Volume 1, 2025, Pages 22-32 *Journal of Computer and Communication Networks*
| Regular Article | Open Access

Performance Evaluation of Fog Computing for Latency and Energy Efficiency in IoT Applications

Natarajan K
Department of Electrical and Electronics Engineering, Trinity College of Engineering and Technology, Peddapalli, Telangana, India.
drknatarajan17@gmail.com

Article Info
Journal of Computer and Communication Networks
https://www.ansispublications.com/journals/jccn/jccn.html

© The Author(s), 2025.
https://doi.org/10.64026/JCCN/2025003

Received 15 December 2024
Revised from 20 January 2025
Accepted 18 February 2025
Available online 24 February 2025
Published by Ansis Publications

Corresponding author(s):
Natarajan K, Department of Electrical and Electronics Engineering, Trinity College of Engineering and Technology, Peddapalli, Telangana, India.
Email: drknatarajan17@gmail.com

Abstract – A simulation through iFogSim evaluates the performance between cloud-only and edge-ward placement strategies in Fog computing environments when used for traffic surveillance and an online multiplayer game. An evaluation operated via iFogSim framework conducts tests on essential metrics, which include latency and energy consumption alongside network usage, RAM usage, and data transfer rate measurements. Our study establishes that using edge-ward strategy leads to substantial performance improvement through minimal latency while decreasing both energy consumption and network usage particularly when Fog devices operate at network edges during the traffic surveillance scenario. The edge-ward strategy delivers better scalability with shorter control loop delays which delivers enhanced performance for users in the analysis of the online game case study. On the other hand, the fog-based strategy maintains steady performance improvements for RAM utilization through enlarged deployment numbers combined with sustained minimal overhead. Fog computing demonstrates strong potential for improving the performance alongside resource usage in both IoT applications and latency-centric implementations thus serving as a viable solution for solving traditional cloud-based system constraints.

Keywords – Fog Computing, Edge-Ward Execution, iFogSim, Latency Optimization, Energy Efficiency, Network Utilization, Real-Time EEG Processing.

I. INTRODUCTION

The term “Fog Computing” was coined by Cisco Systems as a novel approach to facilitate wireless transmission of data to scattered devices inside the Internet of Things (IoT) system. Gultepe et al. [1] assert that the name was coined to describe this concept since fog is essentially cloud situated nearer to the earth's surface. Consequently, cloud computing conducted nearer to users' networks is known as fog computing. This refers to virtual framework situated between cloud data centers and consumer devices on the web. Consequently, fog computing may enhance service quality regarding latency, energy usage, and decreased data flow over the Internet, among other factors. The primary characteristic of fog computing is its capacity to facilitate applications necessitating mobility, spatial awareness, and low latency. This capability is facilitated by the application of fog computing models near end-users in a broadly scattered fashion.

Within this field of computing, Mishra et al. [2] proposed a scheduling methodology for the assignment of modules on fog nodes, concentrating only on the optimization of reaction time among elements, while neglecting the whole end-to-end service time. They formulated a resource estimation and pricing model for the IoTs that quantifies the resources required for a certain service, although fails to provide a solution for selecting the appropriate node for service allocation. The scheduling of latency-sensitive services has been extensively researched in the Cloud. Virtual machine (VM) methods for distributed clouds have been established in [3]. These methodologies aim to achieve appropriate virtual machine placement on physical nodes to decrease network latency between them.

Nonetheless, current mapping approaches depend only on service requirements and physical infrastructure knowledge, neglecting user-related information, such as geolocation, which is essential for the scheduling process in edge computing. Consequently, they are not immediately applicable to our issue. Within the framework of individual cloud providers, Cao, Li, and Stojmenovic [4] formulated methodologies for service element placement which, akin to the previously described distributed scenario, optimize the allocation of services among servers within a single center, thereby reducing the transfer duration between components. Consequently, these techniques fail to consider user information, rendering their application unsuitable for scheduling at the edge. Latency-sensitive applications, such augmented reality, driverless cars, and industrial automation, need immediate reaction times to provide maximum user experience and operational efficiency. Conventional cloud computing, albeit providing extensive processing capabilities, often incurs delay owing to data transmission over the network. Edge computing mitigates this problem by positioning computer resources nearer to information sources, facilitating real-time data handling and evaluation.

Nevertheless, edge devices possess constrained computing capabilities and storage capacity in comparison to cloud servers. Edge cloud collaboration leverages the advantages of both paradigms to successfully tackle the issues posed by latency-sensitive applications. Performance analysis and comparison between edge-ward and cloud-only placement strategies in Fog computing systems will be performed for two applications such as traffic surveillance and a latency-sensitive digital game. The goal of this research is to discover how well Fog computing performs by measuring latency and energy consumption along with network utilization and RAM usage and data transfer rate for determining application resource optimization and performance enhancement.

The remaining sections of our paper has been arranged as follows: Section II provides a literature review of (i) fog computing architectures and their applications, (ii) modeling tools for edge and fog computing, and (iii) performance metrics in cloud and fog computing. Section III describes the methodology we employed, and includes simulation framework, mathematical models for performance evaluation, experimental configuration and scalability as well as execution time analysis. In Section IV, a detailed presentation and discussion of results has been done, and it integrates two case study (Traffic Surveillance Application, and Latency-sensitive Online Game). Lastly, Section V concludes the research and shows how the application of fog computing is fundamental in operations with high latency needs and limited resources.

II. RELATED WORKS

Fog Computing Architectures and Their Applications

Baccarelli et al. [5] contend that the fog computing reference architecture is a substantial area of study. Several designs for fog computing have been presented over the past few decades. They mostly originate from the essential three-layer framework. Fog computing enhances cloud computing services by incorporating intermediate fog layers between the cloud structure end devices at the network edge. **Fig. 1** illustrates the fog computing hierarchical structure. In this design, every smart item and end device is linked to one of the fog units using wireless connectivity technologies (mostly includes Bluetooth, ZigBee, 4G, 3G, WiFi, and WLAN etc.) or via a physical link. Fog units can be interlinked and communicate with each other via wired or wireless transmission methods. Each fog unit is connected to the cloud network through the IP core network. This architecture may provide technological support for Mobile Internet, CPS, and IoT, facilitating effective storage services and data processing. Fog computing is capable of enhancing the effectiveness and quality of service (QoS) for Cyber-Physical Systems (CPS), which incorporate storage capabilities, communication, and computing to track or manage physical facilities and items, particularly in the context of the current data explosion.

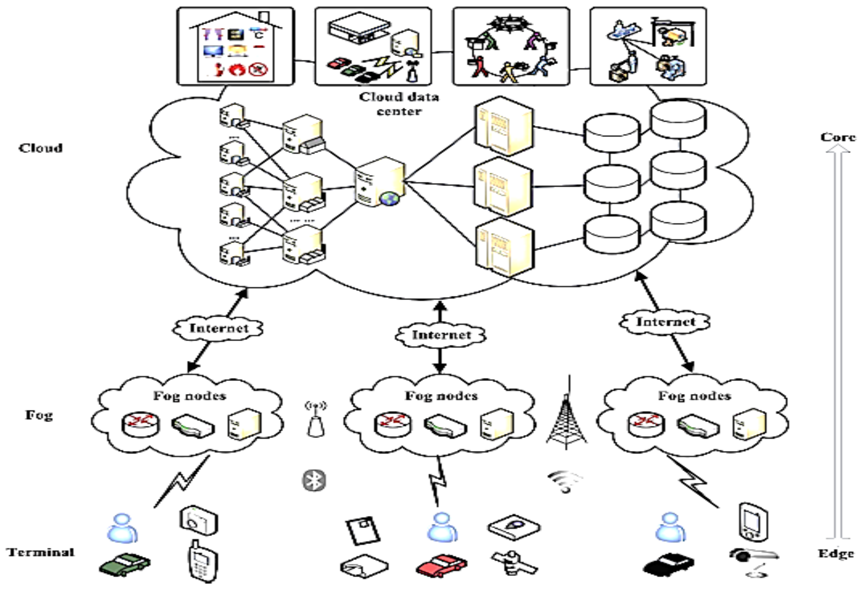


Fig 1. Hierarchical Fog Computing Framework

Hu et al. [6] focused on fog computing but did not propose a cohesive architecture that may be used across many application contexts. The duration of cloud processing may vary based on server demand and network speed. Particularly for mobile devices, delays may be prolonged due to the comparatively limited wireless network bandwidth. To accommodate the pervasive devices, several academics have suggested the mobile fog computing system. This computational approach improves performance and decreases energy usage in mobile environments. Mobile fog computing complements fog computing by delivering low-latency mobile services.

Simulation Tools for Fog and Edge Computing

According to Adday et al. [7], simulation has been widely used to replicate conventional network architectures, including prevalent WSNs. Some of these simulators include Avrora, ATEMU, J-Sim, OMNeT++, EmStar, TOSSIM, and NS-2. These simulators are often employed to build and evaluate network standards, particularly during the preliminary design phase. They were not conceived for edge and fog computing settings; hence, they fall beyond the purview of this work. Although several simulators exist for cloud computing, there are very few available for simulating fog and edge computing situations.

Gupta et al. [8] succinctly delineate a range of notable simulators employed for edge and fog modeling and conduct a qualitative comparison among them. FogNetSim++ [9] is a fog simulation application that offers end-users comprehensive configuration options for modeling an extensive fog network. It is built upon OMNeT++ [10], an open-source program that offers a comprehensive framework for simulating network features using discrete event modeling. FogNetSim++ allows scholars to integrate tailored mobility simulators, node scheduling methods, and manage handover procedures. A traffic control structure is assessed to illustrate the efficacy and scalability of the FogNetSim++ model for memory use and CPU. The scholars recommend a network metric benchmark, including latency, handovers, packet error rate, and execution delay. Nonetheless, FogNetSim++ still lacks capability for virtual machine migration across fog units.

iFogSim [11] is a simulation toolkit for fog computing, which enables end-users to model architectures and run modeled applications to assess systems of measurement such as energy consumption, network utilization, and latency. iFogSim is derived from and built upon Cloud Sim and facilitates the modeling of different systems to assess resource planning and control methods. It evaluates efficiency parameters and models cloud data centers, edge devices, network connections, sensors, stream-processing applications, and data streams. Furthermore, iFogSim incorporates modelled services for the management of resources and monitoring of energy at two distinct levels: application deployment and application timing. Two application component placement techniques are designed to accommodate various placement instances: (a) cloud-only placement, in which all application component operate inside data centres, and (b) edge-ward location, in which application components function on nodes in proximity to edge tools.

According to Prem Sankar and Ghaddar [12], extensions exist to facilitate the formulation of data placement methods aligned with particular goals, including the reduction of energy use, network congestion, and service latency. It is important to highlight that due to the many parallels between the fog and cloud computing models, CloudSim may function independently to implement various characteristics of fog computing. iFogSim has several restrictions. This information, although facilitating the identification of device locations receiving service from fog servers, remains static and is not refreshed by any mobility simulation. Moreover, while iFogSim's foundation on Cloudsim offers benefits, it is constrained to DES (discrete event simulation) and its adaptability is restricted. Both IoTsim and EdgeCloudSim, similar to iFogSim, are founded on CloudSim. EdgeCloudSim is especially intended to evaluate the networking and computational requirements of edge computation.

EdgeCloudSim, contrary to iFogSim, accommodates mobility. It offers the mobility, edge server, and network connection model to assess the different aspects of edge computation. Besides its modeling features, EdgeCloudSim is user-friendly, offering a method to get device and application configurations from XML files rather than requiring programmatic definitions. IoTSim was developed to emulate edge computing scenarios in which substantial information volumes are transferred to a big data computing framework by IoT applications. Consequently, it incorporates a large data processing layer and storage into CloudSim. The storage layer simulates storage and network latencies for IoT implementations. The processing layer emulates MapReduce to facilitate the batch-based computing model. Both IoTsim and EdgeCloudSim have a similar discrete event simulation restrictions and scalability as iFogSim.

Gill and D. Singh [13] recently introduced a prototype simulator, FogTorchII, which builds upon their prior research, Fog Torch. FogTorchII refers to an open-source model created in Java, primarily intended to facilitate application deployment in the fog. It may assess fog computing infrastructure installations by modeling software features (operating frameworks, systems, and programming languages etc.), hardware features (storage, RAM, and CPU cores), and QoS parameters such as data rate and delay. FogTorchII employ Monte Carlo simulation to include fluctuations in communication networks utilized as an input. The last output comprises the consolidated findings about QoS guarantee and fog resources use, shown by the amount of storage and RAM used. A recognized and significant disadvantage of FogTorchII is its adaptability, a concern that Svorobej et al. [14] aim to mitigate by using heuristics to diminish the search space.

Performance Metrics in Cloud and Fog Computing

According to Hamdan, Ayyash, and Almajali [15], performance metrics are significantly influenced by the computing architecture or layer in which the IoT applications are staged. In a mist framework, IoT device accessibility for managing offloaded duties is a significant worry, but in a cloud layer/model, accessibility is assured. Within the cloud framework, assessing reaction duration is essential. Accessibility and reaction duration are essential in fog, cloud, mist, and edge computing. The extent to which every performance statistic is evaluated varies. Furthermore, emerging computing paradigms like fog and edge, which inherit specific traits from cloud computing, may exhibit parallels regarding potential performance measures. Certain metrics are inherently prevalent across all levels but possess specific criteria, like resource use, which may be quantified based on VMs, containers, or a combination of both, contingent upon the relevant computing layer. These measurements are categorized individually for every stratum.

According to White [16], the various performance measurements are distributed throughout the taxonomy, depending on the features of each cloud model. Certain metrics need examination only for certain cloud models, like privacy, which is essential for private cloud systems. Within these systems, sensitive organizational data is sent among authorized users, necessitating the prevention of data leakage. Moreover, with federated clouds, privacy becomes a critical issue as several suppliers collaborate, potentially leading to conflicting privacy standards. Conversely, there are special indicators that are unique to the cloud and are similarly significant for every cloud architecture. Metrics encompass: Throughput; Numerical analysis techniques; the quantity of Scoring choices, Contrary decisions, Contrary actions, and Fluctuating mitigations; Energy Use; and Temperatures. Nonetheless, there are additional metrics that must be assessed for all cloud models, although with varying levels of significance. Metrics encompass: Fault Detection, User SLA Violations (except Private), Profit/Cost, Scalability, Latency Delay Time, Response Time, Max Running Containers or VMs, Deprovisioned Provisioned Resources, Resource Lifetime, Resource Load, Resource Utilization.

III. METHODOLOGY

Simulation Framework and Environment Setup

iFogSim served as the foundation for simulation experiments that deal with cloud data centers as well as Fog nodes and edge devices. A high-performance computing device ran the system featuring an Intel Core i7-10750H processor together with 16 GB DDR4 RAM alongside Ubuntu 20.04 LTS. The simulations operated from Java (OpenJDK 11) [17] with added components to iFogSim for controlling workload management and monitoring performance. The research design utilized a multi-level network system to observe results under modified test situations. The network consisted of smartphones and IoT sensors together with surveillance cameras as edge devices and local servers as Fog units and cloud system data centers as the core components. The testing environments displayed a range of network speeds and delays together with computing resources which enabled researchers to assess the advantages of Cloud-based implementation in relation to Edge-based deployment.

Mathematical Models for Performance Evaluation

Latency Computation Model

The system latency calculations used the method of measuring total processing delay together with communication delay that occurred between distinct layers. The system latency for each Fog computing topology required Eq. (1) to evaluate the total processing and communication delays.

$$L_{total} = \sum_{i=1}^{n} \left(L_{proc,i} + L_{comm,i} \right) \tag{1}$$

The total transaction latency equals the delay experienced at i^{th} computational nodes including both processing along with communication times ($L_{proc,i} + L_{comm,i}$). The EEG-based gaming system requires multiple sensor-processing unit interactions thus developers computed its transaction latency using Eq. (2).

$$L_{EEG} = \sum_{j=1}^{m} \left(\frac{D_j}{B_j} + \frac{C_j}{F_j} \right) \tag{2}$$

The calculation considers three key parameters which include data size D_j for the j^{th} transaction, available bandwidth B_j and CPU cycle requirement C_j alongside processing unit speed F_j.

Energy Consumption Model

The system's total energy usage calculation depended on both Fog node power consumption together with cloud server power usage. The mathematical model described energy consumption behavior of computing devices through Eq. (3) at a specific time t.

$$E(t) = P_{idle} + U(t) \times (P_{max} - P_{idle}) \tag{3}$$

The equation describes power utilization based on three components: P_{idle} being the power at idle state, P_{max} representing power at full utilization with $U(t)$ showing CPU utilization at time t. The total energy consumption during the simulation duration T depended on the integral calculation of power usage throughout the period in Eq. (4).

$$\mathrm{E_{total}} = \int_0^{\mathrm{T}} \mathrm{E(t)} \ \ \mathrm{dt} \tag{4}$$

For mobile and IoT devices, which operate on battery power, the remaining battery level at time t was estimated using Eq. (5).

$$\mathrm{B(t)} = \mathrm{B(0)} - \int_0^{\mathrm{t}} \frac{\mathrm{E(\tau)}}{\mathrm{C_{battery}}} \ \ \mathrm{d\tau} \tag{5}$$

where $B(0)$ is the initial battery capacity, and $C_{battery}$ is the total battery charge.

Network Usage Model
The network utilization measurement focused on dividing the total data transfer volume by the available data bandwidth. The simulation period data transfer equation was presented as Eq. (6).

$$D_{\text{total}} = \sum_{k=1}^{P}(S_k + A_k \times T_k) \tag{6}$$

The equation uses three variables: static data size S_k together with average arrival rate A_k and active transmission time T_k. Network utilization was computed using Eq. (7).

$$NU = \frac{D_{\text{total}}}{B_{\text{network}}} \tag{7}$$

where $B_{network}$ is the total network bandwidth available.

Heap Memory Allocation Model
The system surveillance tools tracked heap memory allocation for various workload scenarios. The processing memory requirement for each workload was calculated using Eq. (8).

$$M_{heap} = \sum_{l=1}^{q} M_{obj,l} \times N_l) \tag{8}$$

where $M_{obj,l}$ is the memory footprint of the l^{th} object type, and N_l is the number of such objects instantiated.

Experimental Configurations And Network Parameters
The network configuration for the EEG-based game simulation operated with changing numbers of WiFi gateways which maintained a constant device-to-gateway ratio throughout. Five different physical topology configurations were tested during the study which used increasing numbers of WiFi gateways as well as connected smartphones. The simulation implemented the network link properties as presented in **Table 1**.

Table 1. Properties of the Links in the EEG-Based Gaming Simulation Network Through Network Link Properties

Device 1	*Device 2*	*Latency (ms)*	*Bandwidth (Mbps)*
EEG Headset	Smartphone	10	1
Smartphone	WiFi Gateway	5	10
WiFi Gateway	ISP Gateway	20	50
ISP Gateway	Cloud Data Center	100	1000

Scalability and Execution Time Analysis
Simulation assessment for scalability involved running tests using different topological configurations which spanned from 4 edge devices up to 64 devices and from 1 WiFi gateway to 16 gateways. The computational execution duration of every configuration was determined through Eq. (9).

$$T_{exec} = \sum_{n=1}^{N}\left(T_{init,n} + T_{comp,n} + T_{trans,n}\right) \tag{9}$$

The computational process includes three main components: initialization time $T_{init,n}$ for the n^{th} device and processing duration $T_{comp,n}$ along with data transmission time $T_{trans,n}$. The system execution time grew proportionally to the addition of more nodes through an estimated mathematical relationship in Eq. (10).

$$T_{exec} \approx a \times N + b \tag{10}$$

where a and b are system-dependent constants determined through empirical fitting.

IV. RESULTS AND DISCUSSION

This section presents simulation findings of fog computing ecosystem for application case scenarios. Subsequently, we assessed the efficiency of the two placements techniques (i.e., edge-ward and cloud-only) for energy expenditure, network use, and latency for every case scenario. Ultimately, we assessed iFogSim's scalability concerning RAM usage and data transmission rates across several simulation situations.

Case Study 1 — Traffic Surveillance Application
This scenario was executed and modeled via iFogSim. The subsequent outcomes were noted according to the new model. Five criteria were used to assess the model's performance: data transfer, energy consumption, network usage, RAM utilization, and average latency rate.

Average Control Latency
Loop latency is a critical element in IoT applications that requires evaluation. This pertains to the latency occurring in communications between the cloud and smart cameras, which may be quantitatively assessed as indicated in Eq. (11).

$$Average\ Latency = CC - ET \tag{11}$$

The delays in the application loop are determined by Eq. (11), in where CC represents the CloudSim Clocks and ET denotes the Emitting Duration of a tuple. ET is determined by transmitting the time from one module to another. Upon the detection and tracking of a single vehicle, as seen in the first curve, the performance of cloud-only dataset centers significantly escalates in module execution, indicating a rise in latency see **Fig. 2**. The Fog placement effectively maintains low latency by situating modules inside the control loop ending to the networking edge.

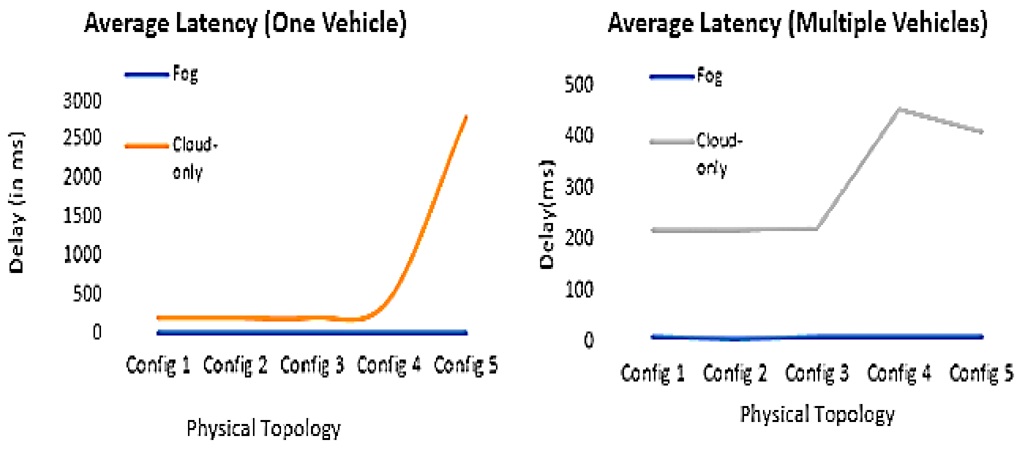


Fig 2. Latency of a Single Vehicle vs Multi-Vehicle Detection

The findings in the second curve indicate that Multiple Vehicles Detection exhibits lower latency compared to targeting a single car. Cloud-only exhibits increased latency relative to the initial curve, while Edge ward demonstrates consistency in maintaining low latency. This results from the substantial amount of data transferred to the cloud system through intelligent cameras for processing and analysis. The new model exhibits very reduced latency while using Fog devices.

Energy Use
The energy use is computed for the whole network architecture using Eq. (12).

$$\text{Energy} = \text{CEC} + (\text{NT} - \text{LUUT}) \times \text{HP} \tag{12}$$

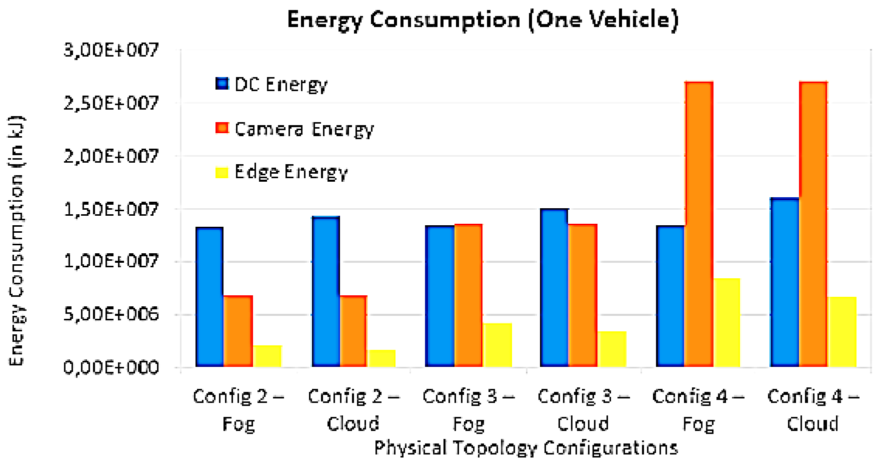
The energy use is determined by the energy of all hosts throughout a specified execution duration, in which CEC represents the present energy use, NT denotes the current time, LUUT indicates the latest usage update duration, and HP signifies the host energy within LU (last utilization). LU is determined by the formula $LU = Min\ (1, \frac{TMA}{TM})$, where TMA refers to the allotted host MIPS.

Fig. 3 illustrates that the energy expenditure in multiple vehicle monitoring exceeds that of single vehicle tracking. Numerous cameras use substantial power for motion detection while recording video streams. Consequently, as seen in **Fig. 3**, energy consumption in these devices escalates with the expansion of monitored regions. It may be determined that when operators transition to fog phones, energy usage in the cloud data center diminishes.

Network Usage
Network Usage denotes the quantity used during the execution of a simulation. This may be quantitatively expressed by Eq. (13) as follows

$$Network\ Usage = \frac{TL \times TC}{MST} \tag{13}$$

Where TL signifies overall latency and TS implies overall tuple size, correspondingly. MST represents the maximum time used for simulation. In **Fig. 4**, the addition of one car substantially elevates the overall number of devices linked to the system, resulting in a large rise in network demand for both cloud-only and edge-only applications. This observation may be ascribed to the predominance of data-intensive communications occurring via low-latency networks in Fog-oriented executions.



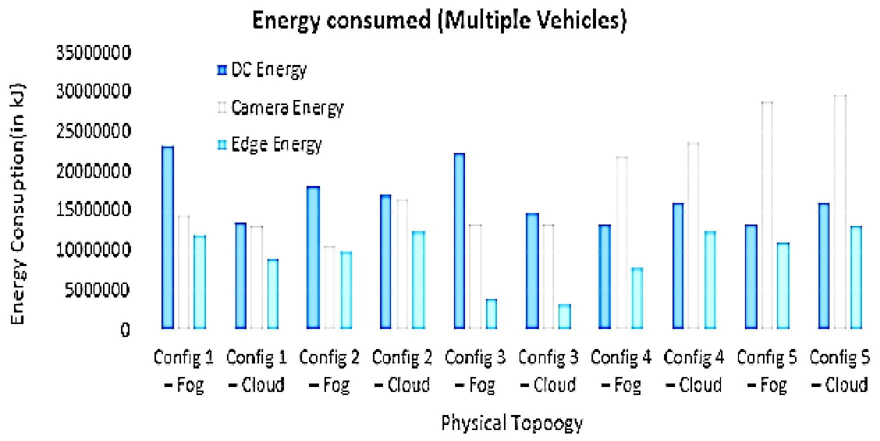
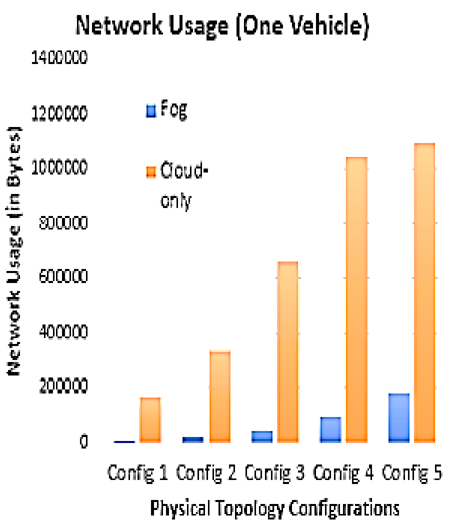


Fig 3. Energy Consumption



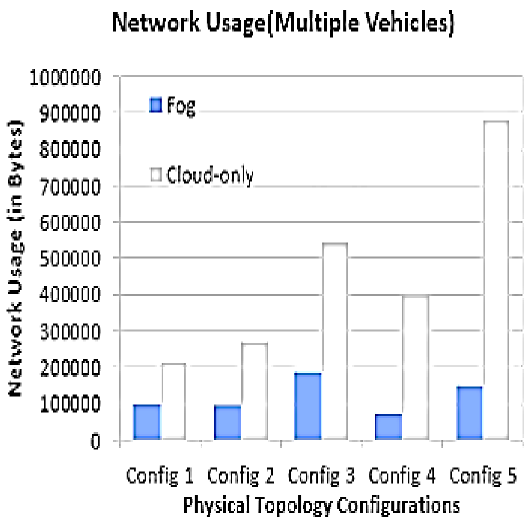


Fig 4. Network Usage or Single-Vehicle or Multi-Vehicle Detection

Componets like Object Tracker and Object Detector are therefore placed on edge computing systems that basically decreases the amount of data transferred to the main cloud system data centers. When identifying several cars, the Cloud exhibits elevated network use compared to edge devices, which likewise see a significant surge as the workload escalates, necessitating data transmission to the cloud for storage when the Fog system is overwhelmed.

RAM Used

The massif heap profiler from the Valgrind software package [18] was employed to assess heap assignment during modeling of diverse input demands and topology sizes. The increase in workload and physical topology does not substantially affect heap allocation. **Fig 6** illustrates that when the quantity of sensors (smartphones) and gateways rises, iFogSim exhibits minimum overhead space scaling.

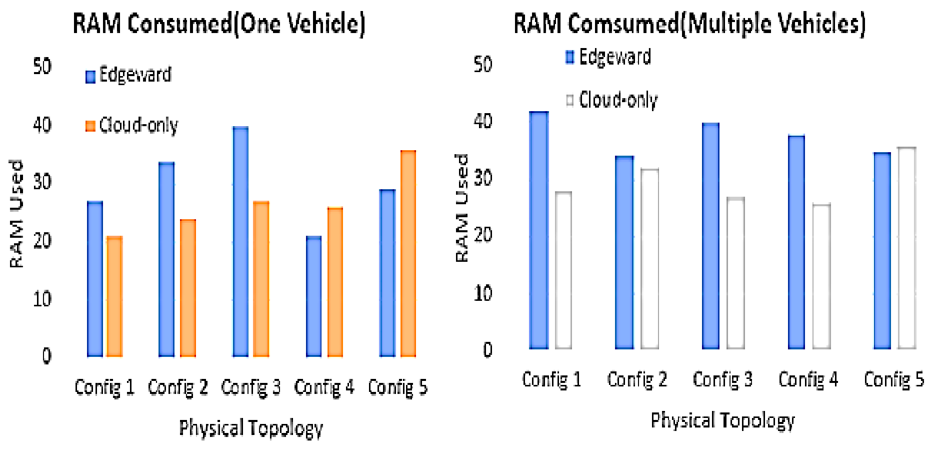


Fig 5. Analysis of RAM use During the Simulation

Fig 5 indicates that the RAM used for monitoring a single vehicle is less than the RAM required when many vehicles are participating in the cloud. Fog exhibits a substantial rise in RAM consumption on devices due to an extensive workload being processed.

Data Transfer Rate
The data transmission rate denotes the quantity of data exchanged between devices. The speed at which a certain volume of data may transmit is affected by bandwidth. A larger bandwidth of a certain route correlates with an increased data transmission rate. **Fig 6** below indicate that Cloud has a fast data transmission speed, which becomes inconsistent as the number of configurations increases.

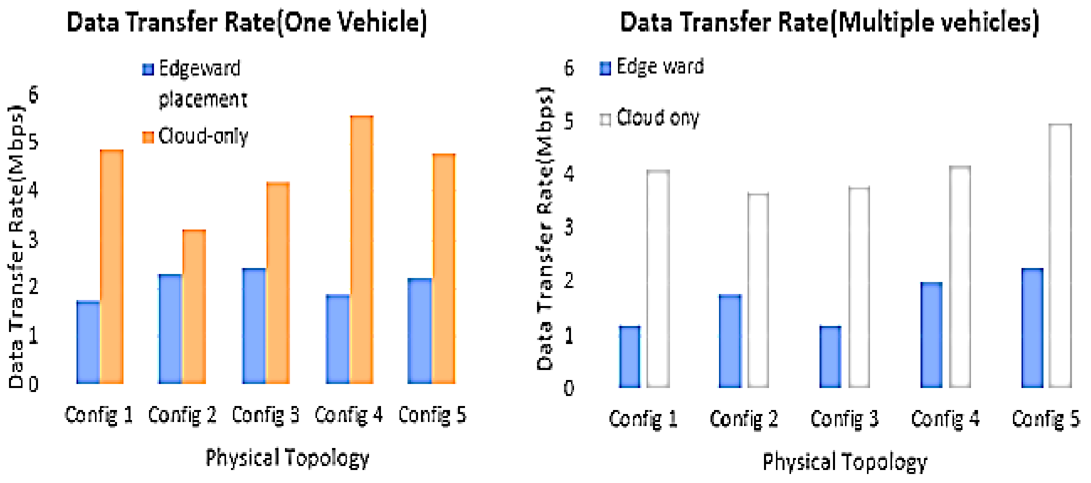


Fig 6. Data Transfer Rate Between Devices

The finding indicates that data transmitted to the Cloud does not increase with workload, and also demonstrates a higher transfer rate compared to Edge devices in both circumstances. In fog conditions, tracking a single car has superior performance compared to tracking several vehicles simultaneously. The data transmission rate while targeting many cars is very high.

Case Study 2 — Latency-Sensitive Digital Gaming
The model of this case scenario was conducted for a duration of 3 hours, during which numerous systems of measurement given by iFogSim were gathered. The simulation findings illustrate the effects of varying input tasks and allocation strategies on network utilization and total transmission delay. Each headgear is linked to a smart device using Bluetooth connection. Devices connect to the web via WiFi linked to ISP Gateways. To evaluate iFogSim's efficiency over different network configurations, we have altered the quantity of WiFi access points while maintaining a constant number of devices linked to every gateway. Approximately 5 hardware network layouts have been modeled: Config 1, 2, 3, 4, 5, with 1, 2, 4, 8, and 16 WiFi gateways, correspondingly, each linked to four devices engaged in the EEG Tractor-Beam gaming simulator. The

application efficiency on the Fog is contingent upon the latency of the connections interconnecting devices. Within the simulated topology, various devices exhibit distinct capacities, as indicated in **Table 2**.

Table 2. Specification of Network Connections for EEG Tractor-Beam Gaming Simulator

Sources	Destinations	Latencies (in ms)
ISP Gateway	Cloud DC	100
WiFi	ISP Gateway	4
Smartphones	WiFi	2
EEG Headset	Smartphone	6

Average Control Loop Latency
The fundamental EEG Tractor gaming simulator's control loop [19], concerning response latencies, is the loop that converts the user's brain activity into the game condition shown on smart devices. This necessitates real-time connectivity between the device and smartphones holding the brain condition classification component, as well as rapid processing inside the categorization module. Latency within this loop may significantly impair end-users' experience since it impacts the entity with whom the user personally engages. **Fig. 7** depicts the mean execution latency of this control loop. **Fig. 7** illustrates a significant reduction in control loop execution latency when using the Edge-ward placement technique that utilizes Fog devices for processing. The decrease becomes much more evident when tuple emission rates and topology sizes grow (i.e. Headset B).

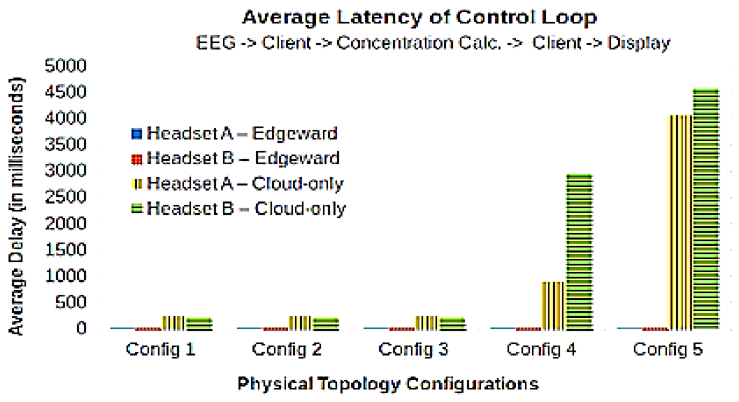


Fig 7. Mean Control Loop Latency

Network Usage.
Fig 8 illustrates the network usage of the EEG Tractor gaming simulator. The rise in the number of smart devices linked to the application substantially elevates the network load reliant only on cloud resources. **Fig. 8** illustrates that the incorporation of Fog devices significantly reduced network use.

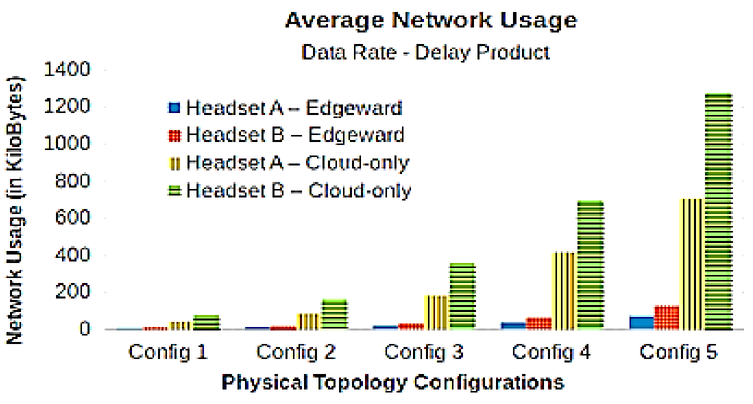


Fig 8. Comparison of Network Usage.

This result might potentially be seen as evidence of the scalability of Fog-based systems. The unregulated increase in network utilization during cloud-based execution may result in network congestion, therefore impairing the application's performance. Such circumstances may be more effectively circumvented by the use of fog-based deployment.

Energy Consumption.
Fig. 9 illustrates the energy usage of several categories of devices in the simulation. **Fig. 9** illustrates that utilizing Fog instruments within an Edge-ward deployment approach decreases energy usage in cloud system data centers whereas somewhat elevating energy usage in edge computation.

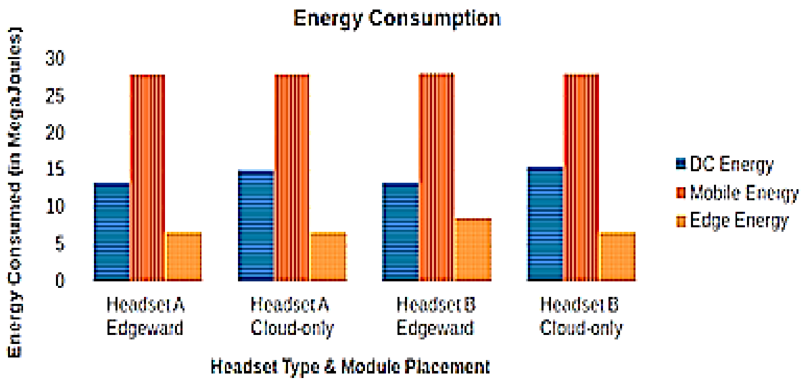


Fig 9. Energy Usage of Smart Devices in Fog and Cloud Computing

Table 3. Specification of Network Connections within the Physical Architecture for Intelligent Surveillance

Sources	*Destinations*	*Latencies (ms)*
ISP Gateway	*Cloud DC*	*100*
Area GW	*ISP Gateway*	*2*
Camera	*Area Switch*	*2*

RAM Usage

Fig 10 illustrates that heap allocation remains relatively stable despite an increase in workload and physical topology size in **Table 3**. **Fig 10** illustrates that iFogSim exhibits little memory overhead whenever the amount of gateways and sensors (smartphones) rises from 1 to 16, and 4 to 64, respectively.

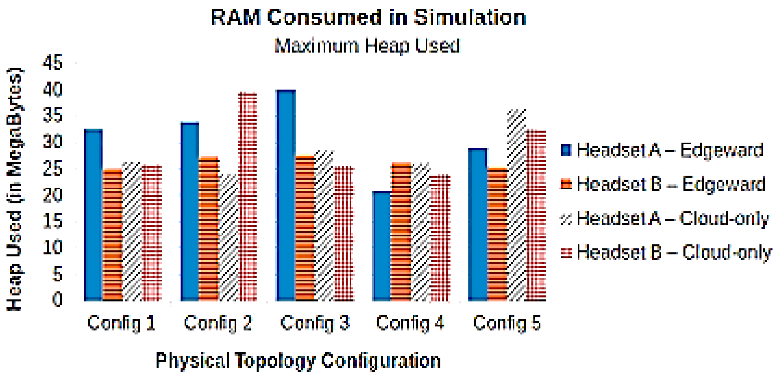


Fig 10. RAM Usage of The Model for Diverse Input Workloads and Topology Sizes

Data Transfer Rate

The simulation duration for different input workloads and topologies was quantified and shown in **Fig. 11**. **Fig. 11** illustrates that execution time escalates with an increase transmission rate and number of devices. The increase in simulation is approximately linear, allowing for execution within an acceptable timeframe of 25 seconds, even with the addition of a substantial number of gateways.

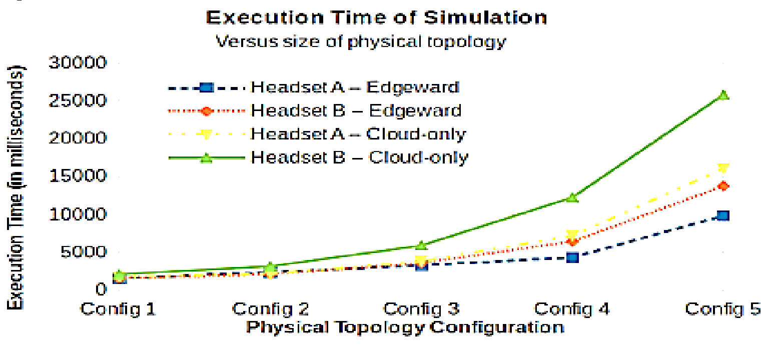


Fig 11. Simulation Execution Time for Different Topology Sizes and Input Workloads

V. CONCLUSION

The research shows how implementing Fog computing brings important advantages when used in operations with high latency needs and limited resources. An edge-ward operation strategy demonstrated better performance in traffic surveillance with reduced latency and energy usage along with decreased network utilization when compared to operating in the cloud exclusively according to the study results. The deployment of Fog infrastructure reduces cloud dependency thereby causing decreased data transfer volumes and facilitating better scalability when dealing with extensive and elaborate networks. The research demonstrates that when more devices and workload requirements appear the Fog-based system presents better RAM utilization and general system performance results. The combination of Fog computing with IoT applications produces enhanced performance and scalability and optimized resource management so it becomes a promising solution for upcoming IoT and smart applications.

CRediT Author Statement

The author reviewed the results and approved the final version of the manuscript.

Data Availability

The datasets generated during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interests

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Funding

No funding was received for conducting this research.

Competing Interests

The authors declare no competing interests.

References

[1]. I. Gultepe et al., “FOG Research: A review of past achievements and future Perspectives,” Pure and Applied Geophysics, vol. 164, no. 6–7, pp. 1121–1159, Jun. 2007, doi: 10.1007/s00024-007-0211-x.

[2]. S. K. Mishra, D. Puthal, J. J. P. C. Rodrigues, B. Sahoo, and E. Dutkiewicz, “Sustainable service allocation using a metaheuristic technique in a FOG server for industrial applications,” IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4497–4506, Jan. 2018, doi: 10.1109/tii.2018.2791619.

[3]. M. Masdari, S. S. Nabavi, and V. Ahmadi, “An overview of virtual machine placement schemes in cloud computing,” Journal of Network and Computer Applications, vol. 66, pp. 106–127, Jan. 2016, doi: 10.1016/j.jnca.2016.01.011.

[4]. N. J. Cao, N. K. Li, and I. Stojmenovic, “Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers,” IEEE Transactions on Computers, vol. 63, no. 1, pp. 45–58, May 2013, doi: 10.1109/tc.2013.122.

[5]. E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, “Fog of everything: Energy-Efficient networked computing architectures, research challenges, and a case study,” IEEE Access, vol. 5, pp. 9882–9910, Jan. 2017, doi: 10.1109/access.2017.2702013.

[6]. P. Hu, S. Dhelim, H. Ning, and T. Qiu, “Survey on fog computing: architecture, key technologies, applications and open issues,” Journal of Network and Computer Applications, vol. 98, pp. 27–42, Sep. 2017, doi: 10.1016/j.jnca.2017.09.002.

[7]. G. H. Adday, S. K. Subramaniam, Z. A. Zukarnain, and N. Samian, “Investigating and Analyzing Simulation Tools of Wireless Sensor Networks: A Comprehensive survey,” IEEE Access, vol. 12, pp. 22938–22977, Jan. 2024, doi: 10.1109/access.2024.3362889.

[8]. H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, “iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments,” Software Practice and Experience, vol. 47, no. 9, pp. 1275–1296, Jun. 2017, doi: 10.1002/spe.2509.

[9]. T. Qayyum, A. W. Malik, M. a. K. Khattak, O. Khalid, and S. U. Khan, “FogNetSIM++: a toolkit for modeling and simulation of distributed fog environment,” IEEE Access, vol. 6, pp. 63570–63583, Jan. 2018, doi: 10.1109/access.2018.2877696.

[10]. A. Varga, “A practical introduction to the OMNET++ Simulation Framework,” in EAI/Springer Innovations in Communication and Computing, 2019, pp. 3–51. doi: 10.1007/978-3-030-12842-5_1.

[11]. K. S. Awaisi, A. Abbas, S. U. Khan, R. Mahmud, and R. Buyya, “Simulating Fog computing applications using iFOGSiM toolkit,” in Springer eBooks, 2021, pp. 565–590. doi: 10.1007/978-3-030-69893-5_22.

[12]. G. Premsankar and B. Ghaddar, “Energy-Efficient service placement for Latency-Sensitive applications in edge computing,” IEEE Internet of Things Journal, vol. 9, no. 18, pp. 17926–17937, Mar. 2022, doi: 10.1109/jiot.2022.3162581.

[13]. M. Gill and D. Singh, “A comprehensive study of simulation frameworks and research directions in fog computing,” Computer Science Review, vol. 40, p. 100391, Mar. 2021, doi: 10.1016/j.cosrev.2021.100391.

[14]. S. Svorobej et al., “Simulating fog and Edge Computing Scenarios: An Overview and Research challenges,” Future Internet, vol. 11, no. 3, p. 55, Feb. 2019, doi: 10.3390/fi11030055.

[15]. S. Hamdan, M. Ayyash, and S. Almajali, “Edge-Computing Architectures for Internet of Things Applications: A survey,” Sensors, vol. 20, no. 22, p. 6441, Nov. 2020, doi: 10.3390/s20226441.

[16]. G. P. White, “A survey and taxonomy of strategy-related performance measures for manufacturing,” International Journal of Operations & Production Management, vol. 16, no. 3, pp. 42–61, Mar. 1996, doi: 10.1108/01443579610110486.

[17]. D. R. Cok, “OpenJML: JML for Java 7 by extending OpenJDK,” in Lecture notes in computer science, 2011, pp. 472–479. doi: 10.1007/978-3-642-20398-5_35.

[18]. M. Becker and S. Chakraborty, “A Valgrind tool to compute the working set of a software process,” arXiv (Cornell University), Jan. 2019, doi: 10.48550/arxiv.1902.11028.

[19]. W. Lu, Y. Wei, J. Yuan, Y. Deng, and A. Song, “Tractor Assistant Driving control method based on EEG combined with RNN-TL Deep Learning algorithm,” IEEE Access, vol. 8, pp. 163269–163279, Jan. 2020, doi: 10.1109/access.2020.3021051.