer receives the required result [2]. The main purpose of computation offloading is to reduce the response time of tasks by reducing processing delay to improve the quality of service (QoS) [4].

This work aims to improve computation offloading from the edge to the cloud in IoT networks by making efficient offloading decisions. The goal is to minimize the mean response time of tasks and balance the load between edge nodes and the cloud. The challenge is determining which tasks should be performed at the edge and which should be offloaded to the cloud.

The rest of the paper is organized as follows: Some related studies are mentioned in Section II, the proposed model is explained in Section III, the implantation details and simulation results are in Section IV, the obtained results are discussed and compared in Section V, and finally, Section VI concludes this work.

II. LITERATURE SURVEY

Many techniques and algorithms for computation offloading in the Edge-Cloud system have been presented in recent research. Optimization methods, fuzzy logic, machine learning, and deep learning approaches, as well as model-based approaches like queueing theory and game theory, are all used for computation offloading. In [8], a multi-queue model was proposed to study the impact of two offloading policies, namely the locality-first policy and the probability-based policy. Analytical solutions for task average response time and energy consumption are obtained, and extensive simulation results proved the accuracy of the proposed model.

In [9], an online offloading policy is proposed to solve the fog node selection problem for both single-user and multi-user scenarios. Each fog node was modeled as an M/M/1/K Markov model, and the sequential decision-making problem

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

was formulated as a Restless Multi-Armed Bandit (RMAB) problem. The upper bounds on performance were calculated analytically and then validated using numerical experimental results. .

In [10], authors proposed the performance evaluation and optimization of a cloud-fog-Edge computing infrastructure using queueing theory. In particular, a Markov queueing system model is proposed for offloading considering task deadline expiration. In addition, a computational resource allocation approach is proposed and validated by comparing analytical and simulation results.

The genetic algorithm (GA) was used in [11] to find the best task allocation, while queuing theory was used to model the service time and the queuing time. The proposed offloading algorithm is compared to another optimization method, Particle Swarm Optimization (PSO). Their experiments show that the GA-based offloading algorithm performs better than both the round-robin offloading and PSO-based offloading algorithms.

n [12], the authors presented a novel approach using fuzzy logic algorithms, considering application characteristics, resource allocation, and resource heterogeneity. Their experiments show that different offloading decisions for latency-sensitive applications may lead to various service times for tasks due to communication type and computational resources.

In [13], an autonomous computation offloading framework is proposed to cope with the large dimension of the offloading decision-making problem. Different simulations have been conducted, including those using deep neural networks, multiple linear regression, the hidden Markov model, and the proposed deep learning-based hybrid model. Simulation results show that the proposed deep learning-based hybrid model suits the problem with near-optimal accuracy regarding offloading decision-making, latency, and energy consumption.

In [14], authors used the queueing theory to model a Fog-Cloud architecture to simulate the traffic delay and formulate a non-linear multi-objective optimization problem to reduce the execution delay, energy consumption, and remote execution cost. Moreover, the stochastic gradient descent (SGD) algorithm was proposed to find the optimal trade-off between the offloading objectives. Simulation results show the effectiveness of the proposed system.

In [15], the authors used the queueing theory to develop a queueing model, in which offloading decisions are made based on the required processing time for each task. Offloading is controlled by a fixed, pre-defined processing time threshold. This paper proposes an adaptive offloading algorithm that automatically chooses the perfect offloading threshold value, depending on the current load.

III. PROPOSED MODEL FOR OFFLOADING

The proposed network consists of N edge nodes covered by a single cloud node. The cloud node has m identical virtual servers. Each edge node covers a specific geographic area and generates tasks in a Poisson process exponentially distributed with an average of $1/\lambda$. Each task is generated with its own processing time that is exponentially distributed with an average of $1/\mu$. The edge node‘s CPU processes tasks at a rate of μ tasks per second, while the cloud node‘s virtual server processes tasks at a rate of $k\mu$ tasks per second, where k is the speedup factor, that is, how often the processing speed increases.

The edge node takes an offloading decision based on the required processing time of each task. If task processing time is greater than the current threshold value τ, then the task will be offloaded to the cloud for processing; otherwise, if it is

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

equal to or less than τ, it will be processed at the edge node. That leads to the fact that service time in the cloud and each edge node will have a truncated exponential distribution, i.e., a general distribution and not an exponential distribution. Thus, the queuing systems at each edge node and the cloud will be M/G/1 and M/G/m, respectively.

Offloading threshold τ divides the exponential curve of the service time into two truncated exponentials, as shown in Fig. 2.

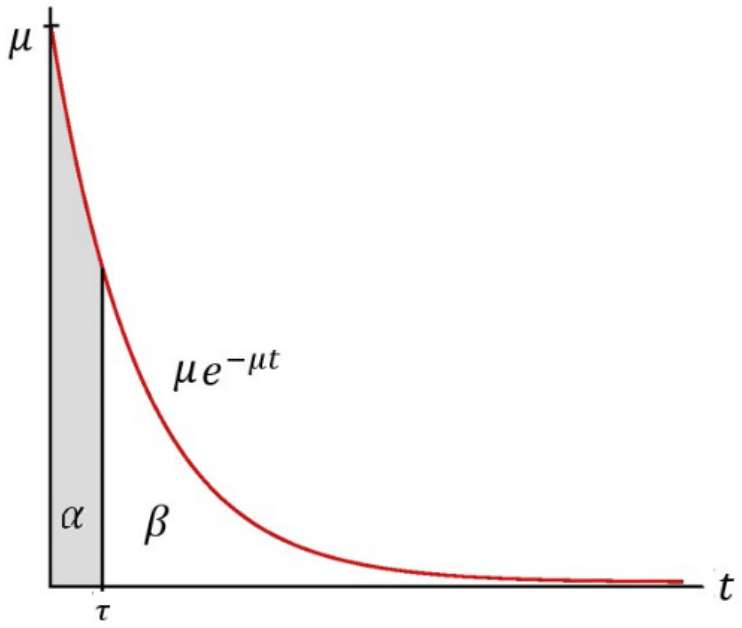

Figure 2: Offloading threshold τ divides the exponential curve of the service time into two truncated exponentials.

Let the first area under the curve be α, which is the probability that the task will be processed at the edge node. The second area under the curve will be β, which is the probability that the task will be processed at the cloud, and can be obtained:

$$\alpha = \mu \int_0^{\tau} e^{-\mu t} dt = 1 - e^{-\mu\tau} \tag{1}$$

$$\beta = \mu \int_{\tau}^{\infty} e^{-\mu t} dt = e^{-\mu\tau} \tag{2}$$

Let λ_E be the arrival rate at each edge node, and let λ_C be the arrival rate at the cloud that covers all N edge nodes. Based on the Poisson process's splitting property, λ_E and λ_C are both Poisson processes and can be obtained:

$$\lambda_E = \alpha\lambda \tag{3}$$

$$\lambda_C = N\beta\lambda \tag{4}$$

Based on the above equations, the threshold value controls the percentage of tasks offloaded or processed locally at the edge nodes. In mathematical models, the traffic intensity A can be estimated as the product of the arrival rate and the mean service time [16]. Therefore, the traffic intensity of the $\mathrm{M/G/1}$ queueing system at the edge node is:

$$A_E = \lambda_E E\left[S_E\right] \tag{5}$$

where $E[S_E]$ is the expected mean service time of the edge node; note that $E[S_E]$ is also indirectly affected and controlled by the offloading threshold value. This paper proposes a system with an adaptive offloading algorithm that allows edge nodes to continuously calculate the most suitable offloading threshold based on the current arrival rate. The selected offloading threshold guarantees that traffic intensity remains constant or within its specified acceptable range while the arrival rate increases or decreases. Fig. 3 illustrates the offloading model proposed.

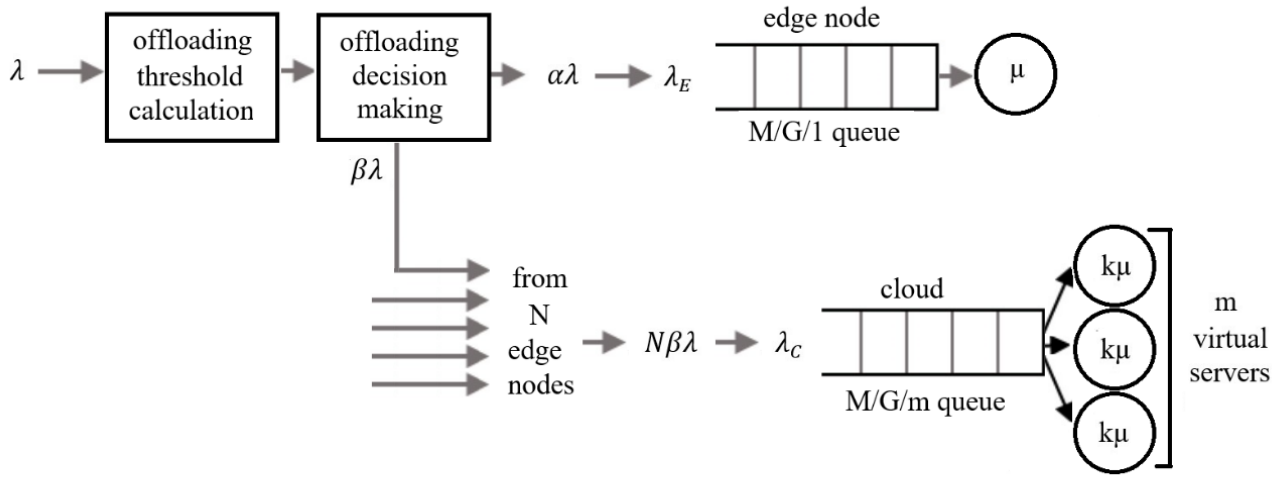

Figure 3: Offloading proposed system model.

IV. IMPLEMENTATION AND RESULTS

The proposed technique is programmed using the C++ programming language, and the work environment was the latest version of the Microsoft Integrated Development Environment (IDE), Microsoft Visual Studio 2022, Version 17.1. Several experiments were simulated to prove the effectiveness of the system. Each experimental value has been recorded and verified after a simulation run that is equivalent to more than 34 days in real time (3 million seconds) to collect enough tasks to determine an accurate mean of response times.

Experiment: **(1) Impact of increasing the arrival rate on a system with a fixed offloading threshold value (without the adaptive offloading threshold algorithm):**

To test the behavior of the system, the proposed system is simulated with a fixed offloading threshold value $\tau = 500$, while the arrival rate λ varies from 0.00001 to 0.00026 task/second. The system included $N = 800$ edge nodes; each edge node has a CPU with a processing rate of $\mu = 0.002$ task/second; all edge nodes are offloading tasks to one cloud that includes $m = 5$ virtual servers; each virtual server speeds up the processing 20 times faster than the edge node CPU, i.e., $k = 20$. The results are shown in Fig. 4, where W_E represents the edge mean response time and W_C represents the cloud mean response time. Note that both curves are rising continuously, and around 63% of tasks are offloaded to the cloud, and the rest, 37% of them, are processed at the edge node, based on (1) and (2).

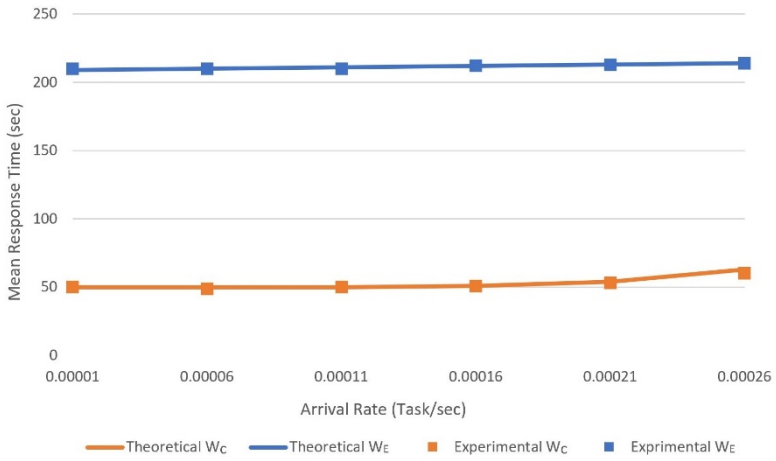

Figure 4: Results of Experiment (1).

Experiment (2): Impact of increasing the offloading threshold value on a system with a fixed arrival rate:

To evaluate the impact of changing the offloading threshold value, the proposed system is simulated with a fixed arrival rate of $\lambda = 0.001$ task/second, while the offloading threshold value varies from 100 to 1200. The system included $N = 800$ edge nodes; each edge node has a CPU with a processing rate of $\mu = 0.002$ task/second; all edge nodes are offloading tasks to one cloud that includes $m = 5$ virtual servers; each virtual server speeds up the processing 20 times faster than the edge node CPU, i.e., $k = 20$.

The result is shown in Fig. 5; note that the impact of τ is very strong on the mean response times, especially W_E. Table (1) shows the related percentage of offloaded tasks at each value, based on equations (1) and (2).

TABLE I
PERCENTAGE OF OFFLOADED TASKS FOR EACH VALUE OF EXPERIMENT (2)

τ	100	200	300	400	500	600	700	800	900	1000	1100	1200
α	18.13 %	32.97 %	45.12 %	55.07 %	63.21 %	69.88 %	75.34 %	79.81 %	83.47 %	86.47 %	88.92 %	90.93 %
β	81.87 %	67.03 %	54.88 %	44.93 %	36.79 %	30.12 %	24.66 %	20.19 %	16.53 %	13.53 %	11.08 %	9.07 %

α is the percentage of tasks processed at the edge node, and β is the percentage of tasks processed at the cloud (offloaded); for each offloading threshold value of Experiment (2).

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

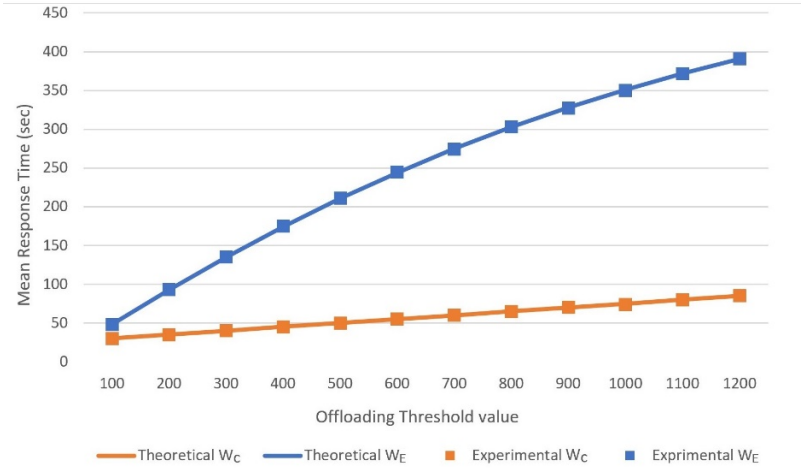

Figure 5: Results of Experiment (2).

Experiment (3): Impact of increasing the arrival rate on a system with the proposed adaptive offloading threshold algorithm:

To prove the effectiveness of the proposed adaptive offloading threshold algorithm, we simulated a system with the same values as in the experiment (1): a system with $N = 800$ edge nodes; each edge node has a CPU with a processing rate of $\mu = 0.002$ task/second; all edge nodes are offloading tasks to one cloud that includes $m = 5$ virtual servers; each virtual server speeds up the processing 20 times faster than the edge node CPU, i.e., $k = 20$; the offloading threshold is calculated automatically by the edge node depending on the current arrival rate λ, which is increased gradually from 0.00001 to 0.00026 task/second.

The traffic intensity of the edge node A_E is supposed to remain constant between 0.0060 and 0.0055 to prevent long queues in the edge node and to maintain the W_E in an acceptable range. The offloading threshold will be chosen within a specified domain with a minimum and maximum value; in this experiment, the minimum value is set to 100, the maximum value to 800, and the precision to 5.

The results in Fig. 6 show a decay in both mean response time curves, despite increasing the arrival rate. Table 2 illustrates more experimental details. Fig. 7 shows the traffic intensity A_E with adaptive offloading threshold algorithm in Experiment (3).

TABLE II
OFFLOADING THRESHOLD VALUES FOR EACH ARRIVAL RATE OF EXPERIMENT (3)

λ	0.00001	0.00003	0.00005	0.00007	0.00009	0.00011	0.00013	0.00015	0.00017	0.00019	0.00021
τ	800	645	465	370	315	280	250	230	215	200	190
α	79.81%	72.47%	60.54%	52.29%	46.74%	42.88%	39.35%	36.87%	34.95%	32.97%	31.61%
β	20.19%	27.53%	39.46%	47.71%	53.26%	57.12%	60.65%	63.13%	65.05%	67.03%	68.39%
A_E	0.00238	0.00554	0.00596	0.00594	0.00593	0.00599	0.00586	0.00587	0.00593	0.00585	0.00591

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

τ is the current offloading threshold, α is the percentage of tasks processed at the edge node, β is the percentage of tasks processed at the cloud (offloaded), and A_E is the traffic intensity of edge node; for each arrival rate of Experiment (3).

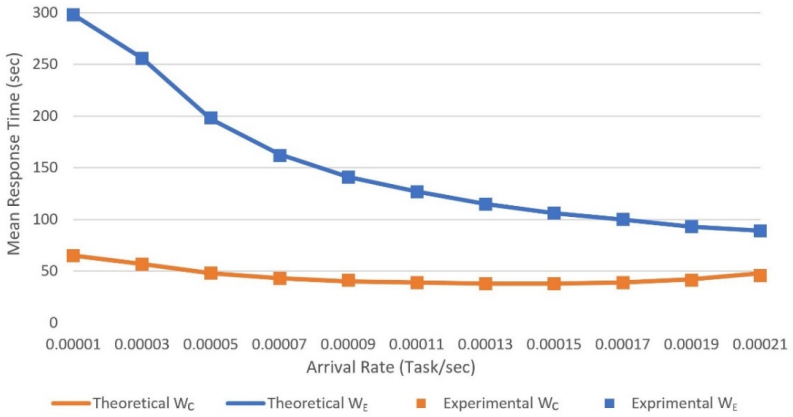

Figure 6: Results of Experiment (3).

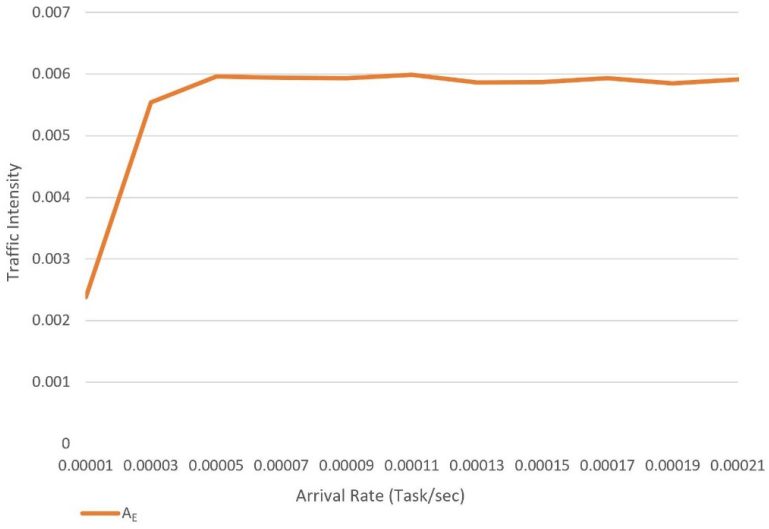

Figure 7: Traffic intensity of the edge node with adaptive offloading threshold in Experiment (3).

V. DISCUSSION

In the results of Experiment (1), Fig. 4, note that both W_E and W_C curves are almost horizontal or slightly rising, offloading threshold is fixed regardless of the arrival rate, and around 63% of tasks are offloaded to the cloud, while the rest, 37% of tasks, are processed at the edge node.

To deal with the limited processing and storage capabilities of the edge nodes, the proposed adaptive offloading algorithm controls the mean response time by maintaining the traffic intensity of the edge node almost constant (within the pre-specified range; i.e., A_E never exceeds 0.0060), as shown in Fig. 7. This prevents long queues at the edge node and maintains the W_E at an acceptable low value.

In the results of Experiment (3), Figure 6, where the proposed algorithm is used, the W_E curve starts at a high value, around 298 seconds, because τ is high; then it decays because the edge node starts to decrease τ, so it has more tasks that require less processing time, i.e., more short tasks. Concerning the W_C curve, it also starts at around 65 seconds because τ is high, so the cloud processes only the time-consuming tasks, i.e., a few long tasks; then it decays because it has more tasks but with shorter required processing times. At the end, the W_C curve raises slightly; because τ becomes very low, and the cloud handled most of the load.

From the results above, the proposed adaptive offloading algorithm in the third experiment, at $\lambda = 0.00021$, minimizes the W_E by up to 58% and the W_C by up to 25% compared to the results of the first experiment with a fixed offloading threshold value.

VI. CONCLUSIONS

The proposed model is simulated with an adaptive offloading algorithm that allows edge nodes to continuously determine the most suitable offloading threshold based on the current arrival rate. The traffic intensity value is supposed to remain constant, or within its specified acceptable range, while the arrival rate increases or decreases. Experimental evaluations show that the proposed adaptive offloading algorithm minimizes the edge mean response time by up to 58% and the cloud mean response time by up to 25% compared to the existing fixed, pre-defined offloading threshold value.

Funding

None

ACKNOLEDGEMENT

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

CONFLICTS OF INTEREST

The author declares no conflict of interest.

REFERENCES

[1] M. Sheikh Sofla, M. Haghi Kashani, E. Mahdipour, and R. Faghih Mirzaee, "Towards effective offloading mechanisms in fog computing," *Multimed Tools Appl*, vol. 81, no. 2, pp. 1997–2042, Jan. 2022, doi: 10.1007/s11042-021-11423-9.
[2] S. Shahhosseini et al., "Exploring computation offloading in IoT systems," *Inf Syst*, 2021, doi: 10.1016/j.is.2021.101860.
[3] K. Gasmi, S. Dilek, S. Tosun, and S. Ozdemir, "A survey on computation offloading and service placement in fog computing-based IoT," *Journal of Supercomputing*, vol. 78, no. 2, pp. 1983–2014, Feb. 2022, doi: 10.1007/s11227-021-03941-y.
[4] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward Computation Offloading in Edge Computing: A Survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019, doi: 10.1109/ACCESS.2019.2938660.

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

[5] J. Almutairi and M. Aldossary, "Modeling and analyzing offloading strategies of IoT applications over edge computing and joint clouds," *Symmetry (Basel)*, vol. 13, no. 3, pp. 1–19, Mar. 2021, doi: 10.3390/sym13030402.
[6] A. Shakarami, M. Ghobaei-Arani, M. Masdari, and M. Hosseinzadeh, "A Survey on the Computation Offloading Approaches in Mobile Edge/Cloud Computing Environment: A Stochastic-based Perspective," *Journal of Grid Computing*, vol. 18, no. 4, pp. 639–671, Dec. 01, 2020, doi: 10.1007/s10723-020-09530-2.
[7] M. Masdari and H. Khezri, "Efficient offloading schemes using Markovian models: a literature review," Computing, vol. 102, no. 7, pp. 1673–1716, Jul. 2020, doi: 10.1007/s00607-020-00812-x.
[8] Y. Zhang, B. Feng, W. Quan, G. Li, H. Zhou, and H. Zhang, "Theoretical analysis on edge computation offloading policies for IoT devices," IEEE Internet Things J, vol. 6, no. 3, pp. 4228–4241, Jun. 2019, doi: 10.1109/JIOT.2018.2875599.
[9] M. Yang, H. Zhu, H. Wang, Y. Koucheryavy, K. Samouylov, and H. Qian, "An Online Learning Approach to Computation Offloading in Dynamic Fog Networks," IEEE Internet Things J, vol. 8, no. 3, pp. 1572–1584, Feb. 2021, doi: 10.1109/JIOT.2020.3015522.
[10] R. Fantacci and B. Picano, "Performance Analysis of a Delay Constrained Data Offloading Scheme in an Integrated Cloud-Fog-Edge Computing System," IEEE Trans Veh Technol, vol. , no. 10, pp. 12004–12014 Oct. 2020, doi 10.1109/TVT.2020.3008926.
[11] S. A. Zakary, S. A. Ahmed, and M. K. Hussein, Evolution offloading in edge environment," Egyptian Informatics Journal, vol. 22, no. 3, pp. 257–267, Sep. 2021, doi: 10.1016/j.eij.2020.09.003.
[12] J. Almutairi and M. Aldossary, "A novel approach for IoT tasks offloading in edge-cloud environments," Journal of Cloud Computing, vol. 10, no. 1, Dec. 2021, doi: 10.1186/s13677-021-00243-9.
[13] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach," Journal of Network and Computer Applications, vol. 178, Mar. 2021, doi: 10.1016/j.jnca.2021.102974.
[14] F. Sufyan and A. Banerjee, "Computation Offloading for Smart Devices in Fog-Cloud Queuing System," IETE J Res, 2021, doi: 10.1080/03772063.2020.1870876.
[15] A. S. Ibrahim, H. Al-Mahdi, and H. Nassar, "Characterization of task response time in a fog-enabled IoT network using queueing models with general service times," Journal of King Saud University - Computer and Information Sciences, 2021, doi: 10.1016/j.jksuci.2021.09.008.
[16] V.I.-T.U.of Denmark and undefined 2010, "Teletraffic engineering and network planning," unina.stidue.net, 2009.

