# Towards Energy-Efficient and Cost-Effective Task Offloading in Mobile Edge Computing for Intelligent Surveillance Systems

Manash Kumar Mondal, Sourav Banerjee, *SMIEEE,* Debashis Das, *Member, IEEE,* Uttam Ghosh, *SMIEEE,* Mohammed S. Al-Numay, *SMIEEE,* and Utpal Biswas

***Abstract*—In the era of pervasive digital connectivity, intelligent surveillance systems (ISS) have become essential tools for ensuring public safety, protecting critical infrastructure, and deterring security threats in various environments. The current state of these systems heavily relies on the computational capabilities of mobile devices for tasks such as real-time video analysis, object detection, and tracking. However, the limited processing power and energy constraints of these devices hinder their ability to perform these tasks efficiently and effectively. The dynamic nature of the surveillance environment also adds complexity to the task-offloading process. To address this issue, mobile edge computing (MEC) comes into play by offering edge servers with higher computational capabilities and proximity to mobile devices. It enables ISS by offloading computationally intensive tasks from resource-constrained mobile devices to nearby MEC servers. Therefore, in this paper, we propose and implement an energy-efficient and cost-effective task-offloading framework in the MEC environment. The amalgamation of binary and partial task-offloading strategies is used to achieve a cost-effective and energy-efficient system. We also compare the proposed framework in MEC with mobile cloud computing (MCC) environments. The proposed framework addresses the challenge of achieving energy-efficient and cost-effective solutions in the context of MEC for ISS. The iFogSim simulator is used for implementation and simulation purposes. The simulation results show that the proposed framework reduces latency, cost, execution time, network usage, and energy consumption.**

***Index Terms*—Intelligent Surveillance System, Mobile Edge Computing, Cloud Computing, Task offloading.**

## I. Introduction

INTELLIGENT surveillance systems (ISS) automatically monitor the environment or private infrastructure without or with minimal human interaction [1]. These systems can detect and track objects, such as people or vehicles, identify biometric features, such as faces or fingerprints, analyze events and behaviors, such as traffic violations or suspicious activity, and set off alarms or take other actions, such as locking doors. ISS is deployed in public places, transit, retail, banking, education, and healthcare, which increases security, safety, and effectiveness [2]. ISS contains several components like cameras, video processing servers, and monitoring consoles, where camera sensors record the video footage from the environment and send it to computing nodes for processing and analysis.

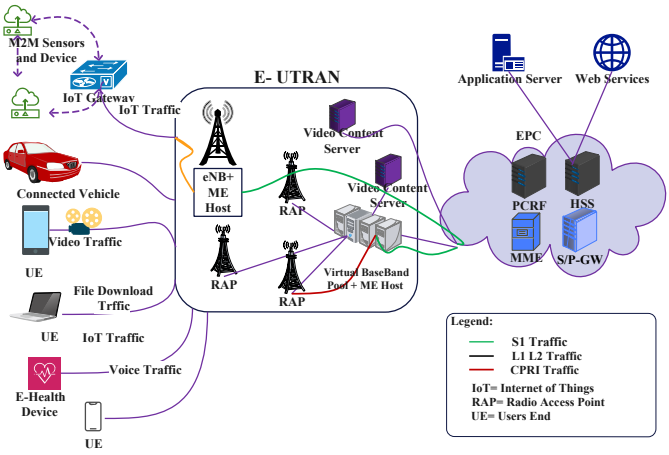



Fig. 1: MEC architecture in 5G scenario

The traditional surveillance systems process and analyze data in the cloud. The cloud service provider and end users maintain a service-level-agreement (SLA) between them. Therefore, these computing nodes are generally deployed in cloud data centers. The end customers utilize the pay-as-you-go (PAYG) scheme for cloud computing services [3]. PAYG is known for cost-effective solutions. Whereas, the Mobile edge computing (MEC) is a one-time establishment that is relatively higher. However, in the long run, MEC will provide a cost-effective solution because no subscription is needed. The main problems of cloud computing-based ISS include latency, cost of execution, bandwidth, energy consumption, data security, and user privacy [4]. Surveillance cameras generate a huge amount of data and transmitting these data to the cloud is time-consuming. Transferring analyzed data back and forth response increases the latency and data centers consume a lot of electrical energy for a tiny job due to their huge infrastructure. Since data transfer is the necessary criterion, data security and user privacy are vital issues of this system.

One of the potential solutions to these problems can be MEC enables ISS. MEC [5] is a paradigm that aims to enhance the capabilities and performance of mobile networks

M. K. Mondal and U. Biswas are with the Department of CSE, University of Kalyani, Kalyani, WB, India (e-mail: manashcse21@klyuniv.ac.in, utpalbiswas@klyuniv.ac.in).

S. Banerjee is with the Department of CSE, Kalyani Government Engineering College, Kalyani, WB, India (e-mail: mr.sourav.banerjee@ieee.org).

D. Das and U. Ghosh are with the Department of CS and DS, Meharry Medical College, Nashville, TN, USA (e-mail: debashis.das@ieee.org,ghosh.uttam@ieee.org).

M. S. Al-Numay is with the Department of Electrical Engineering, King Saud University, Riyadh 11451, Saudi Arabia (e-mail: alnumay@ksu.edu.sa).

TABLE I: List of Abbreviations

| Acronym | Definition |
|---|---|
| AG | Augmented Reality |
| DAG | Directed Acyclic Graph |
| E-UTRAN | Evolved UMTS Terrestrial Radio Access |
| EU | End User |
| GPS | Global Positioning System |
| HSS | Home Subscriber Server |
| ISP | Internet Service Provider |
| LTE | Long-Term Evolution |
| MEC | Mobile Edge Computing |
| MECNs | Mobile Edge Computing Nodes |
| MCC | Mobile Cloud Computing |
| MI | Million Instructions |
| MIPS | Million Instructions Per Second |
| MME | Mobility Management Entity |
| PAYG | Pay-As-You-Go |
| PCRF | Policy and Charging Rules Function |
| PTZ | Pan-tilt-zoom |
| S/P-GW | Serving / Packet Gateway |

by bringing computational resources closer to the network edge [6]. MEC addresses these challenges of cloud-based systems by deploying edge servers, known as mobile edge computing nodes (MECNs) or MEC servers, at the edge of the network [4]. The traditional CCTV cameras continuously record all events whereas intelligent cameras record specialized events. Intelligent cameras require less storage space to store recorded footage. MEC-based ISS processes the raw video footage at the MEC server instead of the cloud. Therefore, these massive amounts of data are not sent to the data center and analyzed in the MEC server. Here, The MEC server is responsible for object detection and object tracking. Region detection and camera feedback operations are done in the intelligent camera itself. The functions like alert generation and display results are also performed by the MEC server.

Therefore, in this paper, we proposed a MEC-based ISS framework. The binary and partial task offloading policies are utilized for uploading the task into the MEC server for analysis. The proposed framework is implemented in the iFogSim simulator.TABLE I defines the list of abbreviations used in this article. We make a comparative analysis between the proposed MEC-based ISS and MCC-based ISS. This proposed framework achieves significant improvements in several parameters like latency, network usage, energy consumption, execution time, and cost.

The subsequent sections of this paper are structured as follows: In Section II various related researches are reviewed. Section III provides a comprehensive overview of various system parameters pertaining to MEC. Section IV delineates the proposed system architecture and its corresponding implementation. System configurations and the experimental setup are expounded upon in Section V. Section VI offers a detailed exposition of the results, followed by an in-depth analysis. Finally, Section VII concludes this article with future works.

## II. Literature Survey

MEC was proposed by ETSI (European Telecommunications Standards Institute) in 2014 [7]. The surveillance system is essential for ensuring security. Previously, surveillance cameras used the main cloud server and conducted data governance. The traditional cloud-based client-server architecture is unable to stream videos from millions of source devices, which is not suitable for time-sensitive or real-time applications. MEC is one of the solutions to these problems. A large and growing body of literature has investigated MEC-related applications including surveillance systems. Computational offloading in MEC with various aspects was highlighted in the article [8]. Wang et al. [9] proposed a novel (L, 2) transfer feature learning (L2TFL) approach for COVID-19 classification in edge computing environments. A social distancing measuring framework using edge computing was proposed in article [10]. In this article, researchers proposed a model using fog and edge computing that computes the distance among several GPS-enabled edge devices in the MEC server. Also, latency and network had improved over MCC one. Sabella et al. [11] suggested a few related studies on the MEC. Routing optimization using a deep learning model is proposed for better network performance in MEC environments [12].

Elephant monitoring near railway tracks using an intelligent surveillance system was found in article [13].Cai et al. [14] proposed a linear multi-agent system on signed communication topology for the bipartite output consensus problem. The researchers provided some tutorials based on the ETSI reference MEC architecture on the 5G scenarios. Various social issues and challenges were covered in the article. The virtual machine (VM) provisioning algorithm was proposed in articles [15], [16]. VM placement and VM selection policies were proposed to reduce the system's energy consumption. He et al. [17] discussed dynamic opinion maximization in social networks. Secure V2V communication in an IoT-enabled MEC environment was discussed in article [18]. Beloglazov et al. [19] proposed an energy ware heuristic resource allocation model for efficient data center management. Ensuring the data security-related study in MEC using block-chain had found in article [20], [21]. Multi-model data fusion was used in neuroimaging to achieve significant benefits in clinical diagnosis and neuroscience research [22]. In article [23], the authors focused on the behavior of the caching system, caching insertion and expulsion policies, and caching optimization depending on wireless networks in the MEC environment. Regarding energy consumption concerns, green cloud computing significantly improves over cloud [24]. Mobile Edge Computing with 5G integration detailed discussed in article [25]. Using convolutional neural networks, transfer learning, and semi-supervised learning category of foods was recognized in the MEC environments [26], [27].

Mahmoodi et al. [28] introduced an innovative approach that integrates scheduling and cloud offloading for mobile computing. They discussed various factors such as business values, market drivers, and computing services like augmented reality, video acceleration, and IoT applications for connected vehicles. The study also highlighted the feasibility of deploying MEC servers at multiple locations, including LTE macro base stations (eNodeB), 3G radio access controllers (RNC), and various multi-radio access technology (RAT) points. Mondal et al. [29] proposed a tiger monitoring framework in edge and fog architecture. Collectively, these studies outline the crucial

TABLE II: Components of MEC

| Name of Components | Description |
|---|---|
| Low Latency | Devices being near to edge server, latency is very low. |
| Location Awareness | MEC server knows the geographical location of end devices. |
| Network context information | MEC server has network information based on the business model. |
| Proximity | Setting at a convenient location. |
| On-Premise | Use local resources, isolated from others. |

role of ISS in the MEC environment.

## III. Background and Overview

In this section, we introduce several terminologies related to the MEC system. This section contains four subsections. It includes two types of task offloading policies (Binary, and Partial) related to MEC, components of MEC, and energy consumption of mobile devices and MEC servers. Fig. 1 depicts the general architecture of MEC in the 5G scenario. Here, various services are accessed from the user's end (UE). E-UTRAN is associated with EPC. EPC consists of PCRF, HSS, MME, and S/P-GW [25]. PCRF is used for policy enforcement, data flow detection, and charging. User and subscriber information is stored in the HSS database. Subscriber authentication roaming and handover-related support are provided by MME.S/P-GW are responsible for routing-related activities in EPC.

### A. Components of MEC

As per the European Telecommunications Standards Institute (ETSI) white paper regarding MEC, it has several components. The components of MEC and its brief description are shown in TABLE II. The important components of MEC are low latency, location awareness, proximity, on-premises deployment, and network context information.

### B. Binary Offloading

Task offloading is an essential feature for cost-efficient task management. Two types of task offloading are available in MEC, binary task offloading, and partial task offloading. Binary overloading states that either the task is completely uploaded to the MEC server or not uploaded. Binary offloading can be represented as:

$$T(L, \tau_d, C_b) \tag{1}$$

This widely used equation (1) represents data offloading, the fact of input-data size of the task $L$ (in bits), the task completion deadline $\tau_d$ (in second), and $C_b$ represents computation workload. The deadline can be two types, hard deadline and soft deadline. These parameters depend on the nature of the task [30], [31].

### C. Partial Offloading

Due to the nature of the task and computation capability, binary offloading is not always ideal for offloading. Hence partial offloading comes into account. In partial offloading, the tasks are partitioned into parts. Some parts of the task are offloaded to the MEC server, others are executed on the mobile device itself. It is also called fine-grained (partial) computation offloading. The task input data are bit-wise independent and can be distributed over several MEC servers for parallel execution. This model belongs to the data partition model.

Another partial offloading model is the task-call graph model where the task will be uploaded or not depending on the dependency of other tasks. This model includes three variations, sequential dependency, parallel dependency, and general dependency. In the task-call graph model, the graph is a *directed acyclic graph* (DAG). $G\{V, E\}$, where $V$ represents the set of vertices as different procedures and $E$ represents the set of dependencies [32]. Fig. 2 depicts these three types of dependency using task-call graphs.

### D. Energy Consumption

In MEC systems, the CPU of mobile devices is fundamentally used for regional computations. The performance of the CPU depends on CPU-cycle frequency $f_m$. It is also called CPU clock speed. Generally, low-power mobile systems used *dynamic frequency and voltage scaling* (DVFS) [16], [33]. Execution latency of a particular task $T(L, \tau_d, C_b)$ can be estimated by the equation as:

$$t_m = \frac{LC_b}{f_m} \tag{2}$$

Total energy consumption of a task $T(L, \tau_d, C_b)$ along with the CPU cycle frequency $f_m$ of a mobile device can be derived as:

$$E_m = \kappa L C_b f_m^2 \tag{3}$$

where $\kappa$ is a constant related to hardware architecture. The server execution time is computed as $t_{s,k} = \frac{w_k}{f_{s,k}}$ where $w_k$ denotes the CPU cycle and $f_{s,k}$ denotes CPU cycle frequency required for the offload task. For MEC servers the energy consumption is different from mobile devices. The queuing delay along with total server computation latency for a device $k$ is symbolized by $T_{s,k}$ and can be calculated as:

$$T_{s,k} = \sum_{i \leq k} t_{s,i} \tag{4}$$

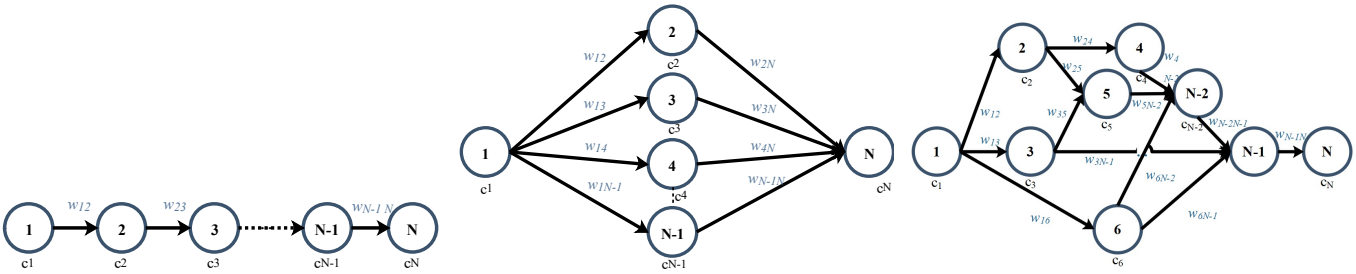
The energy consumption of the CPU at the MEC server can be derived as:

$$E_s = \sum_{k=1}^{K} \kappa w_k f_{s,k}^2 \tag{5}$$

where $w_k$ is the number of required CPU cycles for processing the offloaded workload at the MEC server.Equation (3), and (5) represent the energy consumption for mobile devices and MEC servers respectively [34].

## IV. Proposed Framework

This article proposed a MEC-based intelligent surveillance system and compared it with MCC in various parameters. We have taken a case study of surveillance systems. Using the module placement algorithm deployed applications in the MEC and MCC servers based on computational capability.




(a) Sequential dependency (b) Parallel dependency (c) General dependency

Fig. 2: Topologies of the task-call graphs

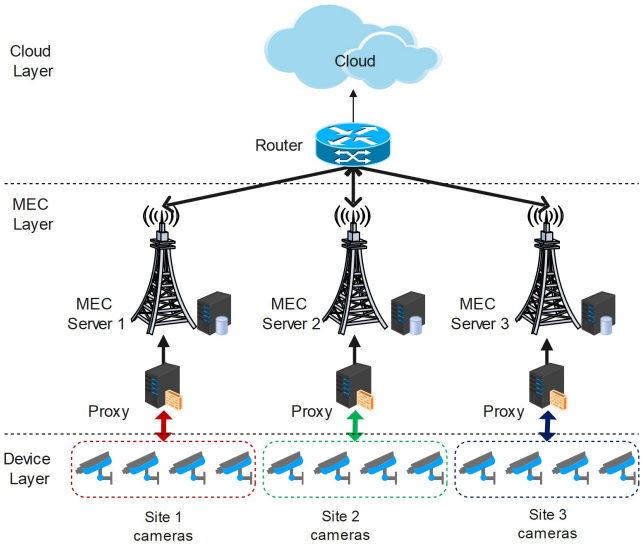



Fig. 3: Proposed MEC-based Intelligent Surveillance System framework

Binary task offloading is not as complex as partial offloading. As per the latency concern, binary offloading is more suitable than partial offloading. We have used both the task offloading schemes in the module placement algorithm to achieve cost-effectiveness. The complete job is divided into a few modules. The partial task offloading determines whether a module will be placed on the MEC server or the device itself.

Fig. 3 depicts the architecture of the ISS in the MEC environment. It is a three-tier architecture, the topmost layer contains a cloud data center. The middlemost layer contains the MEC server. The bottom layer consists of all the devices (sensor and actuator) called the device layer. The cameras generate video streams and upload them to the MEC server via the proxy server for video analytics. Cloud is used for storage purposes of video footage for future use. This figure contains three sites, each site having four surveillance cameras and one MEC server.

The workflow diagram of the proposed framework is depicted in Fig. 4. It consists of four modules *Region Detection*, *Object Tracking*, *Decision Making*, *Camera Feedback* that need to be deployed for the simulation. The *Alert Generation* function produces an alert signal in the MEC server and the result is reflected in the users via display units. In MCC-based placement, some modules deploy in the cloud. Whereas in MEC-based placements, some modules deploy in the MEC server. Sometimes few modules can be deployed in the intelligent camera itself and the rest of the modules deploy in the MEC server. Here, the application model consists of a DAG (Directed Acyclic Graph) $S = G\{V, E\}$. The vertices set $V$ represents the modules and the edge set $E$ represents connections among the modules. The four modules depicted in Fig. 4 belong to the set $V$. Modules are connected via some directed edges belonging to set $E$.

The Algorithm 1 demonstrates the scenario of how the modules are deployed in MCC and MEC servers. The MEC servers consist of VMs for processing the tasks simultaneously. Each VM is deployed for the processing of one task module. The number of VMs deployed on each MEC server depends on the hardware configurations of the MEC server. In the proposed algorithm, $d$ represents the number of edge devices, $f$ represents the number of MEC servers, $w$ represents the number of task modules belonging to apps, and $\theta$ represents the total number of modules. MEC server is selected as the parent among the edge devices. The parent device places the hierarchically upper layer in physical topology. The *PATHS* of DAG with $G\{V, E\}$ represented by $p$. The maximum time required to place the modules is $O(p \times d \times w)$. The maximum time required for choosing the appropriate servers (f) for $\theta$ modules is $O(f \times \theta)$. The time required to traverse the graph is $O(V + E)$. Additionally, Fig. 5 depicts the flowchart of the module placement algorithm 1 in the MEC environment.

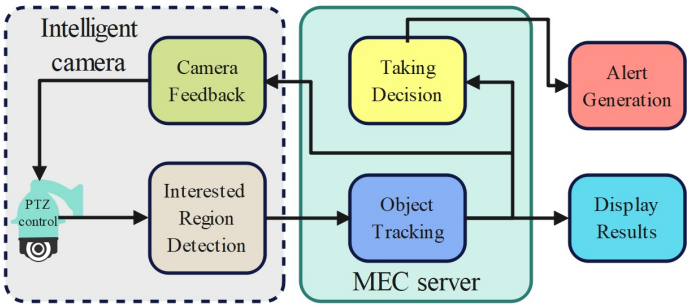



Fig. 4: Workflow diagram of proposed MEC-based intelligent surveillance system

**Algorithm 1** Module placement in MEC

Input: $module\ \theta$
Output: $placeList$

**for** $p \in PATHS$ **do**
  $Over\ all\ available\ paths\ of\ DAG$
  $init.\ placeList := \{\}$;
  **for** $Edge\ device\ d \in p$ **do**
    traverse from leaf to root
    **for** $module\ w \in app$ **do**
      **if** $total\ predecessors\ of\ w\ are\ alloted$ **then**
        add each $w$ to $placeList$;
      **end if**
    **end for**
    **for** $module\ \theta \in placeList$ **do**
      **if** $\theta\ is\ previously\ alloted\ on\ device\ f \in p$ **then**
        Combine $\theta$ with its other upstream instance;
        $f$:=device grasp combine instance;
        **while** $CPU_{\theta}^{rec} \geq CPU_{f}^{avail}$ **do**
          $f := parent(f)$
        **end while**
      **else if** $CPU_{\theta}^{rec} \leq CPU_{d}^{avail}$ **then**
        Append $\theta$ on device $d$;
      **end if**
    **end for**
  **end for**
**end for**

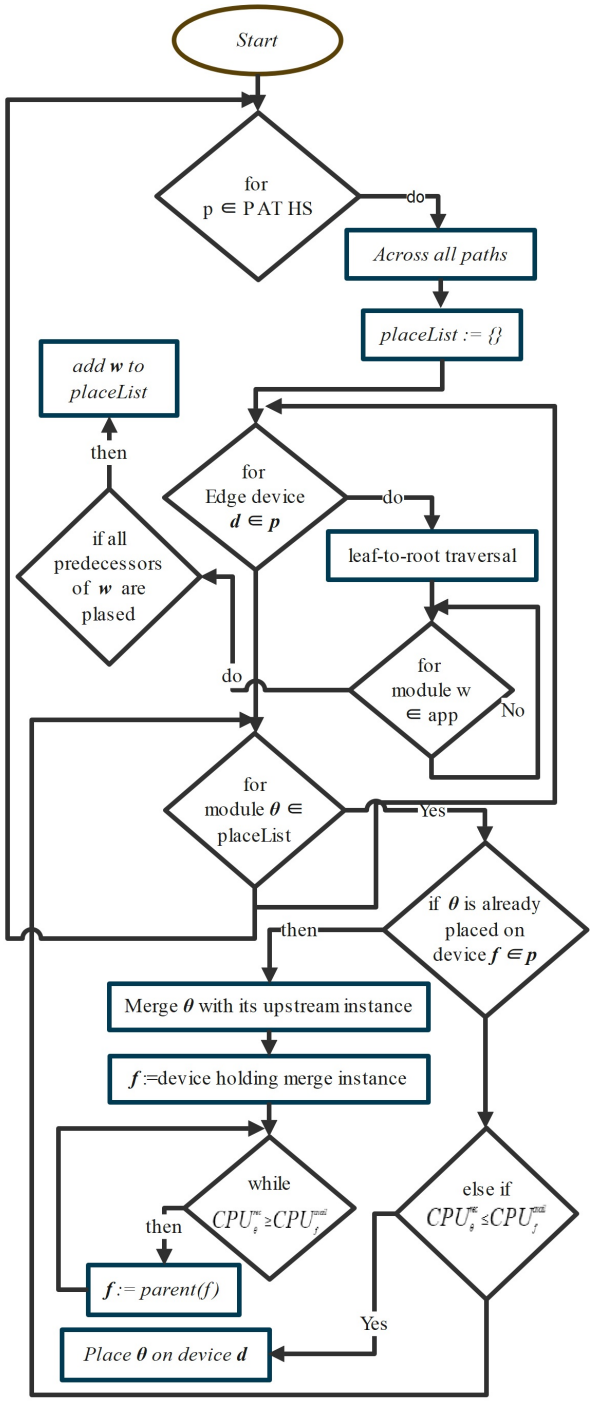



Fig. 5: Flowchart of module placement algorithm

## V. System Configurations for Simulations

To get unbiased results of research, an unbiased system configuration is mandatory. In this study, the iFogSim simulation toolkit is used to get the outcomes. It provides customized experiments, resource management, and effortless simulation [35]. For simulation, we have used Intel Core i9 12th-generation processor with 44GB of RAM.

In system configuration tables various terms have been used. Where *Level* means the hierarchical position of the components, level-0 represents the topmost level, level-1 is the middle layer and, level-2 is the bottom layer. *RAM* represents the physical memory of the system. *MIPS* represents the CPU processing capability. *RatePerMIPS* represents the cost per MIPS. *BusyPower* and *IdlePower* represent the energy consumption in working mode and idle mode [35].

The network latency between the components of the physical topology is shown in TABLE IX. Physical topology represents the network configurations used for simulation. Gigabit LAN is installed between the *Intelligent Camera* and the *MEC server* as a communication link in each topology.

### A. MEC Configurations

In this article, the MEC-based proposed model contains a cloud module, proxy module, MEC server module, and intelligent cameras. The intelligent cameras are connected to MEC servers via some proxy servers. The task is offloaded to the MEC server from the device layer. After completion of the task, it is uploaded to the cloud via a router for future use. TABLE IV and TABLE V show the system configurations of the could and proxy module in the MEC-based placements. TABLE VI shows the system configurations of the MEC server itself.

TABLE III: Intelligent Camera configurations

| CPU Length | NW Length | Interval Time |
|---|---|---|
| 1200 million instructions | 24000 byte | 5 milliseconds |

TABLE IV: Configurations of Cloud module in MCE placement

| Parameter | Cloud (Datacenter) |
|---|---|
| Hierarchical Level | 0 |
| Size of RAM | 80000MB |
| Processing Capability | 44800MIPS |
| RatePerMIPS | 0.01 |
| BusyPower Consumption | 16 × 106.665Watt |
| IdlePower Consumption | 16 × 83.50Watt |
| UploadLink | 100MB |
| DownloadLink | 10000MB |

TABLE V: Configurations of Proxy module in MEC placement

| Parameter | Proxy Server |
|---|---|
| Hierarchical Level | 1 |
| Size of RAM | 4800MB |
| Processing Capability | 3200MIPS |
| RatePerMIPS | 0.00 |
| BusyPower Consumption | 16 × 107.339Watt |
| IdlePower Consumption | 16 × 83.4333Watt |
| UploadLink | 10000MB |
| DownloadLink | 10000MB |

TABLE VI: Configurations of MEC server module in MEC placement

| Parameter | MEC Server |
|---|---|
| Hierarchical Level | 2 |
| Size of RAM | 4800MB |
| Processing Capability | 3200MIPS |
| RatePerMIPS | 0.00 |
| BusyPower Consumption | 16 × 107.339Watt |
| IdlePower Consumption | 16 × 83.433Watt |
| DownloadLink | 10000MB |
| UploadLink | 10000MB |

TABLE VII: Configurations of Cloud module in MCC placement

| Parameter | Cloud (Datacenter) |
|---|---|
| Hierarchical Level | 0 |
| Size of RAM | 80000MB |
| Processing Capability | 44800MIPS |
| RatePerMIPS | 0.01 |
| BusyPower Consumption | 16 × 106.665Watt |
| IdlePower Consumption | 16 × 83.50Watt |
| UploadLink | 100MB |
| DownloadLink | 10000MB |

### B. MCC Configurations

The MCC model contains a cloud module and a router module. Due to the absence of the MEC server in the middle layer, the tasks are fully offloaded to the MCC layer for computation. TABLE VII and TABLE VIII show the configurations of the cloud module and router module in the MCC environment.

## VI. Result and Analysis

For simulations, four sets of configurations have been considered. Config-1 contains 16 intelligent cameras, Config-2 contains 32 intelligent cameras, Config-3 contains 48 intelligent cameras, and Config-4 contains 64 intelligent cameras. Each camera produces the same amount of data every 5 milliseconds (see TABLE III). The simulation results are shown in TABLE X.

TABLE VIII: Configurations of Router module in MCC placement

| Parameter | Router |
|---|---|
| Hierarchical Level | 1 |
| Size of RAM | 4000MB |
| Processing Capability | 3200MIPS |
| RatePerMIPS | 0.00 |
| BusyPower Consumption | 16 × 107.339Watt |
| IdlePower Consumption | 16 × 83.4333Watt |
| UploadLink | 10000MB |
| DownloadLink | 10000MB |

TABLE IX: Network description

| Source | Destination | Latency(milliseconds) |
|---|---|---|
| Intelligent Camera | MEC server | 2 |
| MEC server | Proxy server | 2 |
| Intelligent Camera | Router | 2 |
| Proxy server | Router | 4 |
| Router | MCC server | 100 |

### A. Cost matrices

Fig. 6a shows the execution cost in MEC and MCC environments. The following formulas are used to calculate the total cost:

$$E_{cost} = unit \times cost\ per\ unit \tag{6}$$

$$T_{cost} = E_{cost} + (T_{MIPS} \times L_u \times R_{MIPS} \times L_{times} \times C_i) \tag{7}$$

where, $T_{cost}$ represents total cost, $E_{cost}$ denotes execution cost, $C_i$ denotes the clock of iFogSim simulator. $T_{MIPS}$ represents the total MIPS of the MEC server. The last utilization and last utilization update time are represented by $L_u$ and $L_{times}$ respectively. Fig. 6a demonstrates the MEC-based system is more cost-effective than the MCC system.

### B. Latency

Fig. 6b shows the overall latency of the systems in MEC and MCC environments. The overall latency of the proposed framework is calculated as:

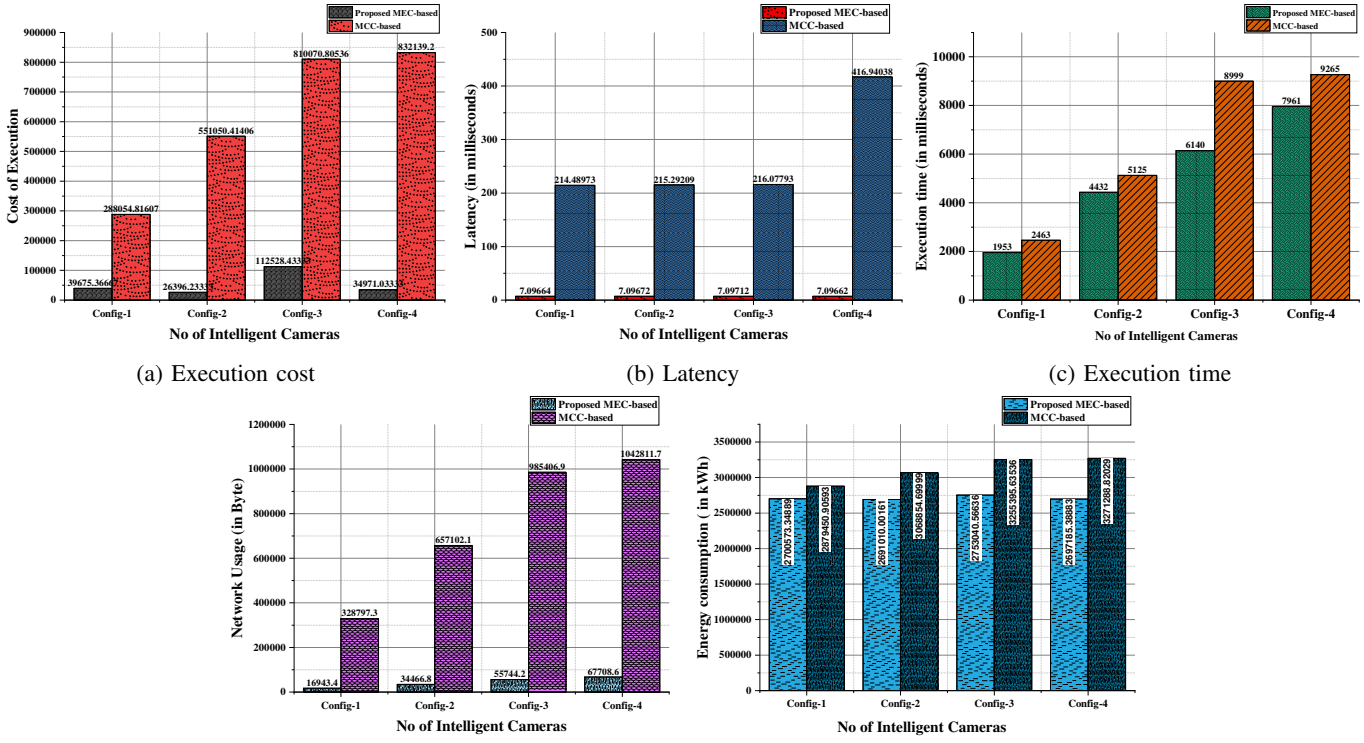
$$L_t = \alpha + \upsilon + \varphi + \lambda \tag{8}$$

Where $L_t$ denotes total latency. $\alpha$ denotes the latency of interested region detection from the raw video footage. $\upsilon$ defines the latency for object tracking. Similarly, $\varphi$ denotes the latency of decision-making. And finally, $\lambda$ defines the latency of camera feedback. This latency completely depends on the deployment of the module in the simulator. The result shows that the system consistently maintains low latency in the MEC environment.

### C. Execution time

Fig. 6c portrays the execution time in MEC and MCC environments. Execution time is a summation of all execution times of all the module applications. It contains the processing time and latency between source (sensor) to destination (actuator) devices. The execution time is calculated as:

$$X_t = E - S \tag{9}$$




(a) Execution cost (b) Latency (c) Execution time

(d) Network usage (e) Energy consumption

Fig. 6: Comparison analysis

TABLE X: Simulation Results

| Parameters ⟶ | Cost of Execution | | Latency (milliseconds) | | Execution time (milliseconds) | | Network usage (Bytes) | | Energy consumption (kWh) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MEC | MCC | MEC | MCC | MEC | MCC | MEC | MCC | MEC | MCC |
| *Config.-1* | 39675.37 | 288054.8161 | 7.096644 | 214.4897267 | 1953 | 2463 | 16943.4 | 328797.3 | 2700573 | 2879451 |
| *Config.-2* | 26396.23 | 551050.4141 | 7.096725 | 215.29209 | 4432 | 5125 | 34466.8 | 657102.1 | 2691010 | 3068855 |
| *Config.-3* | 112528.4 | 810070.8054 | 7.097117 | 216.077933 | 6140 | 8999 | 55744.2 | 985406.9 | 2753041 | 3255396 |
| *Config.-4* | 34971.03 | 832139.200 | 7.096624 | 416.940381 | 7961 | 9265 | 67708.6 | 1042812 | 2697185 | 3271289 |

Where $X_t$ denotes execution time, $E$ denotes task end time and $S$ represents task start time in the simulator. Fig. 6c clearly shows that the MEC system requires a shorter execution time than the MCC system.

### D. Network usages

Network usage is the utilization of resources in the system in terms of sending and receiving data through network interfaces.*NetworkUsageMonitor* class is used to calculate network usages using the formula:

$$N_u = L \times \psi \tag{10}$$

In equation (10) the $N_u$ denotes the entire network usage by the system in MEC and MCC-based environments. $L$ represents latency. Where $\psi$ denotes the *toupleNeWSize* in the iFogSim simulator. Fig. 6d shows that the higher workload will require higher system resources. The result clearly shows that the network usage in the MEC environment is relatively lower than the MCC in different configurations.

### E. Energy consumption

The energy consumption of each MEC server is calculated by the power of all hosts in the specific time quantum. In the iFogSim simulator, a method called *UpdateEnergyConsumption* is used to calculate total energy consumption. That method is used for MEC and MCC in both cases. Fig. 6e shows the energy consumption in MEC-based placements is comparatively lower than in MCC-based placements. Overall, these results indicate that the proposed task offloading framework provides cost-effective and energy-efficient ISS in the MEC environment.

## VII. Conclusion

This study set out to develop an energy-efficient and cost-effective framework for ISS in the MEC environment. The second aim of this study was to make a comparative study of the proposed MEC-based framework with MCC based framework. The research has found that the amalgamation of binary and partial task offloading policy in the MEC environment is more cost-effective and energy-efficient than in the MCC environment. The findings of this study confirm

that the proposed framework provides impressive results on latency, cost of execution, network usage, execution time, and energy consumption. The primary limitation of this research lies in its implementation within simulation environments. Furthermore, the incorporation of stochastic task offloading is absent in this study. Thirdly, the assessment of the efficacy of object detection and tracking algorithms is not undertaken. Subsequent research endeavors may delve into the exploration of module placement strategies and the formulation of energy-aware resource management policies, aiming to enhance the energy efficiency and cost-effectiveness of intelligent surveillance systems.

## ACKNOWLEDGMENT

This work was supported by the National Science Foundation, under award number 2219741. Mohammed AL-Numay acknowledges financial support from the Researchers Supporting Project Number (RSP2024R150), King Saud University, Riyadh, Saudi Arabia.

## REFERENCES

[1] S. W. Ibrahim, "A comprehensive review on intelligent surveillance systems," *Communications in science and technology*, vol. 1, 2016.

[2] W. Q. Yan, *Introduction to intelligent surveillance: surveillance data capture, transmission, and analytics*. Springer, 2019.

[3] S. E. Whang, D. Marmaros, and H. Garcia-Molina, "Pay-as-you-go entity resolution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1111–1124, 2012.

[4] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *2017 Global Internet of Things Summit (GIoTS)*. IEEE, 2017, pp. 1–6.

[5] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2017.

[6] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[7] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, vol. 29, pp. 854–864, 2014.

[8] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE communications surveys & tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[9] S.-H. Wang, D. R. Nayak, D. S. Guttery, X. Zhang, and Y.-D. Zhang, "Covid-19 classification by ccshnet with deep fusion using transfer learning and discriminant correlation analysis," *Information Fusion*, vol. 68, pp. 131–148, 2021.

[10] M. K. Mondal, R. Mandal, S. Banerjee, U. Biswas, P. Chatterjee, and W. Alnumay, "A cps based social distancing measuring model using edge and fog computing," *Computer Communications*, vol. 194, pp. 378–386, 2022.

[11] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, pp. 84–91, 2016.

[12] Q. He, Y. Wang, X. Wang, W. Xu, F. Li, K. Yang, and L. Ma, "Routing optimization with deep reinforcement learning in knowledge defined networking," *IEEE Transactions on Mobile Computing*, 2023.

[13] M. K. Mondal, R. Mandal, S. Banerjee, U. Biswas, J. C.-W. Lin, O. Alfarraj, and A. Tolba, "Design and development of a fog-assisted elephant corridor over a railway track," *Sustainability*, vol. 15, no. 7, p. 5944, 2023.

[14] Y. Cai, H. Zhang, Y. Liu, and Q. He, "Distributed bipartite finite-time event-triggered output consensus for heterogeneous linear multi-agent systems under directed signed communication topology," *Applied Mathematics and Computation*, vol. 378, p. 125162, 2020.

[15] C.-C. Lin, P. Liu, and J.-J. Wu, "Energy-efficient virtual machine provision algorithms for cloud systems," in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. IEEE, 2011, pp. 81–88.

[16] R. Mandal, M. K. Mondal, S. Banerjee, and U. Biswas, "An approach toward design and development of an energy-aware vm selection policy with improved sla violation in the domain of green cloud computing," *The Journal of Supercomputing*, vol. 76, pp. 7374–7393, 2020.

[17] Q. He, H. Fang, J. Zhang, and X. Wang, "Dynamic opinion maximization in social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 350–361, 2021.

[18] D. Das, S. Banerjee, P. Chatterjee, U. Ghosh, and U. Biswas, "A secure blockchain enabled v2v communication system using smart contracts," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4651–4660, 2022.

[19] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[20] D. Das, S. Banerjee, P. Chatterjee, U. Ghosh, and U. Biswas, "Blockchain for intelligent transportation systems: Applications, challenges, and opportunities," *IEEE Internet of Things Journal*, 2023.

[21] Q. He, Z. Feng, H. Fang, X. Wang, L. Zhao, Y. Yao, and K. Yu, "A blockchain-based scheme for secure data offloading in healthcare with deep reinforcement learning," *IEEE/ACM Transactions on Networking*, 2023.

[22] Y.-D. Zhang, Z. Dong, S.-H. Wang, X. Yu, X. Yao, Q. Zhou, H. Hu, M. Li, C. Jiménez-Mesa, J. Ramirez *et al.*, "Advances in multimodal data fusion in neuroimaging: Overview, challenges, and novel orientation," *Information Fusion*, vol. 64, pp. 149–187, 2020.

[23] S. Safavat, N. N. Sapavath, and D. B. Rawat, "Recent advances in mobile edge computing and content caching," *Digital Communications and Networks*, vol. 6, no. 2, pp. 189–194, 2020.

[24] R. Mandal, M. K. Mondal, S. Banerjee, P. Chatterjee, W. Mansoor, and U. Biswas, "Pbv msp: A priority-based vm selection policy for vm consolidation in green cloud computing," in *2022 5th International Conference on Signal Processing and Information Security (ICSPIS)*. IEEE, 2022, pp. 32–37.

[25] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[26] Y. Zhang, L. Deng, H. Zhu, W. Wang, Z. Ren, Q. Zhou, S. Lu, S. Sun, Z. Zhu, J. M. Gorriz *et al.*, "Deep learning in food category recognition," *Information Fusion*, p. 101859, 2023.

[27] Q. He, X. Wang, Z. Lei, M. Huang, Y. Cai, and L. Ma, "Tifim: A two-stage iterative framework for influence maximization in social networks," *Applied Mathematics and Computation*, vol. 354, pp. 338–352, 2019.

[28] S. E. Mahmoodi, R. Uma, and K. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 301–313, 2016.

[29] M. K. Mondal, R. Mandal, S. Banerjee, M. Sanyal, U. Ghosh, and U. Biswas, "Fog assisted tiger alarming framework for saving endangered wild life," in *SoutheastCon 2023*. IEEE, 2023, pp. 798–803.

[30] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing." *HotCloud*, vol. 10, no. 4-4, p. 19, 2010.

[31] S. Melendez and M. P. McGarry, "Computation offloading decisions for reducing completion time," in *14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2017, pp. 160–164.

[32] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2014, pp. 352–357.

[33] J. Lee, B.-G. Nam, and H.-J. Yoo, "Dynamic voltage and frequency scaling (dvfs) scheme for multi-domains power management," in *2007 IEEE Asian Solid-State Circuits Conference*. IEEE, 2007, pp. 360–363.

[34] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, pp. 2322–2358, 2017.

[35] H. Gupta, A. Vahid 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, 2017.