AOI :10.100.234513.0258
13th international conference on Microelectronics, Circuits and Systems, Micro2026

Efficient Energy Management For IoT Based Systems Using Multi-Server In Edge Computing

Payal Kavadia*, Zaineb Naaz*, Dr. Vidushi Sharma**

*Department of Computer Science and Technology,
**Department of Electronics and Communication Engineering,
Gautam Buddha University, Greater Noida - 201312, UP, India
Corresponding author: vidushi@gbu.ac.in

ABSTRACT

The expansion of Internet of Things (IoT) has reshaped the daily life by connecting wide range of smart devices, that work intelligently and autonomously. The large volumes of data, requires high computational power and extensive storage, which is provided typically by cloud platforms. However, depending solely on Cloud for time-sensitive or mission critical applications can pose challenges, as they require minimal service latency and immediate responses. Thus, Edge computing has emerged as a layer that enables real time and localized processing of the tasks. With fluctuations in data load, the performance of edge nodes may vary depending on their resources and capacity. So, it is essential to schedule tasks based on the priority and available resources, which enhances the Quality of Service (QoS) and the performance of edge nodes. Queuing Theory, which is based on task arrival and service patterns, provides a structured framework for modelling such systems with either single server or multiple servers. The Queuing theory presents different models such as, single-server and multi-server models which can be employed to schedule tasks based on the available resources at the servers and improve overall performance of the system. This study provides an analytical approach by comparing these two models at the Edge Computing layer, which is based on time taken for executing all the data packets in the system, average waiting time or average length of the system.

Keywords: IOT, Edge Computing, Queuing Models, Load Balancing

C. INTRODUCTION

The Internet of Things (IoT) has changed our understanding of how devices can manage even the smallest tasks through real-time monitoring, intelligent decision-making, and timely interventions. These devices work by constantly gathering and transmitting data to centralized servers or cloud platforms for further storage and analysis. This is field is growing so fast that it is expected to see about 40 billion IoT devices all around the world by 2035. As the data is growing exponentially, there is a need to find better ways of task offloading, pre-processing of the task and making optimal use of available resources.

Although cloud servers offer massive computational and storage resources, their centralized feature often creates bottlenecks like high latency, bandwidth limitations, and network congestion**.** To fix these challenges, an intermediate layer has been introduced by researchers, called Edge Nodes. This layer is introduced between IoT devices and cloud servers, bringing computation closer to the data source. This change in the architecture of IoT system has improved the features of latency and bandwidth usage, and also reduced dependency on central or cloud servers, making the system faster and more efficient.

The edge devices have limited computational and storage resource**s**, and because of these limitations, it can affect the overall efficiency, scalability, and reliability of the IoT system. Hence, there is a growing need for methodologies that can efficiently manage data traffic while simultaneously enabling real-time decision-making in edge computing environments. As traditional edge computing systems behave like a single server model, it can be thought as behaving similarly to a single server model in Queuing Theory.

Queuing Theory, a branch in mathematics, can help in understanding how the tasks form a queue and how they are processed by servers. This theory can help in understanding how the requirements of low latency, optimal resource utilization and real-time decision-making capabilities in edge computing, can be achieved. This theory models and assesses task arrival and service patterns in systems that have single or multiple servers. In this context, models such as single-server and multi-server are employed to simulate various operational scenarios. These models enable comparative analysis of system performance, waiting time in the queue, length of the queue, and resource utilization under varying data loads and service demands. The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 provides an overview of edge computing and queuing theory. Section 4 explains the proposed model. Section 5 presents simulation setup parameters, performance metrices and analyses the results. Finally, Section 6 concludes the work.

II. RELATED WORK

Over the years, various methodologies have been explored in the field of edge computing to enhance the

978-81-689089-0-1

efficiency and effectiveness of this layer. In [1] quantitative performance analysis of edge nodes is done under varying conditions. This analysis is evaluated based on three metrices: data load, data processing time and energy consumption. In [2], a multi-level queuing model classifies tasks into Very Urgent, Urgent, Moderate, Non-Urgent categories by calculating utilization factor of each task The results demonstrate a significant reduce in queue delays for priority-based tasks and improves system efficiency. Further research [3] explored queueing models for QoS in IoT interactions, considering message drop probabilities and the prioritisation of critical data. The study also proposed a delay-aware application offloading model. In [4], a dynamic QoS-based task scheduling method for IoT edge networks, where tasks are moved across priority queues based on delay, load, and urgency, is presented. This approach reduces waiting time, improves throughput, and ensures that critical tasks are processed first. It enhances overall performance under varying IoT workloads. Another work [5] introduces LMGT, a game-theoretic load-balancing approach designed to manage high data volumes in IoT-based edge computing. In [6], a study based on middleware architecture for distributed IoT using combination of cloud, fog/edge, queue modelling. The middleware components are modelled as multi-server queuing system to represent how tasks are processed across cloud-fog infrastructure. A dynamic scaling algorithm is used to evaluate performance under increasing load and shows a sub-linear scalability which is considered to be realistic and practical compared to ideal models. In [7], a priority-based allocation using single server (M/M/1) model and energy aware offloading method is adopted to improve responsiveness, reliability and overall efficiency. [8] presents a secure edge-computing architecture for IoT that performs encrypted data aggregation at edge nodes to reduce latency and overhead. Results show improved energy efficiency, lower communication cost, and enhanced security compared to traditional cloud-based IoT models.

Although there exists study about single-server queuing model at edge level and different simulation models showcasing IoT interactions, there is a significant gap in case of multi-server queuing model at edge nodes. The performance parameters can be used to improve the efficiency of a multi-server system model. In this research, we have proposed a multi-server system that can be used to process incoming tasks in edge computing.

III. PRELIMINARIES

In this section, we start with looking into the traditional edge computing systems and then followed by single-server (M/M/1) and multi-server (M/M/c) queuing models.

C. Edge Computing

Edge computing is a computing model that shifts data processing and storage away from centralized cloud data centers to the “edge” of the network, closer to where data is generated or consumed. The term “Edge” refers to intermediate devices like routers, gateways, and switches which acts as a link between data sources and cloud centers. The data sources include IoT devices like sensors, wearables and smartphones.

As show in Figure 1, the edge nodes collect data from various IoT sources like sensors, smartphones, preprocesses it and then sends preprocessed data to the cloud centers. The edge nodes are responsible for functions like data aggregation, which increases the responsiveness of the system and thus the lifespan of an IoT network gets extended.

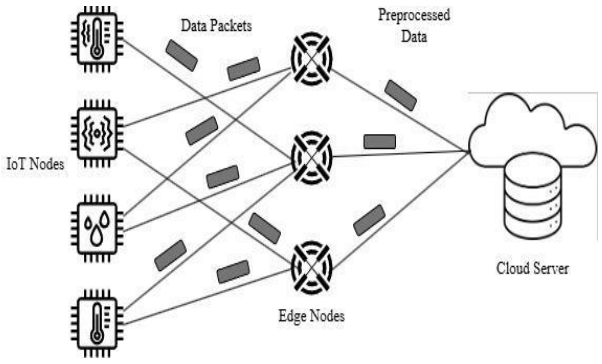

Fig. 1 Edge Computing Paradigm

The traditional edge computing systems behave like a single server model which can be considered as to be similar to an M/M/1 queuing model.

B. Single-Server (M/M/1) Model

This is a basic queuing model where the tasks follow a random arrival and are served by a single server. The arrival pattern is defined as λ and the service pattern is defined as μ. The utilization factor (ρ) is calculated using the ratio of arrival pattern (λ) and service pattern (μ). This factor is vital for predicting wait times and queue lengths. While this model is effective for minimizing delays under light loads, queues start forming at the server as traffic grows. When the utilization factor (ρ) becomes greater than 1, the queue grows indefinitely, making it impossible for a single-server system to process data packets. To address the bottleneck caused by high utilization factor, we can transition to a multi-server model, a solution well-supported and easily modelled through Queuing Theory.

978-81-689089-0-1

C. Multi-Server (M/M/c) Model

This model works similarly to the M/M/1 model but introduces multiple servers to handle incoming tasks. In this setup, we assume that tasks arrive randomly and are completely independent of one another. The arrival rates (λ) and service rate per server (μ), along with number of servers© is used to calculate the utilization factor (ρ). The service pattern for a multi-server model (μ_s) is calculated using number of servers © times service rate at each server (μ).

The service pattern is defined as follows:

$$\mu_s = c * \mu, where\ c = no.of\ servers \quad (1)$$
$$\mu_s = service\ rate\ of\ system$$

This calculation ensures that number of tasks are always greater than number of servers belonging to a queue in a multi-server model.

The utilization factor (traffic intensity) is determined as,

$$\rho = \frac{\lambda}{\mu_s}, \quad (2)$$
$$\rho\ is\ traffic\ intesity, \lambda = Poisson\ arrival\ rate$$

This factor should be less than 1 for processing of tasks effectively.

The probability of zero customers is defined as,

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{\left(\frac{\lambda}{\mu}\right)^n}{n!} + \frac{\left(\frac{\lambda}{\mu}\right)^c}{c!}\left(\frac{1}{1-\rho}\right)\right]^{-1}, \quad (3)$$
$$where\ P_0\ is\ the\ probability\ when\ server\ is\ idle$$

The length of the queue is measured using,

$$L_q = (P_0\,(\lambda/\mu)^c \rho)/(c!\,(1-\rho)^2) \quad (4)$$

$where\ L_q\ is\ average\ number\ of\ tasks\ in\ a\ queue$

The average waiting time is measured using,

$$W_q = L_q/\lambda \quad (5)$$

$where\ W_q = waiting\ time\ in\ the\ queue$

The performance of a multi-server queuing model can be analysed by using key parameters such as the average waiting time in the queue and the number of tasks waiting in the queue to be processed. These parameters can be used as important benchmarks to measure system efficiency and can be compared directly to the single-server model, which can provide a thorough understanding of how the server configurations impact the edge computing environment. A comparison between single-server and multi-server models clearly shows how deploying multiple servers influence overall system performance, particularly with respect to resource utilization, data processing efficiency, and network congestion.

The tracking of residual energy after each batch of data packet processing enables a detailed assessment of energy consumption throughout packet processing. As the multi-server model distributes incoming workloads across multiple servers, it fundamentally promotes more balanced resource utilization, which in turn contributes to observable energy savings as compared to a single-server paradigm. This characteristic of distributed processing makes the multi-server framework particularly well-suited for energy-aware edge computing environments.

The monitoring of total time required to process all data packets provides a complete evaluation of system throughput and computational efficiency. The evaluation parameters like queue lengths, waiting time, energy utilization, and end-to-end processing time, form a well-structured framework for depicting the performance of multi-server queuing systems in various data load scenarios.

IV. SYSTEM MODEL

In this section, a clear overview of how the multi-server model functions and the specific system environment required to simulate it is shown.

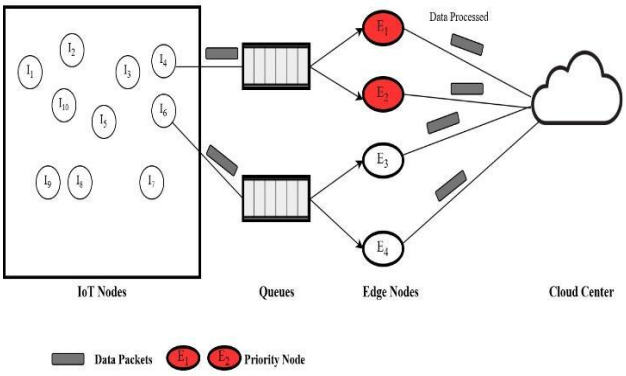

Fig. 2 Multi-Server (M/M/c) Network Model

The main goal of this model is to ensure that the tasks are offloaded in such a way that energy use is efficient across the edge computing environment. As show in Figure 2, we have ten IoT devices (I_1-I_{10}) that are scattered randomly in a defined space. Also, four edge nodes, that are labelled E_1 through E_4, are kept at fixed positions. A cloud server is also kept at a fixed position, where all the pre-processed data gets routed.

The system is built to handle four distinct types of data that are, Sensor Data, AI tasks, Emergency alerts, and Video files. As shown in the figure above, the edge nodes are paired up as E_1 and E_2, that act as a two-server team for Queue 1, while E_3 and E_4 act as servers for Queue 2.

978-81-689089-0-1

This means that for both queues, the number of servers (c) is equal to two.

To ensure critical information is handled quickly, all emergency tasks are automatically sent to Queue 1 to be processed by nodes E_1 or E_2. For every other type of task, the system is much more flexible, distributing them across all four nodes based on which one has the most energy or the lightest workload at that moment.

As shown in Figure 3, when a new task arrives, the system immediately identifies its type. If it is an emergency, it goes straight to the E_1/E_2 nodes, otherwise, it can go to any node from E_1 to E_4. the system constantly monitors the energy levels of each node. It always tries to pick the node with the highest energy level first. If two nodes have the same energy, it breaks the tie by choosing the one with the fewer data loads.

If a node's energy drops too low and hits a specific threshold, the system automatically redirects tasks to a different node with more resources. This continuous load balancing act keeps the network stable and reliable. Finally, as shown in Figure 3, we calculate packet loss once every available server has completely run out of energy. This packet drop continues, if there is no more energy left in the system.

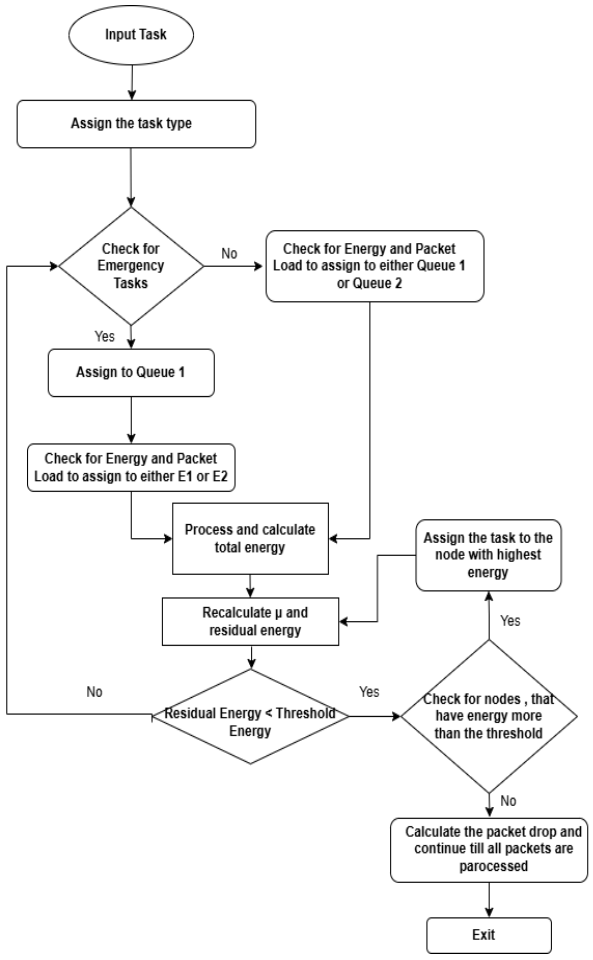

Fig. 3 Model Flowchart

A. Performance Metrices

To compare the suggested multi server model with the single server model, below metrices are used:

- **Queue Length**: It represents the total number of tasks waiting in the queue to be processed. It is based on the arrival rate, service rate and number of servers.
- **Waiting Time**: It refers to how long the task waits in the queue before being processed. It is also based on service rate, arrival rate and number of servers.
- **Residual Energy**: This is the specific amount of energy left at each node after all the data packets are processed.
- **Execution Time**: This is the total amount of time taken by the system to complete the processing of all the available data packets.

V. SYSTEM SIMULATION

The simulation of the proposed system is performed using MATLAB. The IoT Nodes are distributed randomly in the defined area, but the edge nodes and the cloud node are placed at fixed positions. The data is considered to be homogeneous and the number of tasks in the queues are always more than the number of servers (to form a queue of data). The packet size of data sent by IoT nodes is considered to be of 4000 bits. The other details about the system simulation parameters are given in the following Table 1:

TABLE 1
SIMULATION PARAMETERS

Parameters	Values
Number of normal nodes(I_1-I_{10})	10
Number of Edge Nodes(E_1-E_4)	4
Field Size	1000m × 1000m
Number of packets	500,750
Batch Size	10 Data Packets
Packet Size	4000bits
Electronics Energy (E_{elec})	50nJ/bit
Energy for Data Aggregation (E_{DA})	50nJ/bit/signal
ε_{fs}	10pJ/bit/m^2
ε_{amp}	0.0013pJ/bit/m^4
E_0	20J
Energy Threshold	4J

978-81-689089-0-1

A. Results & Discussions

1) Performance Of Multi-Server (M/M/c) Model

To understand how well the system works, this section determines the results of all the parameters mentioned above

- **Queue Length:** Figure 4 illustrates how queue lengths are distributed over the different nodes in the system. For Edge Nodes 1 and 2, the queue length is 0.027 at a load of 500 packets, and then increasing to 0.035 when the load is 750 packets. Similarly, Edge Nodes 3 and 4 has a queue length of 0.017 for 500 packets, which also rises to 0.035 at a load of 750 packets.

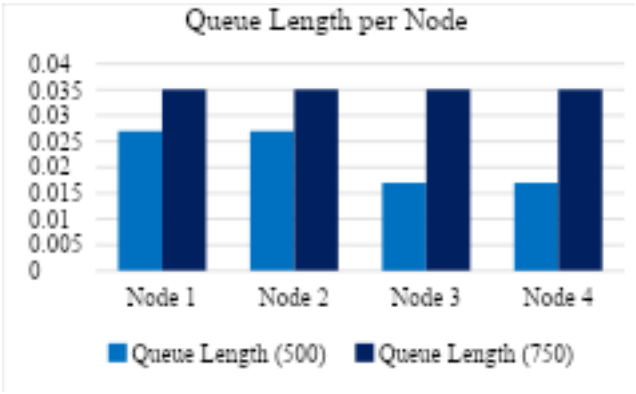

Fig. 4 Queue Length per Node

- **Waiting Time:** Figure 5 illustrates the waiting times for each node under varied data loads. Edge Nodes 1 and 2 show identical performance, with a waiting time of 0.011 seconds for 500 packets and 0.019 seconds for 750 packets. Similarly, Edge Nodes 3 and 4 share similar results, maintaining waiting time of 0.008 seconds and 0.017 seconds for 500 and 750 data packets, respectively.

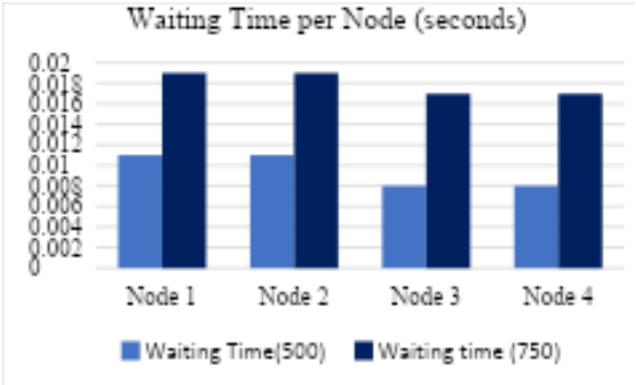

Fig. 5 Waiting time per Node (seconds)

- **Residual Energy:** Figure 6 illustrates the remaining energy levels of the system following the successful processing of 500 and 750 data packets. While the residual energy naturally decreases as the packet load increases, the system remains operational and has not yet reached its critical depletion threshold.

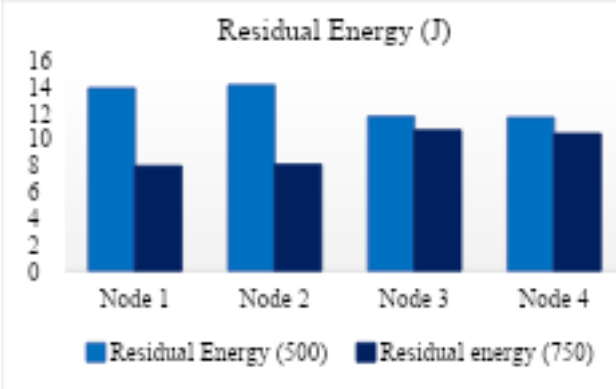

Fig. 6 Residual Energy per Node (J)

- **Execution Time:** The graph in Figure 7 illustrates the system's performance in terms of execution time as the packet load increases from 500 to 750. While a higher volume of data packets naturally leads to longer processing periods, the rise in execution time remains marginal, demonstrating the system's efficiency under increased traffic.

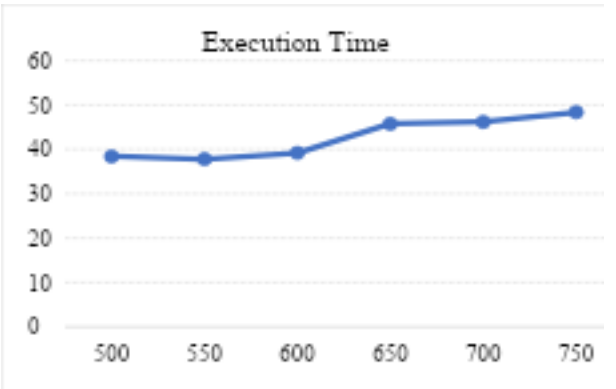

Fig. 7 Overall Execution Time (seconds)

2) Performance Comparison of single-server (M/M/1) and multi-server (M/M/c)

This section compares the basic single-server queuing model (M/M/1) and multi-server queuing model (M/M/c).

- **Queue Length:** As illustrated in Figure 8, the comparative analysis of the single-server and multi-server models across all nodes for 500 data packets, demonstrates a significant performance disparity. While the single-server model results in substantial

978-81-689089-0-1

queue accumulation, the multi-server system maintains negligible queue lengths, showcasing its superior efficiency in handling data traffic.

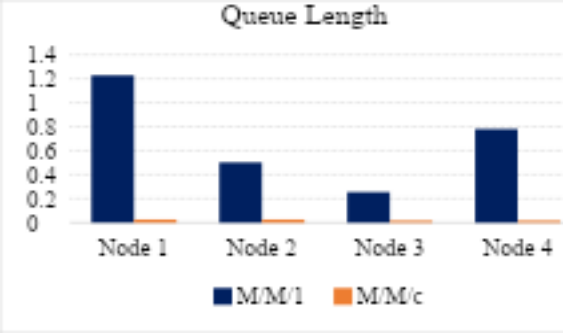

Fig. 8 Queue Length single-server Vs multi-server

- **Waiting Time:** The following graph in Figure 9 illustrates the substantial difference in the waiting time between single and multi server models when the load is 500 data packets. While the basic model experiences delays at each node, the waiting time for the multi-server model is minimal, highlighting the efficiency of the multi-server model in reducing latency.

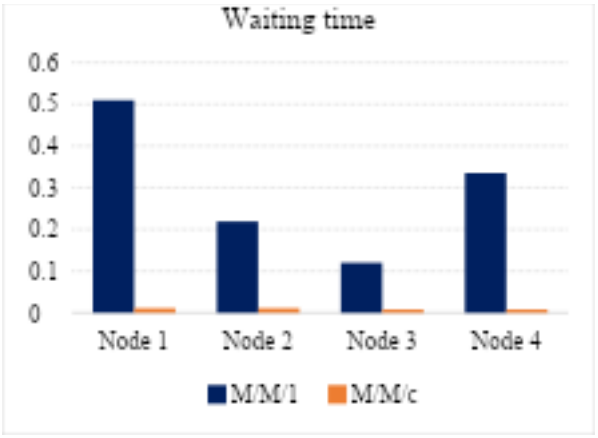

Fig. 9 Waiting time single-server Vs multi-server

- **Overall Residual Energy of the System:** Figure 10 illustrates the residual energy levels across the edge computing system. Starting from an initial capacity of 20J per node which amounts to a total system energy of 80J, it determines the residuals after complete task execution. It indicates that the multi-server model maintains a marginally higher energy level compared to the single-server system across packet loads varying from 500 to 750

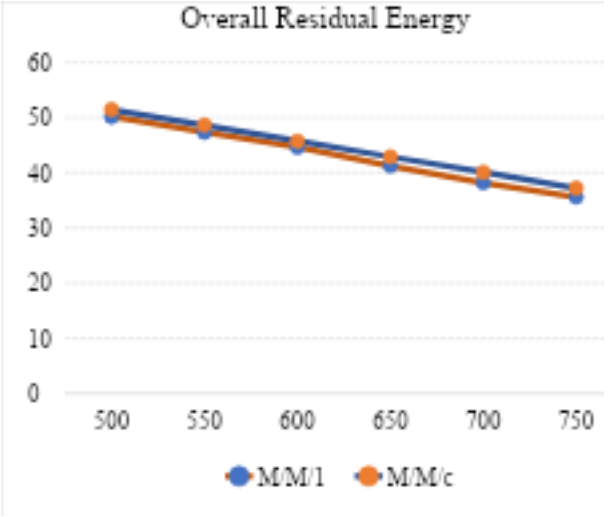

Fig. 10 Overall residual Energy (J) single-server Vs multi-server

- **Execution Time:** The graph in Figure 11 illustrates that the multi-server model maintains a performance advantage over the basic single-server model in terms of total execution time for varying data loads. While both models experience longer durations as the number of data packets increases, the basic model displays a 41% increase in time compared to a controlled 26% rise for the multi-server model, when packets are increased from 500 to 750. This difference confirms that the multi-server model achieves faster execution under higher data loads.

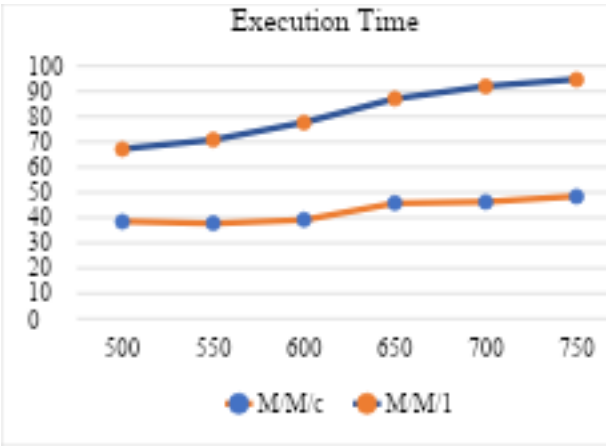

Fig. 11 Execution Time single-server Vs multi-server

The proposed model, that is multi-server works much better than basic single-server system when managing the length of the queue and waiting time for the system. Also, a significant gap is found in processing the entire data packets in single-server and multi-server systems, where the multi-server system completes a task much faster as compared to the single-server basic system.

978-81-689089-0-1

VI. CONCLUSION

In this paper, we present a smart task offloading model designed to boost efficiency through a multi-server system that resembles the M/M/c queuing model phenomena. By moving to this multi-server setup, the system achieves a major performance leap, cutting down both waiting times and queue lengths by about 95% at every node. We also see a significant jump in processing speeds, as the execution time drops by over 74% when handling 500 data packets and stays 48% lower even as the load climbs to 750 packets. In addition to improving speed, the model enhances battery conservation by balancing the workload more effectively, which keeps the overall residual energy higher across the edge computing network. As a result, it reduces data loss and ensures a more stable and dependable data processing environment.

REFERENCES

[1] Z. Naaz and V. Sharma, “Performance Analysis of Edge Node in IoT Network,” *in Proceedings of the 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT),* Greater Noida, India, 2024, pp. 546–551, doi: 10.1109/IC2PCT60090.2024.10486497.

[2] M. S. Bali, S. Vashist, M. Sharma, and S. Verma, “An Effective Technique to Schedule Priority Aware Tasks to Offload Data on Edge and Cloud Servers,” *Measurement: Sensors, vol. 26,* p. 100703, 2023*,* doi: 10.1016/j.measen.2023.100703

[3] G. Bouloukakis, I. Moscholios, N. Georgantas, and V. Issarny, "Performance Analysis of IoT Interactions via Simulation-Based Queueing Models," *Future Internet*, vol. 13, no. 4, p. 87, Mar. 2021, doi: 10.3390/fi13040087.

[4] Q. Aini, W. Al-Khateeb, N. Ali, R. Salleh, and M. Y. I. Idris, "Dynamic Performance Based on QoS for IoT Services in Edge Computing: A Task Scheduling Approach," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 7, pp. 546–552, 2020, doi: 10.14569/IJACSA.2020.0110770.

[5] Z. Naaz, G. Joshi, and V. Sharma, “Load-balancing model using game theory in edge-based IoT network,” *Pervasive and Mobile Computing*, vol. 109, p. 102041, Apr. 2025, doi: 10.1016/j.pmcj.2025.102041.

[6] D. Rathod and G. Chowdhary, "Scalability of M/M/c Queue based Cloud Fog Distributed Internet of Things Middleware," *International Journal of Advanced Networking and Applications, vol. 11, 2019,* doi: 10.35444/IJANA.2019.11015.

[7] Efficient Task Offloading in Edge Computing Using Priority Based Queuing and Energy Optimisation, *Proc. IEEE International Conference on Intelligent Systems and Embedded Computing*, August 2025, doi: 10.1109/CISES66934.2025.11264850

[8] Z. Naaz, G. Joshi, and V. Sharma, “Secure and Efficient Edge Computing Framework for IoT,” *in Proceedings of the 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT),* Delhi, India, Jul. 2023, doi: 10.1109/ICCCNT56998.2023.10307200.

[9] J. K. Sharma, *Operations Research: Theory and Applications*, 6th ed., Trinity Press, 2016, ISBN 978-9385935145.

[10] K. B. K. Reddy, M. Balaji, A. M. Baba, Y. G., and D. Ramesh, “An enhancing load balancing and security in edge computing: Comprehensive review analysis,” in Challenges in Information, Communication and Computing Technology, vol. 2, V. Sharmila et al., Eds. London: Taylor & Francis, 2025, pp. 721–726. doi: 10.1201/9781003559092-124.

[11] M. S. Aslanpour, A. N. Toosi, M. A. Cheema, M. B. Chhetri, and M. A. Salehi, “Load balancing for heterogeneous serverless edge computing: A performance-driven and empirical approach,” Future Generation Computer Systems, vol. 154, pp. 266–280, 2024, doi: 10.1016/j.future.2024.01.020.

[12] M. K. Deepak, K. Upadhyay and M. Alam, "Load Balancing Techniques in Fog and Edge Computing: Issues and Challenges," *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)*, Greater Noida, India, 2024, pp. 210–215, doi: 10.1109/IC2PCT60090.2024.10486765.

[13] D. Jiang, B. Zhu, X. Liu and S. Mumtaz, "ALCoD: An Adaptive Load-Aware Approach to Load Balancing for Containers in IoT Edge Computing," *IEEE Internet of Things Journal*, vol. 11, no. 23, pp. 37480–37492, Dec. 1, 2024, doi: 10.1109/JIOT.2024.3427642.

[14] Z. Naaz, G. Joshi and V. Sharma, "Secure and Efficient Edge Computing Framework For IoT," in *Proceedings of the 2023 International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 6–10, 2023, doi: 10.1109/ICCCNT56998.2023.10307200.

978-81-689089-0-1

[15] Z. Naaz, N. Chauhan, and V. Sharma, "OTM: An efficient task management and offloading scheme using proximity-based clustering, fuzzy logic, and optimization in edge-cloud IoT systems," Journal of Grid Computing, vol. 24, no. 1, p. 1, 2026, doi: 10.1007/s10723-025-09820-7.

Cite this article as:

Payal, Vidushi and et. al. "Efficient Energy Management For IoT Based Systems Using Multi-Server In Edge Computing", Proceedings of 13th international conference on Microelectronics, Circuits and Systems, Micro2026, (Vol-II). Displayed as online on 18th July 2026.

Link: http://actsoft.org/science/micro2026-pro/234-micro2026.pdf

978-81-689089-0-1