2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA) | 979-8-3503-7611-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICSINTESA62455.2024.10747972

INTEGRATING BLOCKCHAIN WITH IoT-EDGE-CLOUD NETWOTK FOR TRACKING OFFLOADED TASKS

Sarah R. Al-Hafidh
College of Information Engineering
Al-Nahrain University
Baghdad, Iraq
0009-0009-3950-2044

Emad H. Al-Hemiary
College of Information Engineering
Al-Nahrain University
Baghdad, Iraq
0000-0002-1564-0479

***Abstract*— Edge and cloud computing offer cost-effective computational options, including free services, for IoT devices with limited resources. Offloading refers to the practice of transferring tasks to the edge and/or cloud systems. Meanwhile, blockchain technology is increasingly used in various applications for its ability to decentralize decision-making and ensure data integrity and traceability in a transparent manner, eliminating the need for intermediaries. However, integrating edge and cloud computing with blockchain presents challenges. Blockchain integration can increase processing times, add complexity, and require additional storage for the blocks it generates. This paper proposes a network that combines edge and cloud servers to handle task offloading with different execution times from various devices, such as IoT, mobile, and other computing devices. To ensure the integrity of these offloaded tasks, the network incorporates blockchain technology, using a simplified, voting-based consensus algorithm for block validation. The theoretical framework is based on queueing models, particularly those with general service time distribution and single or multiple server setups. Experimental results, drawn from tests with both local and remote clouds, indicate that integrating blockchain only slightly affects overall response times. However, implementing blockchain requires additional storage, which poses a challenge for devices with limited resources.**

Keywords—, blockchain, edge-cloud computing, response time, tasks offloading, voting.

I. INTRODUCTION

Computation offloading refers to the process of transferring resource-intensive computational tasks to another computation resource via fast links. This is done to enhance performance and overcome the limitations of the terminal device [1]. The primary objective is to efficiently utilize all system resources for processing all generated tasks effectively [2]. Task offloading involves deciding which tasks should be offloaded to the cloud, offloaded to the edge node, or executed locally on the device (non-offloaded tasks) [3]. In some situations, the task can be partitioned, and the computation will be split between them [4]. Hence, the decision-making process for offloading involves three situations: first, local execution, where no offloading occurs and the task is entirely processed on the device; second, full offloading, where the entire task is transferred to a remote node for processing; and third, partial offloading, where only a specific portion of the task is offloaded to a remote node while the rest is processed locally [5]. In addition, offloading can occur in various scenarios: from a terminal device to an edge node, between one edge node and another, or from a cloud node [6]. On the other hand, blockchain is introduced as a data structure that uses hash functions and asymmetric encryption methods to prevent data fraud and manipulation. The most common hash function used in blockchain is SHA256, which is defined as a type of Secure Hash Algorithm (SHA), and it produces an output of 256 bits [7]. To integrate blockchain with the IoT-edge-cloud (IEC) network, it is necessary to define the core of the blockchain, which is the consensus [9]. The consensus is the validation of every newly created block by all participating nodes; once validated, the block is appended to the end of the blockchain [8]. Popular consensus mechanisms like Proof of Work (PoW) and Proof of Stake (PoS) have their own distinct benefits. However, they also come with certain disadvantages, especially in the context of computational offloading aimed at decreasing response times and providing storage solutions [9].

In this paper, a network of three layers—the terminal device (TD) layer, the edge layer, and the cloud layer—integrated with blockchain technology is designed and implemented. The aim is to simplify the consensus of the blockchain to reduce the computational overhead of the blockchain and work fluently with the ICE network while achieving data integrity, task tracking, and maintaining a loud balance among the network entities.

II. RELATED WORK

In [10] , a model of three layers was proposed: the first layer is an IoT network layer, the middle layer is a blockchain edge layer, and the last layer is a blockchain network layer. This model provides secure authentication, data sharing among IoT programs, and the ability to trace terminal devices. (PBFT) consensus and smart contracts are used in blockchain. The results show the effectiveness of the model.

In [11], the authors introduced the BeCome method, a strategy that uses blockchain technology for computation offloading. This approach aims to achieve multiple goals. It primarily concentrates on minimizing the time and energy required for offloading tasks to edge computing devices. Additionally, it focuses on ensuring both load balancing and the integrity of data. The effectiveness of BeCome has been proven experimentally.

In [12], the authors developed a fog computing network (FCN) combined with blockchain technology to reduce network utility. This reduction encompasses both the energy usage of the FCN and the latency involved in the blockchain network. The architecture of this network is structured into

two tiers: a device layer and a fog layer, with blockchain technology implemented within the fog layer. The outcomes of the simulations indicate that the algorithm they proposed performs effectively and is practically viable.

In [13], a fog network with three virtualization modes is proposed for characterizing the mean response time of the offloaded tasks using a queuing model. The first virtualization mode assumed that the service time was constant; the second assumed a uniform distribution service time; and the third assumed a truncated exponential service time. The results reveal that the response time can be improved when the virtualization is managed appropriately.

In [14], The researchers developed a reputation and voting based consensus mechanism (RVC), designed to decrease the duration of the consensus process. Additionally, they used a filtering algorithm aimed to identify and eliminate malicious nodes. The experiments conducted demonstrate that this new algorithm exceeds various existing consensus algorithms in both time efficiency and consensus rate.

In [15], the authors proposed an adaptive offloading algorithm and queueing model to improve the performance of task offloading by making optimal offloading decisions. The offloading processing time threshold value, on which the offloading decisions are based, is adjusted dynamically to get the optimum value of the threshold that maintains the load balance on the end devices and edge nodes along with the increment and decrement of loads. Experimental results show that the proposed method was efficient in minimizing task average response time and maintaining load balance.

Subsequently, some of the above work considered the integration of blockchain with offloading networks. The main difference between this work and the above is the way of validating the newly created block in the blockchain and how it is combined with task offloading.

III. NETWORK MODEL

The proposed network consists of M TD in the TD layer; each TD generates tasks randomly with an average time of $1/\lambda$, with its processing time (T_p) following an exponential service time μ. If the $T_p < \tau_1$ the tasks are processed in TD with a service rate μ following the M/G/1 queue model. Otherwise, the TD will calculate the hash function SHA256 for the task, record the time of creating the task, the time stamp (TS), record its ID, and record the hash of the previous task it generates. If it is the first task, then the previous hash is zero. Then encapsulate this information in the block; after that, this block will be offloaded to the edge layer. In the edge layer, there are N edge nodes; each of them has n_1 identical virtual servers. All servers will receive the block, and if the $\tau_1 > T_p < \tau_2$, the servers will recalculate the hash function of the received task and compare it with the received one. If they are equal, the edge server will vote (Yes); otherwise, it will vote (No). After voting, the edge servers will share their votes among them to reach a consensus and make the decision of processing the task or discarding it. If the accumulative number of (yes) > $(N/2+1)$, the task will be processed with service rate $k_1\mu$ following the M/G/m queue model, and the block will be added to the ledger that is stored on all edge servers by the appropriate edge server following the round robin (RR) method. If the accumulative number of (yes) < $(N/2+1)$, the task will be discarded and the TD will be informed. If the $T_p > \tau_2$, the tasks will be offloaded to the cloud layer. The cloud layer consists of one cloud with n_2 identical virtual servers. All servers will receive the offloaded tasks and check the received hash with the recalculated one, vote on it with (Yes) or (No), share their votes with one another to reach consensus, and then make a decision about processing the task with service rate $k_2\mu$ following the M/G/m queue model, and add the block to the ledger that is stored on all cloud virtual servers or discarding it and inform the TD. Fig. 1 represents the state transition diagram of this network.

The voting and RR methods have been used to validate newly generated blocks and add them to the ledger to ensure that applying blockchain will not add overhead to the network or increase complexity. Furthermore, using the proposed method will not require extra computation and therefore extra time, as there is no puzzle that need to be solved for validating new blocks. Which means that the proposed consensus will be compatible with the offloading concept of maintaining the resources of the devices by consuming them on consensus. In addition, the proposed consensus doesn't require any third party as it is distributed and decentralized. Fig. 2 shows the block diagram of the consensus-based voting process. It is worth noting that only the header of the block will be stored in the ledger to save storage space. The aforementioned k_1 and k_2 specify how much the edge and cloud are faster at processing tasks than TD.

To calculate the average arrival rate, let λ_{TD}, λ_{Ed}, and λ_{Cl} be the respective rates at which tasks reach the TD CPU, the edge node, and the cloud. These arrival rates are characterized by Poisson processes and can be defined accordingly [15]:

$$\lambda_{TD} = \alpha\lambda \tag{1}$$

$$\lambda_{Ed} = M\beta\lambda \tag{2}$$

$$\lambda_{Cl} = NM\gamma\lambda \tag{3}$$

The symbols γ, β, and α represent the likelihood of processing tasks at the cloud, edge node, and TD, respectively. Additionally, the probability density function (PDF) for the exponential service time is presented in the following manner [16]:

$$f_S(t) = \mu e^{-\mu t} \quad , \qquad t > 0 \tag{4}$$

let the continuous random variable of the service time symbolized as S. The previously mentioned probabilities are derived from this distribution as shown [14]:

$$\alpha = \text{Prop}(S < \tau_1) = \int_0^{\tau_1} \mu e^{-\mu t}\, dt = 1 - e^{-\mu\tau_1} \tag{5}$$

$$\beta = \text{Prop}(\tau_1 \le S < \tau_2) = \int_{\tau_1}^{\tau_2} \mu e^{-\mu t}\, dt = e^{-\mu\tau_1} - e^{-\mu\tau_2} \tag{6}$$

$$\gamma = \text{Prop}(S \ge \tau_2) = \int_{\tau_2}^{\infty} \mu e^{-\mu t}\, dt = e^{-\mu\tau_2} \tag{7}$$

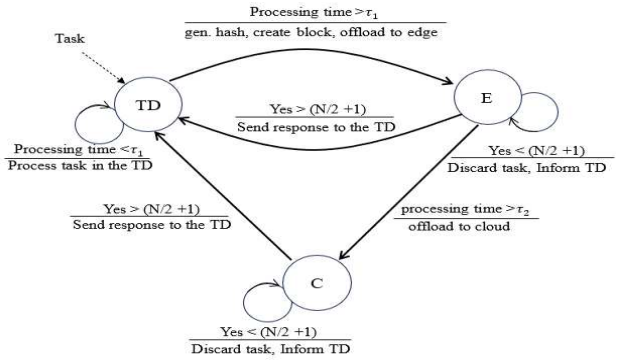

Fig 1: The state transition diagram of the proposed network

To compute the average response time, the consensus time in both edge and cloud environments must be considered, as well as the system time, which includes both processing and queuing times. In the case of the TD, there is no consensus time, only the system time.

A. Consensus Time

Let T_c represents the consensus time, then T_c is the time required for regenerating the task hash, checking it with the received one, voting, and sharing votes among the nodes in the blockchain network. It is worth noting that the consensus time was measured experimentally.

B. Theoretical Average Reponse Time

The average response time of the TD which is denoted as W_{TD} determined by applying the Pollaczek–Khinchine mean value formula, which is specific to an M/G/1 queue's response time [17].

$$W_{TD}=E[S_{TD}]+\frac{\lambda_{TD}E[S_{TD}^2]}{2(1-\lambda_{TD}E[S_{TD}])} \quad (8)$$

Where $E[S_{TD}]$ is the 1st moment of the distribution and it is the TD's average service time. By applying Palm's Identity, $E[S_{TD}]$ is calculated as follows [18]:

$$E[S_{TD}]=\frac{1-(1+\mu\tau_1)e^{-\mu\tau_1}}{(1-e^{-\mu\tau_1})\mu} \quad (9)$$

While $E[S_{TD}^2]$ is the 2nd moment and calculated as:

$$E[S_{TD}^2]=\frac{2-(2+2\mu\tau_1+\mu^2\tau_1^2)e^{-\mu\tau_1}}{(1-e^{-\mu\tau_1})\mu^2} \quad (10)$$

The average response time of the edge node which is denoted as W_{Ed} determined by applying the Pollaczek Khinchine mean value formula, which is specific to an M/G/m queue's response time [19].

$$W_{Ed}\approx E[S_{Ed}]+\frac{(\lambda_{Ed})^{n_1}E[S_{Ed}^2](E[S_{Ed}])^{n_1-1}}{2(n_1-1)!(n_1-\lambda_{Ed}[S_{Ed}])^2 G_{Ed}}+T_C \quad (11)$$

Where G_{Ed} is calculated as follows:

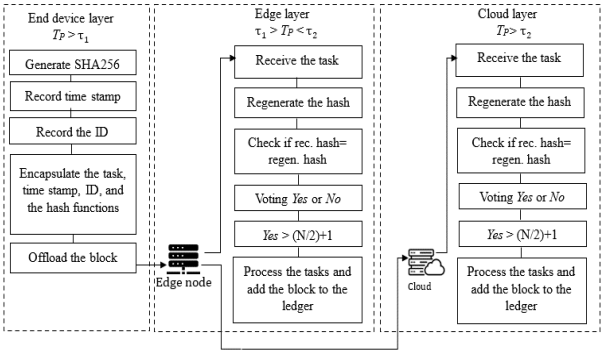

Fig 2: The block diagram of blockchain consensus based-voting

$$G_{Ed}=\sum_{i=0}^{n_1-1}\frac{(\lambda_{Ed}E[S_{Ed}])^i}{i!}+\frac{(\lambda_{Ed}E[S_{Ed}])^{n_1}}{(n_1-1)!\,(n_1-\lambda_{Ed}E[S_{Ed}])}$$

and $E[S_{Ed}]$ is the 1st moment of the edge mean service time and calculated by applying Palm's Identity [15]:

$$E[S_{Ed}]=\frac{(1+\mu\tau_1)e^{-\mu\tau_1}-(1+\mu\tau_2)e^{-\mu\tau_2}}{(e^{-\mu\tau_1}-e^{-\mu\tau_2})\,k_1\mu} \quad (12)$$

Similarly, the 2nd moment $E[S_{Ed}^2]$ of the edge node's service time is:

$$E[S_{Ed}^2]=\frac{(2+2\mu\tau_1+\mu^2\tau_1^2)e^{-\mu\tau_1}-(2+2\mu\tau_2+\mu^2\tau_2^2)e^{-\mu\tau_2}}{(e^{-\mu\tau_1}-e^{-\mu\tau_2})\,k_1^2\mu^2} \quad (13)$$

In a similar way, the average response time of the cloud which is denoted as W_{Cl} determined by applying the Pollaczek Khinchine mean value formula, which is specific to an M/G/m queue's response time [19].

$$W_{Cl}\approx E[S_{cl}]+\frac{(\lambda_{Cl})^{n_2}E[S_{Cl}^2](E[S_{Cl}])^{n_2-1}}{2(n_2-1)!(n_2-\lambda_{Cl}[S_{Cl}])^2 G_{Cl}}+T_c \quad (14)$$

Where G_{cl} is calculated as follows:

$$G_{Cl}=\sum_{i=0}^{n_2-1}\frac{(\lambda_{Cl}E[S_{Cl}])^i}{i!}+\frac{(\lambda_{Cl}E[S_{Cl}])^{n_2}}{(n_2-1)!\,(n_2-\lambda_{Cl}E[S_{Cl}])}$$

and $E[S_{Cl}]$ is the 1st moment of the cloud mean service time and calculated by applying Palm's Identity [18]:

$$E[S_{Cl}]\frac{1+\mu\tau_2}{k_2\mu} \quad (15)$$

Similarly, the 2nd moment $E[S_{CL}^2]$ of the cloud's service time, is:

$$E[S_{Cl}^2]=\frac{2+2\mu\tau_2+\mu^2\tau_2^2}{k_2^2\,\mu^2} \quad (16)$$

IV. IMPLEMENTATION AND RESULTS

Several experiments were implemented using the C++ programming language within the latest Microsoft Visual

Studio 2022, Version 17.7 (Integrated Development Environment). To demonstrate the efficiency of the proposed network, these experiments have been repeated several times. All experiment results were carefully recorded and verified after a simulation period equivalent to 3000000 seconds in real-time (approximately thirty-five days) to ensure a precise measurement of the average response times for all values.

Table I presents the paramters used in the first and second experiments of the proposed network. Where μ is the service rate, M = the number of TDs, N = the number of edge servers; τ_1 is threshold 1 for offloading to the edge, τ_2 is threshold 2 is for offloading to the cloud; these thresholds have been set after many experiments to maintain the resource management. K_1 and K_2 are the speed factors of the edge and cloud, respectively; these speed factors indicate how much the edge and cloud are faster than the TD. The n_1 and n_2 are the number of virtual servers on edge and cloud, respectively.

Experiment 1: The network was implemented on a local laptop with an operating system of Windows 11 Home 64-bit (10.0, Build 22621); the system model is an Inspiron 7306; the BIOS is 1.24; the processor is an 11th Gen Intel(R) core (TM) i7-1165G7 @ 2.80GHz (8CPU), ~ 1.7GHz; and the memory is 16MB of RAM. In this experiment, the arrival rate (λ) of the tasks is gradually increased from (0.00001 to 0.0001) tasks per second. The TD processes tasks at a service rate of μ task per second, and if the processing time of the task is greater than τ_1, the TD will offload the task to the corresponding edge, and it will process the task at a rate of K_1 μ task per second. If the processing time is greater than τ_2, the edge will offload the task to the cloud, and the cloud will process the task at a rate of K_2 μ task per second. It is worth noting that the transimission time and propagation delay have been neglected, as they were small compared to the processing time. The result in Fig. 3 shows the theoretical and experimental average response time in TD, edge servers, and cloud, while the result in Fig. 4 show the delay time of the consensus and it is observed that the delay is relatively small and will affect the average response time slightly. On Fig. 5 the total number of executed tasks is presented.

Experiment 2: In this experiment, the TD and edge layers were implemented on a local laptop, while the cloud layer was implemented on a remote cloud located in New Jersey, United States of America. The remote cloud was a virtual private server (VPS), and the operating system was Windows Server 2022 standard 64-bit (10.0, build 20348). The system model is a virtual machine. Processor is intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz (5CPU) ~ 2.4GHz. Memory is 10240 MB of RAM. The arrival rate (λ) of the tasks is gradually increased from (0.00001-0.00019) tasks per second. The TD processes tasks at a service rate of μ task per second, and if the processing time of the task is greater than τ_1, the TD will offload the task to the corresponding edge, and it will process the task at a rate of K_1 μ task per second. If the processing time is greater than τ_2 the edge will offload the task to the remote cloud,

TABLE I. THE PARAMETERS OF EXPERIMINT THREE

μ	λ	M	N	τ_1	τ_2	K_1	K_2	n_2
10^{-3}	10^{-5}	400	20	400	2500	20	100	30

and it will process the task at a rate of K_2 μ task per second. The transmission time has been measured practically. Fig 6 shows the consensus time of the edge, and it is slightly decreased compared to experiment one due to

freeing up the CPU from processing cloud tasks. Fig 7 shows the consensus time of the remote cloud and it has been increased due to the effect of the transimission time. Fig 8 presents the total number of executed tasks in TD, edge, and cloud.

Table II presents the parameters used in the third experiment of the proposed network. Where μ is the service rate, λ is the arrival rate, M = the number of TDs, N = number of edge servers, τ_1 is threshold one for offloading to edge, τ_2 is threshold 2 for offloading to cloud, these thresholds have been set after many experiments to maintain the resource management. K_1 and K_2 are the speed factors of edge and cloud, respectively; these speed factors indicate how much the edge and cloud are faster than the TD, and n_2 is the number of vertual servers in edge.

Experiment 3: The result of this experiment compared to the result obtained from [14]. The parameters of this experiment are shown in Table II. The number of virtual servers of edge nodes (n_1) has been increased from 50 to 300 edge nodes. The TD will processes the tasks at a service rate of μ task per second. If the task processing time exceeds τ_1, the TD will offload the task to the corresponding edge. The edge will process the task at a service rate of k_1 μ task per second. If the processing time of the task exceeds τ_2, the edge will offload the task to the cloud, which in turn will process the task at a rate of k_2 μ task per second. To make a fair comparison with [14], the blockchain has been applied only to the edge layer. This experiment has been implemented on a local laptop. In addition, the transmission time and the propagation delay have been neglected, as they were small compared to the processing time. Table III shows that [14] consensus time started stable and then began to increase, while the proposed voting consensus time has approximately linear relationship with the number of edge nodes. The number of generated blocks in [14] is much less than the proposed network.

TABLE II. THE PARAMETERS OF EXPERIMINT THREE

μ	λ	M	N	τ_1	τ_2	K_1	K_2	n_2
10^{-3}	10^{-5}	400	20	400	2500	20	100	30

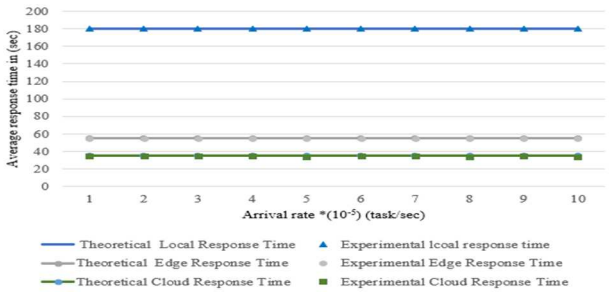

Fig 3 the theoretical and experimental average response time of TD, edge, and cloud

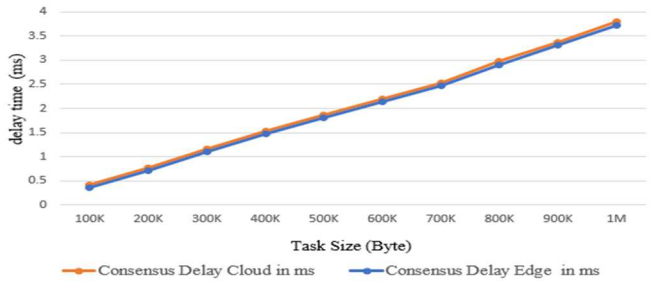

Fig 4: The consensus time in edge and cloud

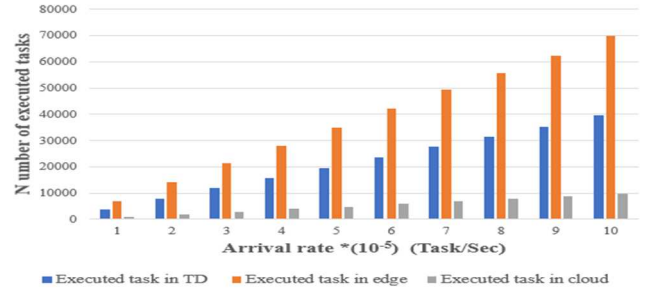

Fig 5: The total numer of executed tasks in TD, edge, and cloud

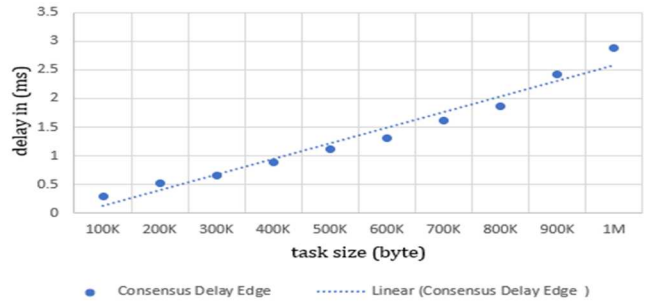

Fig 6: The consensus time in edge when using remote cloud

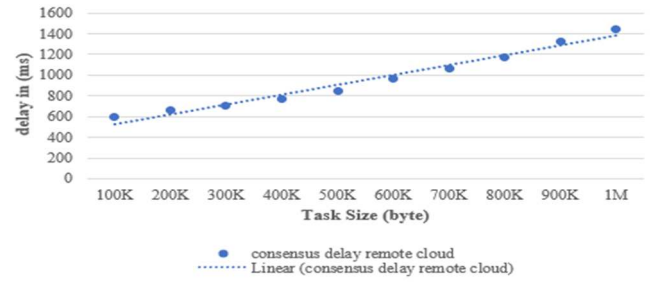

Fig 7: The consensus time in the remote cloud

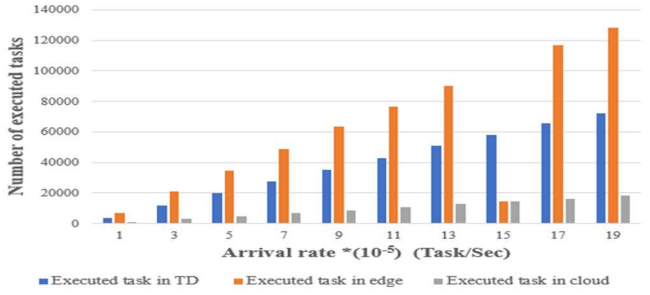

Fig 8: The total number of executed tasks in TD, edge, and the remote cloud

TABLE III. THE RESULTS OF THE COMPASSION BETWEEN [14] AND THE PROPOSED NETWORK

Number of blocks in [14]	Number of blocks	Number of edge nodes	Consensus time (ms) in [14]	Consensus time (ms)
100	6991	50	14	4
100	6991	100	14	6
100	6991	150	15	9
100	6991	200	16	13
100	6991	250	18	16
100	6991	300	22	19

V. CONCLUSION

This paper proposes a network of edge and cloud task offloading integrated with blockchain. The blockchain is used to track the offloaded task and maintain its integrity. The consensus of the blockchain is based on voting, while the selection of the edge and/or cloud to add the block is based on the Round Robin method. The experimental results showed that the offloaded tasks in the proposed network have been tracked and maintained integrity successfully, and the average response time has been affected slightly due to the simplified consensus proposed. Although enabling blockchain keeps tracking and integrity, there is a mandatory need for extra storage of the ledger, which may become an issue for limited resources. In addition, the experimental results showed that using the remote cloud, located in New Jersey, was efficient as it was possible to increase the number of executed tasks and decrease the consensus time in edge. Due to the fact that the remote experiments have been conducted using a shared internet link of approximately 6 Mbps, the overall cloud consensus time has increased (transmission time effects the overall end-to-end delay). Simplifying the consensus of the blockchain required removing encryption of the whole block, which is considered a counter measure to keep privacy. In the future, we will apply encryption to the proposed blockchain to increase security and measure the impact of using voting consensus on energy consumption.

REFERENCES

[1] S. Shahhosseini, A. Anzanpour, Iman Azimi, S. Labbaf, D. Seo, S. Lim, P. Liljeberg, N. Dutt , A. M. Rahmani , "Exploring computation offloading in IoT systems," *Information Systems,* vol. 107, p. 101860, 2022.

[2] S. Chen, Q. Li, M. Zhou, and A. Abusorrah, "Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm," *sensors,* vol. 31, no. 3, p. 779, 2021.

[3] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "A survey on computation offloading modeling for edge computing," *ournal of Network and Computer Applications* , vol. 169, p. 102781, 2020.

[4] A. Masoud Rahmani, M. Mohammadi, A. Hussein Mohammed, S. H. Taher Karim, M. Kamal Majeed, M. Masdari, M. Hosseinzadeh, "Towards Data and Computation Offloading in Mobile Cloud Computing: Taxonomy, Overview, and Future Directions," *Wireless Personal Communications,* vol. 119, no. 1, pp. 147-185, 2021.

[5] Z. Zhang, C. Li, S. L. Peng, and X. Pei, "A new task offloading algorithm in edge computing," *EURASIP Journal on Wireless Communications and Networking,* vol. 2021, pp. 1-21, 2021.

[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 ofGrid Computing,* vol. 18, no. 4, pp. 639-671, 2020.

[7] H. Huang, J. Tian, G. Min,W. Miao, Introduction to blockchain, London, United kingdom: The Institution of Engineering and Technology, 2020, pp. 2-19.

[8] H. -N. Dai, Z. Zheng and Y. Zhang,, "Blockchain for Internet of Things: A Survey," *IEEE Internet of Things Journal,* vol. 6, no. 5, pp. 8076-80p4, 2019.

[9] S. Naz, S. Uk-Jin Lee, "Why the new consensus mechanism is needed in blockchain technology?," in *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*, Antalya, Turkey, 2020.

[10] Sh. Guo, X. Hu, S. Guo, S. Member, X. Qiu, Feng Qi, "Blockchain Meets Edge Computing: A Distributed and Trusted Authentication System," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 3, pp. 1972-1983, 2019.

[11] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, W. Dou, "BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 6, pp. 4187 - 4195, 2019.

[12] X. Huang, X. Liu, Q. Chen and J. Zhang, "Resource allocation and task offloading in blockchain-enabled fog computing networks," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, Norman, OK, USA, 2021.

[13] I. Mohamed, H. Al-Mahdi, M. Tahoun, H. Nassar , "Characterization of task response time in fog enabled networks using queueing theory under different virtualization modes," *Journal of Cloud Computing,* vol. 11, p. 21, 2022.

[14] Z. Liao , S. Cheng, "RVC: A reputation and voting based blockchain consensus mechanism for edge computing-enabled IoT systems," *Journal of Network and Computer Applications,* vol. 209, 2023.

[15] O. M. Al-Tuhafi , E. H. Al-Hemiary, "Adaptive thresholds for task offloading in IoT-edge-cloud networks," in *2023 International Conference On Cyber Management And Engineering (CyMaEn)*, Bangkok, Thailand, 2023.

[16] J. L. Devore, K. N. Berk, M. A. Carlton, Modern Mathematical statistics with applications, New York: Springer, 2012.

[17] W. J. Stewart, robability, Markov chains, queues, and simulation : the mathematical basis of performance modeling, 2009.

[18] Iversen, V.Baek, Teletraffic engineering and network planning, Technical University of Denmark, 2010.

[19] J. (. Medhi, Stochastic models in queueing theory, 2009.