drones

MDPI

*Article*

# Dynamic Offloading in Flying Fog Computing: Optimizing IoT Network Performance with Mobile Drones

**Wei Min [1,*], Abdukodir Khakimov [2], Abdelhamied A. Ateya [3,4], Mohammed ElAffendi [3], Ammar Muthanna [2], Ahmed A. Abd El-Latif [3,5] and Mohammed Saleh Ali Muthanna [6]**

1 China-Korea Belt and Road Joint Laboratory on Industrial Internet of Things, School of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
2 Department of Applied Probability and Informatics, Peoples' Friendship University of Russia (RUDN University), 117198 Moscow, Russia; khakimov-aa@rudn.ru (A.K.); muthanna.asa@spbgut.ru (A.M.)
3 EIAS Data Science Lab, College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia; aateya@psu.edu.sa (A.A.A.); affendi@psu.edu.sa (M.E.); aabdellatif@psu.edu.sa (A.A.A.E.-L.)
4 Department of Electronics and Communications Engineering, Zagazig University, Zagazig 44519, Egypt
5 Department of Mathematics and Computer Science, Faculty of Science, Menoufia University, Menouf 32511, Egypt
6 Institute of Computer Technologies and Information Security, Southern Federal University, 347922 Taganrog, Russia; muthanna@sfedu.ru
* Correspondence: weimin@cqupt.edu.cn

**Abstract:** The rapid growth of Internet of Things (IoT) devices and the increasing need for low-latency and high-throughput applications have led to the introduction of distributed edge computing. Flying fog computing is a promising solution that can be used to assist IoT networks. It leverages drones with computing capabilities (e.g., fog nodes), enabling data processing and storage closer to the network edge. This introduces various benefits to IoT networks compared to deploying traditional static edge computing paradigms, including coverage improvement, enabling dense deployment, and increasing availability and reliability. However, drones' dynamic and mobile nature poses significant challenges in task offloading decisions to optimize resource utilization and overall network performance. This work presents a novel offloading model based on dynamic programming explicitly tailored for flying fog-based IoT networks. The proposed algorithm aims to intelligently determine the optimal task assignment strategy by considering the mobility patterns of drones, the computational capacity of fog nodes, the communication constraints of the IoT devices, and the latency requirements. Extensive simulations and experiments were conducted to test the proposed approach. Our results revealed significant improvements in latency, availability, and the cost of resources.

**Keywords:** Internet of Things; flying fog; offloading; dynamic programming; drones



**Citation:** Min, W.; Khakimov, A.; Ateya, A.A.; ElAffendi, M.; Muthanna, A.; Abd El-Latif, A.A.; Muthanna, M.S.A. Dynamic Offloading in Flying Fog Computing: Optimizing IoT Network Performance with Mobile Drones. *Drones* **2023**, *7*, 622. https://doi.org/10.3390/drones7100622

Academic Editors: Hiroyuki Tomiyama, Ittetsu Taniguchi, Xiangbo Kong, Hiroki Nishikawa and Carlos Tavares Calafate

Received: 10 August 2023
Revised: 29 September 2023
Accepted: 2 October 2023
Published: 5 October 2023



## 1. Introduction

The Internet of Things (IoT) is a revolutionary concept that describes the interconnectivity of everyday objects and devices through the Internet, enabling them to collect, exchange, and process data without the need for direct human intervention [1]. The future of IoT is undeniably intertwined with the advent of fifth-generation cellular (5G) and upcoming sixth-generation (6G) technology. The rollout of 5G networks brings unparalleled data speeds, ultra-low latency, and increased capacity, significantly enhancing the capabilities of IoT devices and applications [2]. With 5G, IoT solutions can deliver real-time insights and responses, enabling faster data processing for critical applications like autonomous vehicles, smart cities, and augmented reality. Furthermore, 5G's network slicing capabilities allow operators to create dedicated segments for IoT services, ensuring efficient resource allocation and improved overall performance [3].

As we look ahead to the era of 6G, even more transformative opportunities for IoT devices emerge. Through virtually eliminating latency and accommodating an unprecedented number of connected devices, 6G is expected to push the boundaries of data speeds to terabits per second [4]. These advancements will unlock the potential for highly immersive IoT applications, such as massive-scale augmented reality experiences, remote robotic control with minimal delay, and the real-time monitoring of vast and complex systems [5]. The convergence of IoT with 5G and 6G technologies will propel the growth of smart cities, precision agriculture, industrial automation, and personalized healthcare, further blurring the lines between the physical and digital worlds [6].

IoT holds immense promise for the future, and its potential is further magnified in the era of 5G and 6G technologies. These next-generation networks will empower IoT applications with unprecedented speed, responsiveness, and capacity, revolutionizing industries and enhancing overall quality of life. Nevertheless, addressing the design challenges associated with security, interoperability, resource efficiency, and scalability is crucial to harness IoT's full potential and drive its continued growth and success in the increasingly connected world [2,7].

Despite the promise of IoT, the deployment and management of large-scale IoT networks present several significant design challenges. One of the primary concerns is ensuring 5G/6G coverage and the availability requirements of IoT networks. Also, ensuring dense deployment at the latency requirements is another significant challenge. Moreover, IoT devices often operate with limited resources, particularly in terms of power and computational capacity. Balancing functionality with energy efficiency becomes a delicate trade-off, especially for battery-operated devices or those deployed in remote and hard-to-reach locations [8].

IoT networks must address the scalability and management of a vast number of connected devices. As the number of IoT devices grows exponentially, network infrastructures must accommodate for the surge in data traffic and processing demands [9]. Effective device management, network monitoring, and predictive maintenance strategies are required to ensure the reliability, availability, and performance of IoT networks as they continue to expand and evolve [10].

Edge computing is a promising paradigm to overcome these issues and empower IoT networks with coverage and resources [11]. Fog computing extends the cloud computing model, where computing capabilities are distributed closer to the network's edge rather than relying solely on centralized cloud servers. This approach drastically reduces latency, minimizes bandwidth usage, and enhances the overall performance of IoT devices by processing critical data at the network edge itself [12]. However, deploying static fog servers has its limitations, especially in scenarios where constant mobility and flexibility are required. Moreover, static fog servers have network coverage and scalability limitations [13].

Flying fog computing is a novel paradigm that empowers fog computing with a level of mobility. By incorporating fog servers into unmanned aerial vehicles (UAVs) such as drones, it leverages the advantages of both fog computing and drone technologies to offer unprecedented benefits [14]. These fog-enabled drones can be deployed rapidly to specific locations, fly directly to where data is generated, and provide on-demand processing capabilities to nearby IoT devices. As a result, they empower IoT networks with enhanced real-time data analytics, low-latency responses, and improved data privacy, all while ensuring a higher degree of fault tolerance [15,16].

Flying fog, i.e., fog servers on drones, can assist IoT networks in many ways [15–17].

a. Edge data processing: Flying fog computing moves data processing tasks closer to the source, which is particularly advantageous when real-time analysis is crucial. Drones equipped with fog servers can process data on-site, reducing the need to transmit raw data over long distances, thus conserving bandwidth and reducing latency.
b. Scalability and flexibility: The mobility of drones enables dynamic scalability, allowing fog servers to be redeployed and allocated based on the changing demands of the

IoT network. Drones can adjust their routes and position autonomously, ensuring seamless adaptability to varying data loads and network requirements.

c. Reliable connectivity in remote and harsh areas: Conventional IoT networks may face connectivity issues in remote or challenging terrains. Flying fog computing overcomes this limitation by flying fog-enabled drones to these locations, creating temporary communication hotspots and ensuring uninterrupted data transmission and processing.

d. Data privacy and security: Since data remains within the local network and is processed in the drone, in proximity to devices, the risk of sensitive information being exposed to external threats is significantly reduced. This added layer of security bolsters the confidence of users and organizations in adopting IoT solutions.

e. Disaster response and emergencies: In disaster-stricken areas, where infrastructure may be compromised or non-existent, fog-enabled drones can rapidly deploy and establish a communication and data processing network, facilitating critical information exchange and aiding rescue efforts.

Flying fog computing represents an innovative and dynamic approach to revolutionizing the capabilities of IoT networks [14]. By integrating fog servers into drones, this visionary technology brings the advantages of fog computing to even the most remote and challenging environments, facilitating efficient data processing, real-time analytics, and robust connectivity [14,15]. As this nascent field continues to advance, it promises to shape the future of IoT networks and pave the way for a more interconnected and responsive world. However, the dynamic and mobile nature of drones poses significant challenges in task offloading decisions to optimize resource utilization and overall network performance. This work aims to develop a dynamic, optimized offloading scheme for flying fog-based IoT networks. The contributions of this work are as follows:

1. Designing a general framework for flying fog-based IoT networks.
2. Developing a computational model of the proposed flying fog-based IoT networks.
3. Developing an optimized dynamic offloading strategy for flying fog-based IoT networks.
4. Developing an evaluation testbed that is used to emulate flying fog-based IoT networks.
5. Evaluating the proposed offloading model via simulations and experiments.

The remainder of this work is structured as follows: In Section 2, we delve into related studies and pertinent works. In Section 3, we introduce our proposed IoT model, which is based on flying fog technology. Section 4 outlines both the proposed computation model and algorithm. In Section 5, we discuss the testbed development for system evaluation. Finally, Section 6 presents the results we obtained using the proposed model.

## 2. Related Work

Fog computing is a promising paradigm that addresses the limitations of cloud-centric IoT networks, particularly in scenarios with low-latency and real-time processing requirements. Bonomi et al. [18] introduced the concept of fog computing, highlighting its benefits in reducing data transfer latency and enhancing overall system responsiveness. Since then, several studies have explored the integration of fog computing with IoT networks to improve computation and storage capabilities at the network edge [19,20]. However, most of the existing approaches assume static fog nodes, and there is a need for dynamic solutions to cater to the unique challenges presented by mobile fog nodes in flying IoT networks.

Computation offloading is a critical technique in IoT-based fog computing environments; it is used to balance computational workloads and optimize resource utilization. The development of offloading algorithms for fog computing-based IoT networks has gained significant attention in recent years. Researchers have explored various approaches to efficiently manage resources and improve the overall performance of such networks [21,22]. This section presents an overview of the most relevant studies and approaches that have been proposed to address the challenges of resource offloading in the context of fog computing-based IoT networks.

Chen et al. [23] investigated the multi-user multi-server offloading problem. The objective was to minimize user costs and maximize edge server profits. To address this, the authors proposed an efficient offloading and resource purchasing strategy involving a joint optimization methodology of two stages. The first stage utilizes a multi-channel access game (MCA-Game) to find Nash equilibrium, employing the MCA-Game algorithm. The second stage uses Stackelberg game theory for resource allocation. The proposed game-based pricing and purchasing (GPAP) method was shown to be incentive-compatible with the existence of Stackelberg equilibrium. Despite the work providing novel and well-evaluated offloading criteria, it did not consider the mobility of edge nodes. Moreover, latency is the main concern in our proposed approach since we target enabling the uRLLC.

IoT networks suffer from bandwidth overhead, increased delays, resource management issues, and reduced system throughput. To address these issues and enhance the quality of service (QoS), Wadhwa et al. [24] presented an optimized task scheduling and preemption (OSCAR) approach for fog-enabled IoT networks. The OSCAR model involved several steps, including task clustering, scheduling the clustered tasks, and resource allocation. The clustering of IoT tasks can be achieved using an expectation–maximization (EM) approach. This was introduced to reduce overhead complexity and bandwidth. The proposed framework ultimately distributes resources using a deep Q network model. IoT tasks are performed through task preemption using a ranking approach, where higher-priority tasks with shorter deadlines replace lower-priority tasks, moving them to a waiting queue. The OSCAR approach was evaluated using the iFogSim platform, and the results showed its superiority over existing models in achieving QoS. This work considered fixed fog computing servers.

Offloading policies can be centralized or distributed. However, with the growth of IoT entities and the expansion of the fog computing layer, centralized scheduling faces challenges of complexity and lack of fault tolerance. Ataie et al. [25] addressed these issues and developed a scalable decentralized approach where a small number of nodes collaborate autonomously to schedule tasks. The authors proposed a scalable technique for offloading time-critical jobs through a semi-network-aware distributed scheduling method. The results showed that, on average, the proposed method outperformed existing approaches.

Offloading strategies face challenges in reducing delays due to the resource constraints of fog nodes and the imbalance of workload distribution caused by plenty of work requests and demanding workloads. To tackle these issues, Tran-Dang et al. [26] presented a dynamic collaborative task offloading (DCTO) method that considers the fog nodes' resource status in order to arrive at an offloading approach. This approach allows tasks to be performed by single or multiple fog nodes, effectively reducing execution latency. Their results demonstrated that the offloading approach significantly reduced latency at high service request rates. Furthermore, the model's low computational complexity enables online implementation, distinguishing it from other algorithms.

Dynamic programming has proven to be a powerful technique for optimization problems in various domains. Shahzad et al. [27] applied dynamic programming to optimize computation offloading decisions for stationary edge nodes. Shahzad et al. [27] proposed a new computational offloading approach using dynamic programming with Hamming distance termination (DPH) to deal with the limited battery power of mobile devices. When the network bandwidth is high, DPH offloads as many tasks as possible to the cloud, enhancing task execution time and reducing the mobile device's energy usage. Moreover, DPH is scalable and can efficiently handle larger offloading problems. Their results showed that the DPH approach achieved minimum energy consumption.

Hybrid approaches that combine multiple optimization techniques have been explored in the context of resource offloading. Wang et al. [28] presented a hybrid algorithm that combined genetic algorithms and integer programming for the purposes of offloading decisions in static edge computing environments. The work enhanced user satisfaction, directly impacting the profitability and reputation of service providers. To achieve this, the authors addressed the task offloading issue within the context of edge–cloud cooper-

ative computing. The proposed model deployed an integer genetic algorithm. Through experiments, the authors demonstrated that the method yielded superior results regarding resource efficiency. While the study demonstrated enhanced resource allocation, it did not investigate the application of hybrid approaches for tackling the unique challenges of flying fog computing-based IoT networks.

Reddy et al. [29] presented a task offloading and scheduling method inspired by an osmotic process. This algorithm involved classifying devices and tasks and subsequently assigning tasks to the appropriate nodes according to their capacity. The proposed osmotic-based scheduling algorithm demonstrated significantly superior performance compared to traditional random and round-robin task offloading algorithms. The comparison was conducted using synthetic data sets, confirming the effectiveness of the proposed approach compared to other algorithms.

Lakhan et al. [30] focused on allocating fog resources in a fog-enabled software-defined network (SDN) consisting of multiple fog nodes. The problem involved meeting strict constraints like mobility, deadlines, and resource capacity while executing applications. The work proposed a container-based approach to enhance fog network performance by reducing latency and energy consumption. Within this approach, they introduced a deep-learning–Q-network approach comprising components like mobility control and resource searching for allocating resources. Their results demonstrated that the developed approach outperformed other approaches and models proposed by other studies, achieving a 30% improvement in application costs.

Fog empowers vehicular ad hoc network evolution; however, balancing the load among fog nodes in such systems is challenging. This devastates the availability of network services and the effectiveness with which resources are used. Thanedar et al. [31] considered this issue by presenting a dynamic resource management (DRM) method that used service migration between nodes to allocate fog resources to autonomous cars. The problem is solved in polynomial time (modeled as a graph). The suggested method considered a set of cars in the overlapping coverage regions of the fog nodes and established communication with the appropriate fog nodes. Extensive simulations were run on the DRM, and the results of these simulations revealed improved capacity, maintainability, accessibility, availability, and throughput.

Hosseini et al. [32] addressed task scheduling in mobile fog computing to handle resource limitations in mobile devices. The existing research on scheduling in fog nodes lacks an in-depth exploration of multiple criteria. The authors addressed this challenge by introducing a scheduling algorithm that combined priority queue and fuzzy logic. The proposed model considered various aspects, including delay, energy, RAM usage, and deadlines, and achieved better results than existing algorithms. The proposed algorithm significantly improved fog computing efficiency, reducing delay and energy consumption while enhancing the service level.

Flying fog is a promising paradigm for IoT networks. It is an emerging research area with potential advantages for extending the capabilities of IoT networks through mobile fog nodes (e.g., drones). While this technology holds promise, researchers have addressed several main problems and issues to make it more practical and effective. A summary of these issues can be seen below [15,16,33]:

a. Mobility and connectivity management: Managing the movement of flying fog nodes and ensuring seamless connectivity with ground-based IoT devices present challenges. Effective handover mechanisms and protocols are needed to maintain connectivity as the drones move.
b. Energy constraints: Flying fog nodes are typically powered by batteries with limited capacity. Prolonged flight durations and data processing tasks may drain energy quickly. Energy-efficient strategies for flight planning, task allocation, and communication are essential.

c. Resource allocation: Optimally allocating computing, storage, and communication resources among multiple flying fog units and IoT nodes is complex. Resource management algorithms must be designed to handle dynamic and heterogeneous environments.
d. Communication latency and bandwidth: Low latency is critical for real-time IoT applications. The limited wireless communication bandwidth between flying fog nodes and ground devices can impact data transmission and processing delays.
e. Security and privacy: Flying fog nodes introduce new security challenges due to their mobility and potential vulnerability to physical attacks. Ensuring data privacy, authentication, and secure communication between nodes is paramount.
f. Scalability: The network must scale efficiently due to the growth in IoT and flying fog devices. The architecture and protocols must be scalable to accommodate growing connected devices.
g. Fault tolerance and reliability: In dynamic environments, flying fog nodes may encounter various challenges, such as signal interference, adverse weather conditions, or mechanical failures. Mechanisms for fault tolerance and reliable operations are crucial.
h. Regulatory and legal considerations: Deploying flying fog nodes in urban areas may involve compliance with aviation regulations, privacy laws, and safety standards. Understanding and adhering to these requirements is necessary for practical implementations.
i. Cost and deployment challenges: The cost of flying fog nodes, their maintenance, and the logistics of deploying them at scale can be significant barriers to adoption. Developing cost-effective and easily deployable solutions is essential.
j. Integration with existing IoT ecosystems: Integrating flying fog technology with existing IoT infrastructure and protocols is complex. Compatibility and interoperability must be considered to leverage the full potential of both technologies.

This work addresses these challenges' first, second, and fourth issues due to the following reasons.

- These challenges are closely related to our major focus and work objectives. These specific challenges have a direct impact on our main problem and are an important component of our research area.
- Our review of the available literature and early investigations demonstrated that these three issues have significance to IoT applications and should be prioritized. By addressing these specific gaps, we may also give useful insights and solutions to the scientific community.
- Each of the chosen tasks has numerous issues that must be thoroughly investigated. We may provide a more extensive analysis and propose well-founded ways to address these difficulties effectively by reducing our focus.
- Addressing all possible difficulties within a single research article may result in a lack of practicality or answers that are too broad in some circumstances. We want to make recommendations that practitioners and researchers may easily put into action.

Few research works have considered flying fog for IoT, and none have considered proposing an optimized offloading model for this paradigm.

## 3. Proposed Flying Fog-Based IoT Model

The proposed model consists of a set of IoT nodes distributed over a geographical area. IoT nodes gather data from the surroundings and always have data to be processed. The proposed model deploys a set of flying nodes, i.e., micro-drones, to provide either a wireless interface or assist IoT nodes in processing their data. The deployed drones are organized into two main sets: relay drones and flying fog drones.

Relay drones are deployed to provide wireless interfaces to IoT nodes or other deployed drones. They act as relays to move data from IoT nodes to the IoT gateway if IoT nodes cannot reach the gateway directly. Also, relay drones transfer data between drones and the IoT gateway. Flying fog drones are drones with mounted fog servers. This set

of drones is deployed to provide computing resources and assist in processing the data of IoT nodes. This empowers the network with the various benefits of the distributed computing paradigm. Figure 1 presents the proposed model for a certain use, namely, a smart city with heterogeneous applications, where many of IoT gateways are deployed far away from the IoT nodes. Thus, in this scenario and similar scenarios, many nodes are out of coverage and need a link with the gateway. In this use case, drones carrying fog nodes fly over the intended regions to provide the required resources to IoT nodes. The significance of introducing flying fog nodes to such a use case can be clearly extracted for dense deployment-based applications. With massive deployment, even if the gateway is close to the IoT nodes, the need for computing resources is strong. Moreover, flying fog provides a high level of system flexibility, mainly in terms of computing, due to drone mobility.

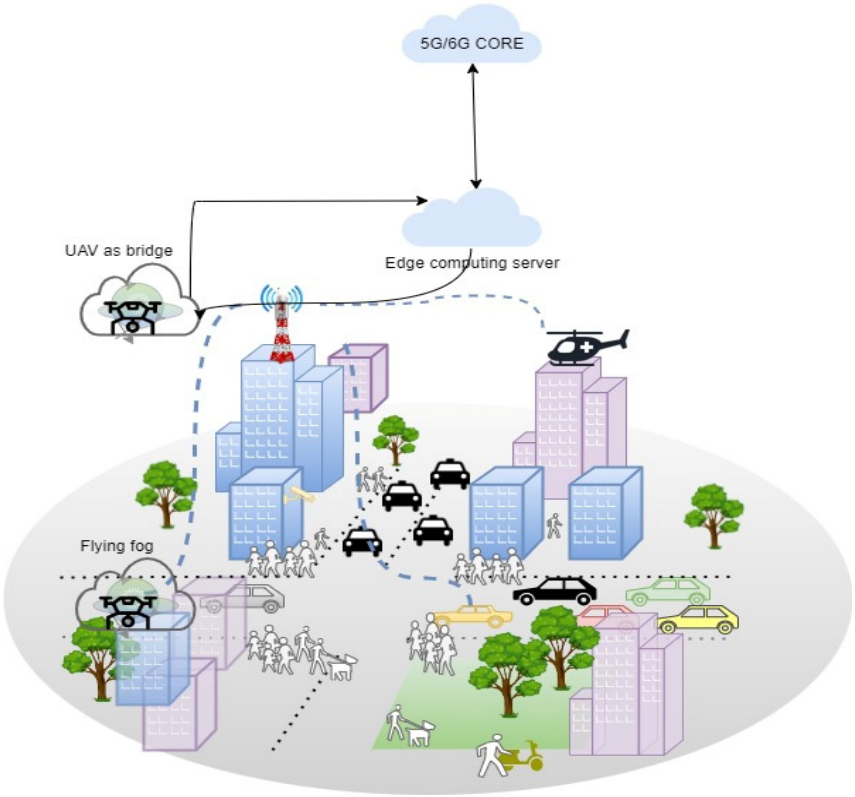


**Figure 1.** System model.

IoT nodes gather data and process it at a nearby flying fog node, multiple access edge (MEC) server at the gateway, or remotely at the central cloud server. For the first scenario, flying fog computing, the fog server in a drone is used for processing data. The fog node is equipped with computing resources and has the ability to store, process, and analyze data. The second scenario, MEC computing, involves processing data at the MEC server. In this case, data is processed at a farther distance, i.e., gateway. In each computing model, the IoT node offloads its data to the processing server either directly or via a relay point.

The proposed model provides a dynamic infrastructure with many use cases, mainly for dense IoT-based applications in various domains. This includes the following applications [34,35]:

a. Smart city applications: Flying fog drones can serve as edge nodes to assist smart cities. They can support many applications, including traffic management, surveillance, public Internet coverage, and public events, enhancing the efficiency and responsiveness of urban infrastructure.
b. Agriculture applications: Flying fog can assist IoT nodes deployed to monitor and analyze crop health, soil conditions, and weather patterns. They can provide precision

agriculture solutions, including targeted irrigation, pest control, and fertilizer application. Also, flying fog-based IoT systems can identify and target areas affected by pests, delivering precise pest control measures while minimizing the use of chemicals. This approach is environmentally friendly and cost-effective.

c. Environmental monitoring: Flying fog can be used with IoT nodes deployed to collect environmental data, such as air quality, water quality, and temperature, in remote or inaccessible locations. These data are valuable for environmental research and monitoring.

d. Industrial IoT (IIoT) applications: Flying fog nodes can be employed in industrial settings to monitor equipment health, perform inspections, and collect data on manufacturing processes. This helps in predictive maintenance, reducing downtime, and improving efficiency.

The set of flying fog drones is $D_f$, defined in Equation (1), where $N_f$ is the total number of fog drones used in the system. The set of relay drones is $D_r$ and is defined in Equation (2), where $N_r$ is the total number of used relay drones.

$$D_f = \left\{D_1,\ D_2,\ D_3, \cdots, D_{Nf}\right\} \qquad \forall\, N_f \in \mathbb{R},\ N_f < N, D_f \subset D, \tag{1}$$

$$D_r = \{D_1,\ D_2,\ D_3, \cdots, D_{Nr}\} \qquad \forall\, N_r \in \mathbb{R},\ N_r < N,\ D_r \subset D, \tag{2}$$

$$D = D_r \cup D_f, \tag{3}$$

where $D$ is the set of drones used in the systems, and $N$ is the size of the set, i.e., the total number of system drones. IoT nodes are assumed to have dynamic workloads that need to be processed, where workloads of devices are independent.

To accomplish a task, the data will be offloaded and processed either at the fog server when a binary computation offloading decision $\alpha_i$ is "1" or at the *MEC* server when a binary offloading decision $\alpha_i$ is "0". When the task is transferred to *MEC*, it goes via a direct interface or a relay drone.

$$\alpha_i = \begin{cases} 1 & offload\ to\ flying\ fog \\ 0 & offload\ to\ gateway\ MEC' \end{cases} \tag{4}$$

The calculation of the highest data transmission rate achievable for the computation task over the wireless channel between the IoT node and the relay drone is as follows.

$$R_{IoT,drone} = w * log_2\Big(1 + \frac{\varepsilon_{IoT,drone}\eta_{IoT,drone}}{\sigma w}\Big), \tag{5}$$

where $w$ represents the bandwidth of the wireless link, $\eta_{IoT,drone}$ is the gain of the channel between *IoT* node and drone, $\varepsilon_{IoT,drone}$ is the power rate of drone and *IoT* nodes, and $\sigma$ is the noise power.

## 4. Computation Model

This section presents the computation model used by the proposed network. IoT nodes possess K separate computation tasks that require execution. Each computation task i is described by a tuple $\{S_i, C_i, T_i^{constraint}\}$, where $S_i$ denotes the workload size, $C_i$ refers to the CPU cycles needed for the workload of task (i), and $T_i^{constraint}$ indicates the necessary time for the task i's completion.

### *4.1. Flying Fog Computing*

In the scenario, IoT nodes search for available resources at flying fogs to determine whether they can handle the task. The time taken to transmit a workload of task (*i*) from the IoT node to the flying fog *j* can be formulated as follows:

$$T_{IoT-fog-j}^{trans-Ti} = \frac{S_i}{R_{IoT,drone}} \ \ \forall\, i, j \in \mathbb{R}, j \le N_f\ , \tag{6}$$

This transmission time mainly depends on the transmission rate and the task size. The time required to process the *i*th task in the *j*th flying fog is formulated as follows:

$$T_{drone-fog-j}^{process-Ti} = \frac{C_i}{\omega_{fog}} \quad \forall\, i, j \in \mathbb{R}, j \leq N_f, \tag{7}$$

where $\omega_{fog}$ is the set of computational capabilities of the flying fog $j$. This time is educed if the current processing resources are high. The total latency for processing the workload of the *i*th task at the *j*th flying fog can then be calculated as follows:

$$T_{drone-fog-j}^{total-Ti} = T_{IoT-fog-j}^{trans-Ti} + T_{drone-fog-j}^{process-Ti}, \tag{8}$$

$$T_{drone-fog-j}^{total-Ti} = \frac{S_i}{R_{IoT,drone}} + \frac{C_i}{\omega_{fog}} \quad \forall\, i, j \in \mathbb{R}, j \leq N_f, \tag{9}$$

*4.2. MEC Computing*

IoT nodes transfer their data to the MEC server at the gateway for edge–cloud server computing. This data transfer is performed via a direct link or through a relay drone. The time taken to transmit a workload of task ($i$) from the $j$th drone bridge to the MEC server is calculated as follows:

$$T_{fog-j-MEC}^{trans-Ti} = \frac{S_i}{R_{drone,\ \ MEC}} \quad \forall\, i, j \in \mathbb{R}, j \leq N_f \ , \tag{10}$$

where $R_{drone,\,MEC}$ is the achievable rate between the drone and the MEC server. For a communication interface with a large transmission rate, the delay for transmission between the drone and IoT node is reduced. The processing delay of task ($i$) at the MEC server is calculated as follows based on the MEC allocated processing resources:

$$T_{server}^{process-Ti} = \frac{C_i}{\omega_{server}} \ , \tag{11}$$

where $\omega_{server}$ is the set of processing resources of the MEC. For latency-sensitive applications, this time should be reduced by allocating more resources. The total delay for processing the workload of the *i*th task at the MEC can then be calculated as follows:

$$T_{server}^{total-Ti} = T_{IoT-fog-j}^{trans-Ti} + T_{fog-j-MEC}^{trans-Ti} + T_{server}^{process-Ti} \quad \forall\, i, j \in \mathbb{R}, j \leq N_f, \tag{12}$$

Due to the completion of the *i*th task, the communication overhead is calculated as follows:

$$T_i^{total-overhead} = \alpha_i T_{drone-fog-j}^{total-Ti} + (1 - \alpha_i) T_{server}^{total-Ti}, \tag{13}$$

*4.3. Problem Statement*

Addressing delay challenges in flying fog computing is critical to enhance IoT network performance and support ultra-low latency communications. We consider optimizing offloading strategy to minimize the total delay in task completion. This constrained optimization problem aims to intelligently allocate computational tasks to appropriate fog nodes or drones while satisfying specific criteria and limitations. The optimization process seeks to minimize latency and ensure reliable communication between IoT and fog nodes. The problem is formulated as follows:

$$min \sum_{i=1}^{N} T_i^{total}, \tag{14}$$

**S.T.**

$$\textbf{C1: } T_i^{total} \leq T_i^{constraint}, \tag{15}$$

$$\textbf{C2: } \alpha_i \in \{0, 1\}, \tag{16}$$

$$\textbf{C3: } \omega_{server} \leq \omega_{server-max}, \tag{17}$$

$$\textbf{C4: } \omega_{fog} \leq \omega_{fog-max} \tag{18}$$

The problem aims to reduce the combined delay (weighted by specific factors) by optimizing the distribution of tasks through offloading strategies. To achieve this goal, the study introduces four essential constraints. Firstly, constraint C1 sets upper limits on the time consumption for the offloading process. By imposing these bounds, we ensure that the total time taken for task offloading remains within acceptable limits. This constraint plays a crucial role in maintaining the overall efficiency and responsiveness of the system.

Secondly, constraint C2 is designed to ensure that the decisions regarding task offloading are binary. In other words, tasks can either be completely offloaded to the fog, MEC, or entirely executed on the IoT devices themselves. This binary characteristic of the offloading decisions contributes to simplifying the optimization process and aids in achieving more practical and feasible solutions. The other two constraints ensure working under the maximum capacity of fog and the MEC server's resources. The third constraint ensures that the processing status of the MEC unit is less than the full processing capacity. Also, the fourth constraint ensures that the flying fog nodes unload their processors by using processors below their maximum capacities. By addressing these constraints, we seek to devise an effective and efficient approach to task offloading, allowing for a reduction in delays in the network while maintaining resource utilization and system reliability at desirable levels.

### *4.4. Computational Offloading Algorithm*

The proposed computational model for the previously proposed flying fog-based IoT network is based on a dynamic programming technique and hamming distance termination [27]. In order to determine the best computation offloading choices for a flying fog computing system, the method provides an all-encompassing process. Initially, IoT nodes offload the $i$th workload to a nearby fling fog server when $\alpha_i = 1$ or to the MEC server when $\alpha_i = 0$. The developed approach is based on dynamic programming using a (K $\times$ K) table, which stores a bit stream. The stored bits indicate the execution server, which is where workloads will be executed, ensuring task processing at the nearby server (either fog or MEC).

The first cell in the table is always left empty, and the other cells are filled with the generated random bit streams as ones (1) in the upcoming horizontal unit and zeros (0) in the upcoming vertical unit. From the table, we can calculate the delay of each task by using each cell; 0 s for the flying fog computing case, and 1 s for the MEC server computing case. This choice is made due to the following reasons:

- Achieving a neutral initial state: Leaving the first cell empty signifies a neutral or undefined starting state for the dynamic offloading decision process. In dynamic offloading, there might not be a clear initial decision or offloading strategy until the algorithm begins evaluating the tasks and their characteristics.
- Random exploration: Filling subsequent cells with random bit streams can introduce an element of randomness or variability into the initial state. This randomness may be useful for exploring different offloading strategies or scenarios, especially when the optimal offloading decision is uncertain or context-dependent.
- Task dynamics: In dynamic offloading scenarios, the characteristics of computing tasks, network conditions, and device capabilities can change over time. Using random bit streams might simulate the evolving nature of these parameters, allowing the algorithm to adapt and make dynamic offloading decisions based on simulated changes.
- Algorithm flexibility: By initializing with randomness, the algorithm is not pushed toward any particular offloading decision at the start. This approach provides flexibility to adapt to various tasks and network conditions encountered during the dynamic offloading process.
- Algorithm exploration: Dynamic programming algorithms often involve exploring different states and transitions to find optimal solutions. The randomness in the

initialization can lead to diverse paths of exploration, helping the algorithm identify suitable offloading decisions.

The algorithm iteratively updates the dynamic programming table to identify the optimal offloading strategy as it considers various combinations of tasks, states, and transitions. The pseudo-code of the developed offloading model is presented in Algorithm 1.

**Algorithm 1.** The flying fog-based computation offloading algorithm using dynamic programming.

**Input data:** $T_i^{constraint}$, $R_{drone,\ MEC}$, and $R_{drone,\ IoT}$
**1.** InitializeTimeMatrices()
**2.** SetCompletionDeadline($T_i^{constraint}$)
**3.** SetTransmissionRate($R_{drone,\ MEC}$, $R_{drone,\ IoT}$)
**4.** GenerateRandomTask()
**5.** randomBitStream = GenerateRandomBitStream()
**6.** Drone_FogDelay = CalculateDelayAt-Drone-Fog(randomBitStream)
**7.** MECServerDelay = CalculateDelayAtMECServer(randomBitStream)
**8.** If (randomBitStream [0] == 1):
**9.** StartCell = Drone_FogDelay
**10.** Else
**11.** StartCell = MECServerDelay
**12.** End if
**13.** For (i = 0: K − 1)
**14.** If (randomBitStream[i] == 1):
**15.** randomBitStream[i] = GenerateRandomBit()
**16.** End if
**17.** PutBitInCorrectPosition(randomBitStream[i], Table)
**18.** If CellVisitedBefore (Table):
**19.** If (NewTotalDelay < PreviousTotalDelay):
**20.** ReplaceTotalDelayWithNew(Table)
**21.** CalculateRemainingBitStreamDelay(Table)
**22.** Else
**23.** KeepPreviousTotalDelay(Table)
**24.** CalculateRemainingCellDelay(Table)
**25.** End if
**26.** End if
**27.** End for
**28.** If ((NumberOfBitsInTable == N) & (Ttotal < $T\_i_{constraint}$) & (HammingDistanceCriterionMet)):
**29.** Return (Ttotal)
**30.** End if
**31.** End

## 5. Developed Testbed for System Evaluation

The Doppler effect occurs when relative motion between a source and an observer exists. It causes a change in the frequency or wavelength of the wave as perceived by the observer, moving towards or away from the source. It plays a significant role in wireless communication systems, especially mobile devices and fast-moving objects. Understanding and mitigating the effects of Doppler shifts is crucial for ensuring reliable and efficient communication [36].

Significant Doppler shift fluctuations are introduced when the drone is used as a relay, which impacts the system as follows [36,37]:

1. Frequency shift: The relative motion between the drone and IoT nodes or between a drone and another causes a Doppler shift in the received signal's frequency. This frequency shift can lead to a deviation from the expected frequency, which may result in signal distortions and errors.
2. Doppler spread: As drones/devices move, the Doppler effect affects different parts of the transmitted signal differently. Some parts may experience a positive frequency shift, while others may have a negative one. This spread of frequencies is referred to

as Doppler spread, which depends on the relative velocity between the transmitter and the receiver. This spread can cause frequency-selective fading and interference between symbols in the received signal, leading to inter-symbol interference (ISI) and impacting the data transmission reliability.

3. Fading and signal loss: The Doppler effect's frequency shifts can lead to fading, where the received signal strength fluctuates rapidly over time. This fading can cause a momentary loss of signal or reduced signal quality, leading to dropped calls, data packet errors, or reduced data rates in IoT systems.

Doppler shifts with time lag are most noticeable in drone-based scenarios. The rate and pattern of the channel's growth also change over time. This occurs for several reasons, the most significant of which are the Doppler shifts brought on by the interaction of drones and IoT nodes in the multipath channel. The delay and Doppler spectra in the channel are influenced by other factors, including the radiation pattern of the antenna and the dynamic changes in individual multipath propagation components (MPCs) [38].

The proposed flying fog-based IoT model should implement a compensation technique to overcome the challenges posed by the Doppler effect. The gain and phase shifts of individual MPCs are just two examples of channel properties that complex time-varying processes can describe. More MPC parameters need to be estimated for a more precise estimation of the channel model. The channel between mobile entities in a flying fog-based IoT system has a continuous function impulse response, denoted by $h(t)$, at which a signal is transmitted and processed. It is calculated as the aggregation of MPCs as follows [38]:

$$h(t,T) = \sum_{p=1}^{p} h_p(t)\delta\big(\tau - \tau_p(t)\big), \tag{19}$$

where $p$ represents the specific MPC, $T$ represents the transmission time of the signal, and $t$ represents the processing time of the signal. The impulse response of the beam p is the $h_p(t)$ component, which is calculated as follows:

$$h_p(t) = \eta_p \hat{\eta}_p(t) e^{j(2\pi v(t) + \phi(t))}, \tag{20}$$

where $\eta_p$ is the attenuation factor of the beam p, and $v(t)$ is the value of the Doppler effect. Then, the received signal $r(t)$ can be calculated as follows:

$$r(t) = s(t) \times h(t), \tag{21}$$

$$r(t) = \int_{-\infty}^{\infty} h(t, t-\tau)s(\tau)d(\tau) = \sum_{p=1}^{p} h_p(t)(s)(t-\tau) \;\forall\; t = mT,\; m = 1, 2, 3, \cdots \tag{22}$$

In this work, we deployed drones to expand the IoT network coverage and assist IoT nodes by providing computing resources. However, the work aims to maintain the QoS of all communications held over the proposed model. To evaluate flying fog-based terrestrial networks, including the considered flying fog-based IoT network, we modified our previously developed emulator (introduced in [38]).

The emulator implements the radio channel between drones and IoT nodes and compensates for the previously introduced channel effects. The emulator deploys software-defined radio (SDR) technology to achieve the following benefits [38,39]:

a. Flexibility and adaptability: SDR allows for flexible and adaptable radio communication systems. This flexibility enables easy reconfiguration and updates, making it possible to support multiple communication standards and protocols with the same hardware.
b. Reduced hardware complexity: SDR reduces the need for specialized hardware components for different radio applications. With SDR, much of the functionality is implemented in software, simplifying the hardware design and reducing the number of physical devices required.
c. Cost-effectiveness: The use of common hardware for multiple communication standards and the ability to reconfigure devices through software updates can lead to cost savings in the long run. SDR eliminates the need to replace or upgrade hardware

for each new communication standard, which is particularly beneficial in rapidly evolving technology landscapes.

d. Interoperability: SDR allows for seamless interoperability between different radio systems and protocols. By reconfiguring the software, an SDR device can communicate with various existing communication standards, enabling better coordination and compatibility between different systems.
e. Enhanced performance: SDR enables the implementation of advanced signal processing techniques such as digital filtering, error correction, and adaptive modulation. These capabilities can improve signal quality, increase data rates, and improve overall performance.
f. Rapid prototyping and development: Developing and prototyping new radio systems is faster and more straightforward with SDR. Researchers can quickly modify and test new algorithms and protocols in software without requiring extensive hardware changes.
g. Spectrum efficiency: SDR allows for dynamic spectrum access, where the radio system can intelligently adapt its frequency and bandwidth usage based on the current spectrum availability. This feature enhances spectrum efficiency and makes better use of available resources.
h. Reducing energy consumption: SDR systems can optimize power consumption by dynamically adjusting signal processing and transmission parameters. This adaptability can lead to energy-efficient radio communications.
i. Remote management and monitoring: With SDR, remote management and the monitoring of radio devices have become easier. Software updates, performance monitoring, and troubleshooting can be carried out remotely, reducing the need for physical intervention.

The emulator was implemented using a universal software radio peripheral (USRP) module, USRP 2954, with the specifications introduced in [40]. The emulator employs a 0.01 to 6 GHz channel with a 160 MHz channel bandwidth. The proposed real-time emulator deploys an embedded FPGA kit that executes the signal processing operations, including the convolution process of Equation (22). The detailed description of the proposed emulator was introduced in [38]. A block diagram of the transmit–receive process over the proposed emulator is shown in Figure 2.

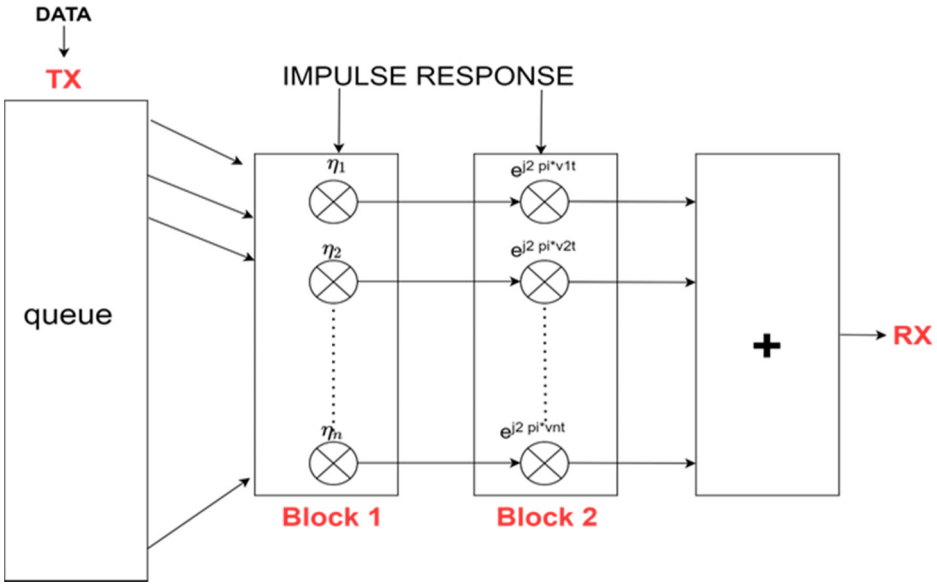


**Figure 2.** Block diagram of the transmit–receive process over the proposed emulator [38].

## 6. Simulation and Results

The proposed flying fog-based IoT model and the developed offloading scheme were evaluated using simulation and emulation processes. In this section, we introduce and discuss the obtained results. We first simulated the system for various scenarios and then emulated some scenarios. The emulator was limited in terms of hardware; thus, not all scenarios could be emulated (e.g., dense deployment scenario).

### *6.1. Simulation Evaluation*

The system was simulated using Matlab for the network with the specifications introduced in Table 1. A network of fifty randomly distributed IoT nodes and a single centralized gateway was simulated. Figure 3 presents the recorded delay versus transmission data rate for MEC and flying fog servers. The delay decreases whenever the transmission data rate increases for all IoT nodes offloading to either a fog or MEC server. However, the delay achieved by flying fog handling is longer than that of the ground MEC unit dedicated to the gateway. This is due to the larger communication distances and limited computational capabilities.

**Table 1.** Simulation parameters.

| Parameter | Value |
|---|---|
| $N_f$ | 5 |
| $S_i$ | ϵ [10–30] MB |
| $N_r$ | 5 |
| $C_i$ | 1900 Cycle/s |
| $T_i^{constraint}$ | 2 ms |
| $R_{IoT,Drone}$, $R_{Drone,\ IoT}$, $R_{Drone\text{-}MEC}$ | ϵ [3–9] Mbps |
| $\omega_{fog}$ | ϵ [0.5–3] GHz |
| $\omega_{MEC}$ | ϵ [12–15] GHz |

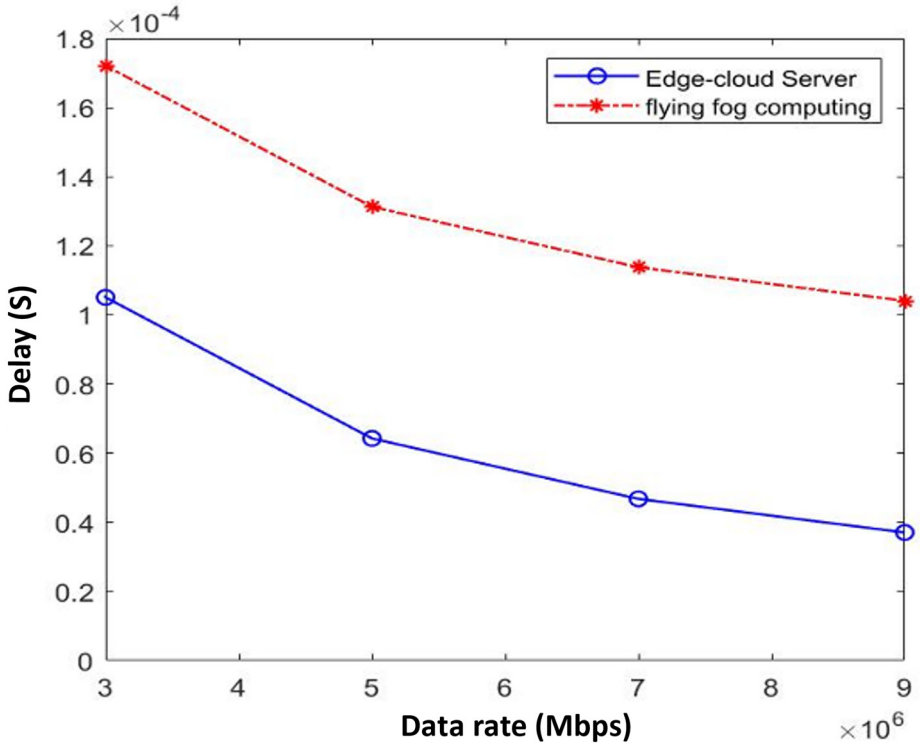


**Figure 3.** Delay vs. data rate.

Moreover, Figure 4 provides the recorded delay versus computational capability in the case of flying fog computing and MEC server computing. With the increased computational capability of the handling server, the total required delay decreased in both cases. This is due to the reduction in processing time. Also, flying fog computing takes slightly longer due to higher communication distances.

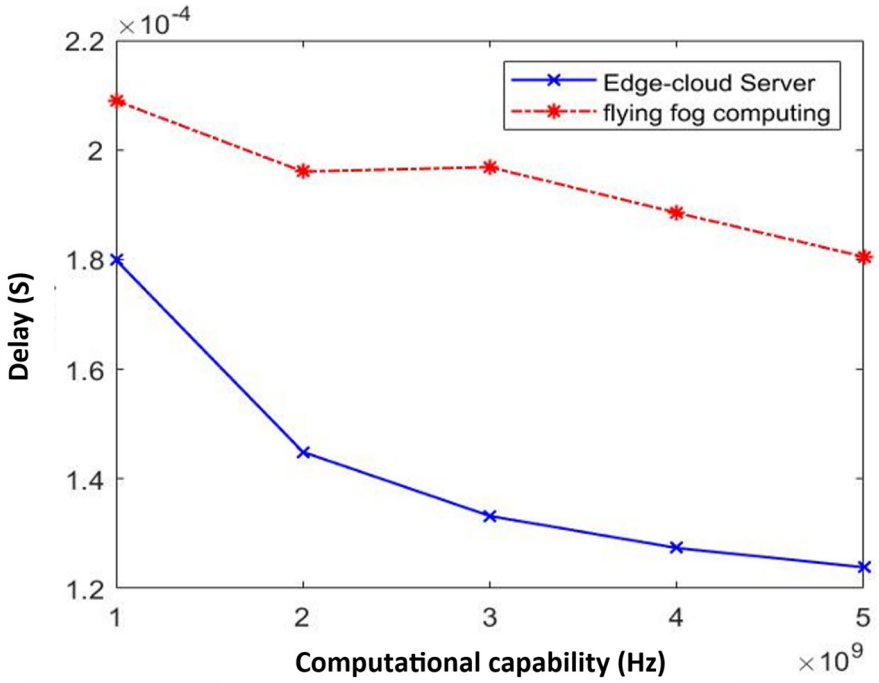


**Figure 4.** Delay vs. computational capability.

Figures 5 and 6 provide the average delay with the change in data size and number of available tasks. The average delay increased linearly with the increase in the data size and number of IoT tasks. The delay, in the case of flying fog computing, is considerably longer than the delay of handling at the MEC server because of the higher computation capacity of the MEC server and the lower communication distances. The longer delays introduced in flying fog computation are reasonable compared to the other benefits that could be enjoyed. Also, this increase does not affect the QoS because the proposed model introduces a time constraint to handle tasks in a way that meets the QoS requirements. After this time, the task is terminated. Accordingly, to reduce the delay of flying fog computation, we need to increase the flying fog computational capacity or the number of fog nodes.

To evaluate the effect of changing the number of deployed flying fog drones on the delay, we simulated a cluster of the IoT using a variable number of flying fog drones. Figure 7 presents the average delay in handling IoT tasks using different numbers of flying fog units. As the number of flying fog drones increases, the delay decreases considerably.

For dense deployment, we simulated the system at different IoT nodes for the same number of flying fog and relay drones. We simulated two systems: the proposed flying fog drone-based IoT and the MEC-based IoT without drone assistance. Figure 8 provides the average latency for both systems. Introducing flying fog drones provides additional computing resources, which reduces the average delay by a considerable value. This is mainly applicable to a larger number of deployed devices. Figure 9 provides the blocking rate of tasks, which is reduced by introducing flying fog.

### *6.2. Experimental Evaluation*

We set up a testbed using our proposed emulator and the structure presented in Figure 10. The testbed deploys three USRPs connected to a workstation. The system uses a

40 MHz channel, a sampling rate of 0.4 GHz, and a 16 QAM modulation scheme. Figure 11 presents the real hardware used for implementation.

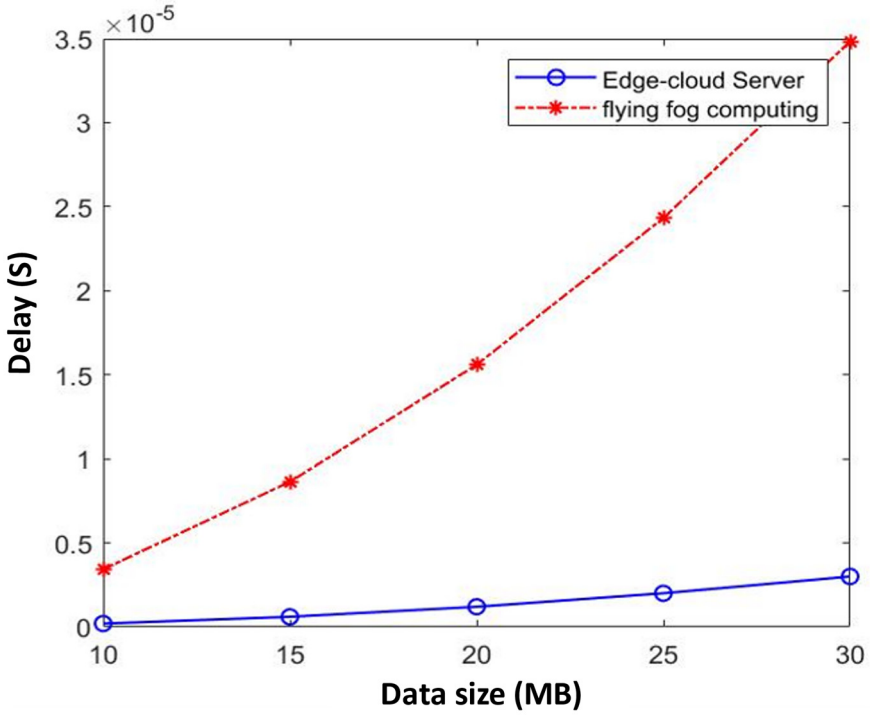


**Figure 5.** Delay vs. data size.

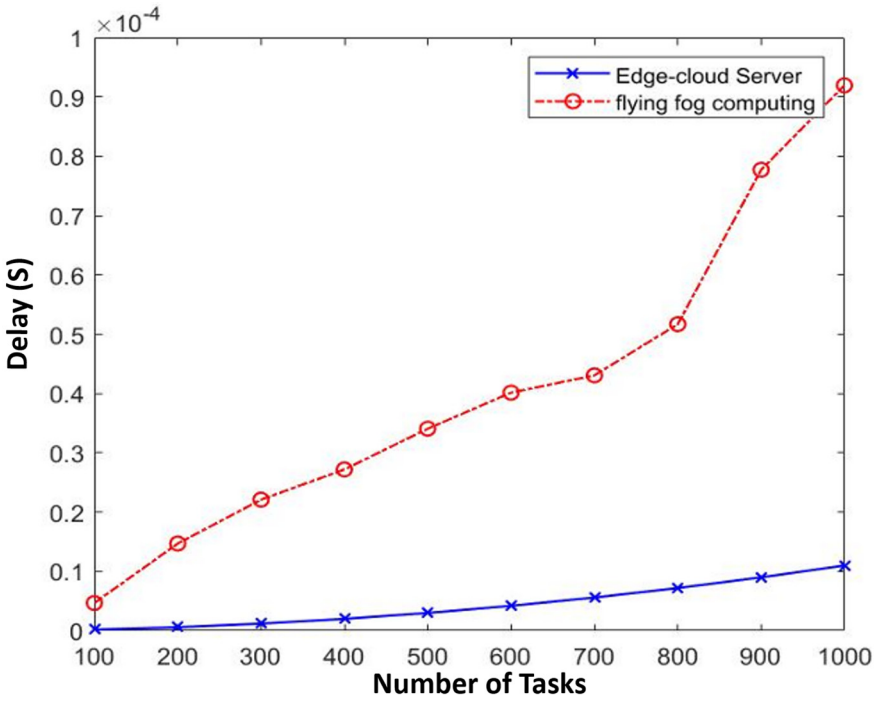


**Figure 6.** Delay vs. number of tasks.

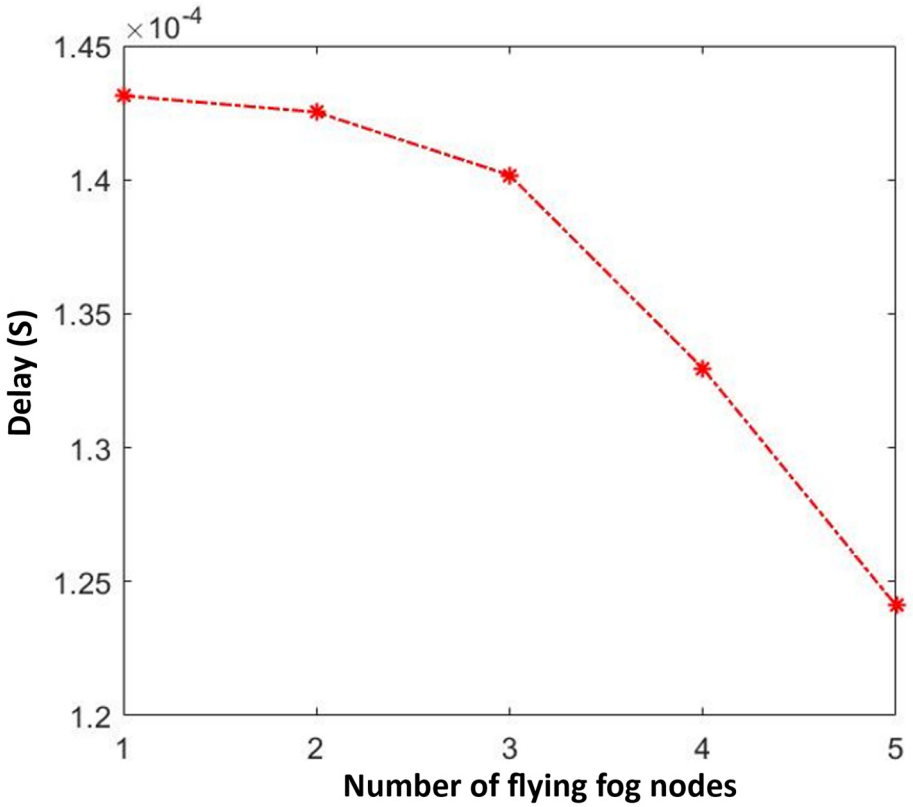


**Figure 7.** Delay vs. the number of flying fog drones.

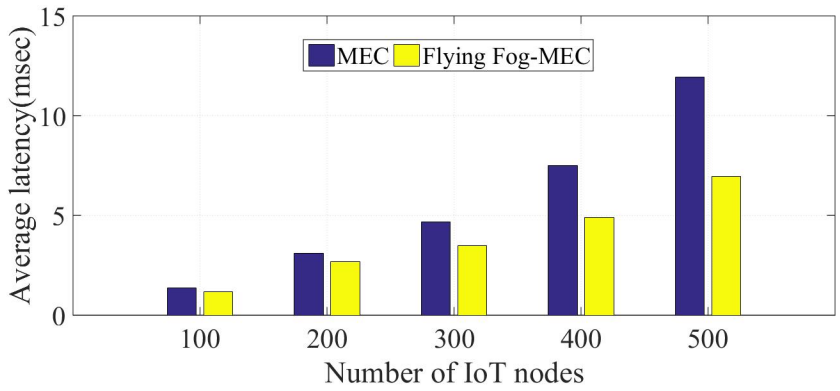


**Figure 8.** Latency vs. number of IoT nodes.

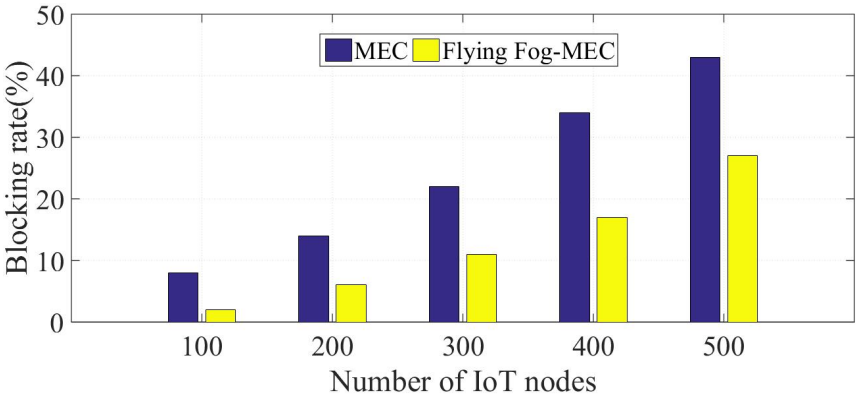


**Figure 9.** Blocking rate vs. number of IoT nodes.

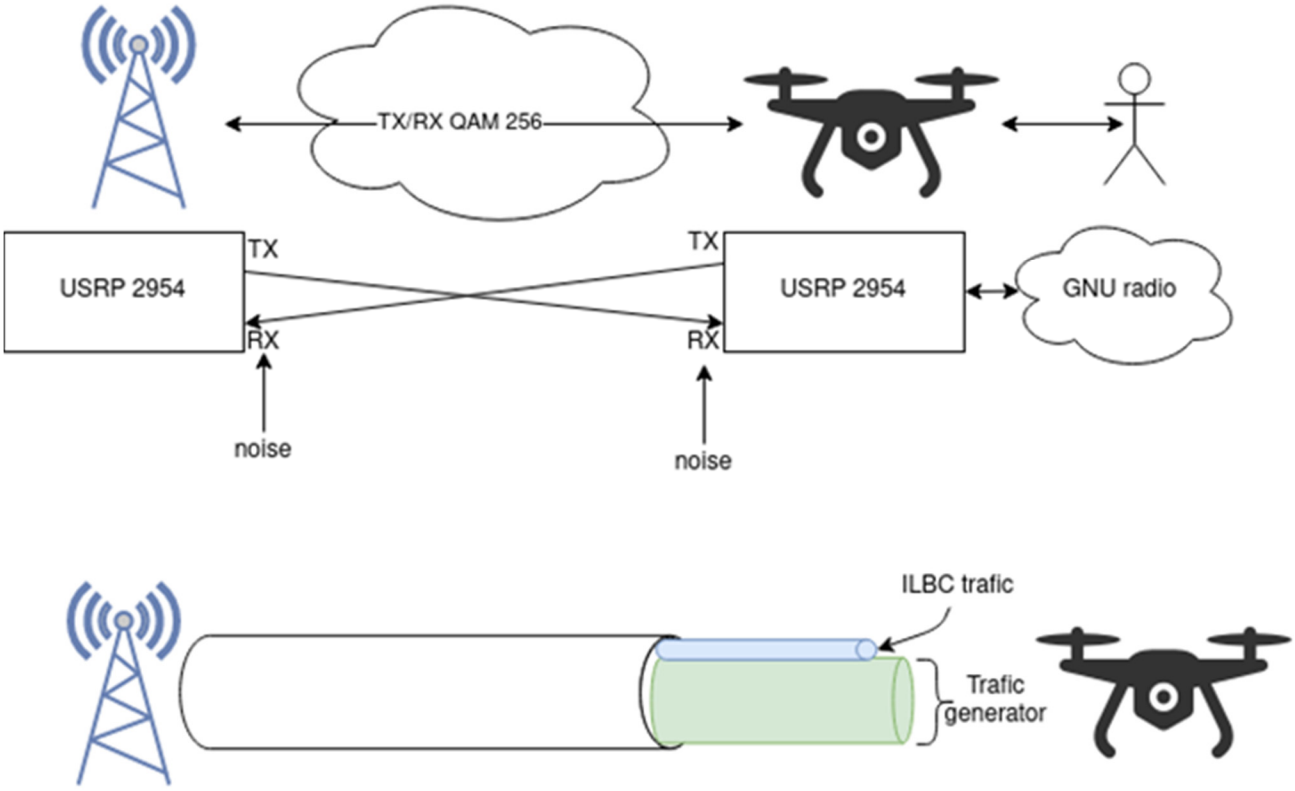


**Figure 10.** Testbed structure.


**Figure 11.** Hardware implementation.

Figures 12 and 13 present the cumulative distribution function (CDF) of average delays for both considered computing paradigms. Figure 12 presents the CDF of delay for the task handling at the flying fog drone; however, Figure 13 presents the results for task execution at MEC. The average delay of task handling at the flying fog exceeds that of the MEC due to the greater capabilities. However, this latency cost achieves higher connectivity and system availability. Figure 14 presents the individual measurements of delays at different sizes of served tasks for both flying fog and MEC. The results are presented at a confidence interval of 0.99.

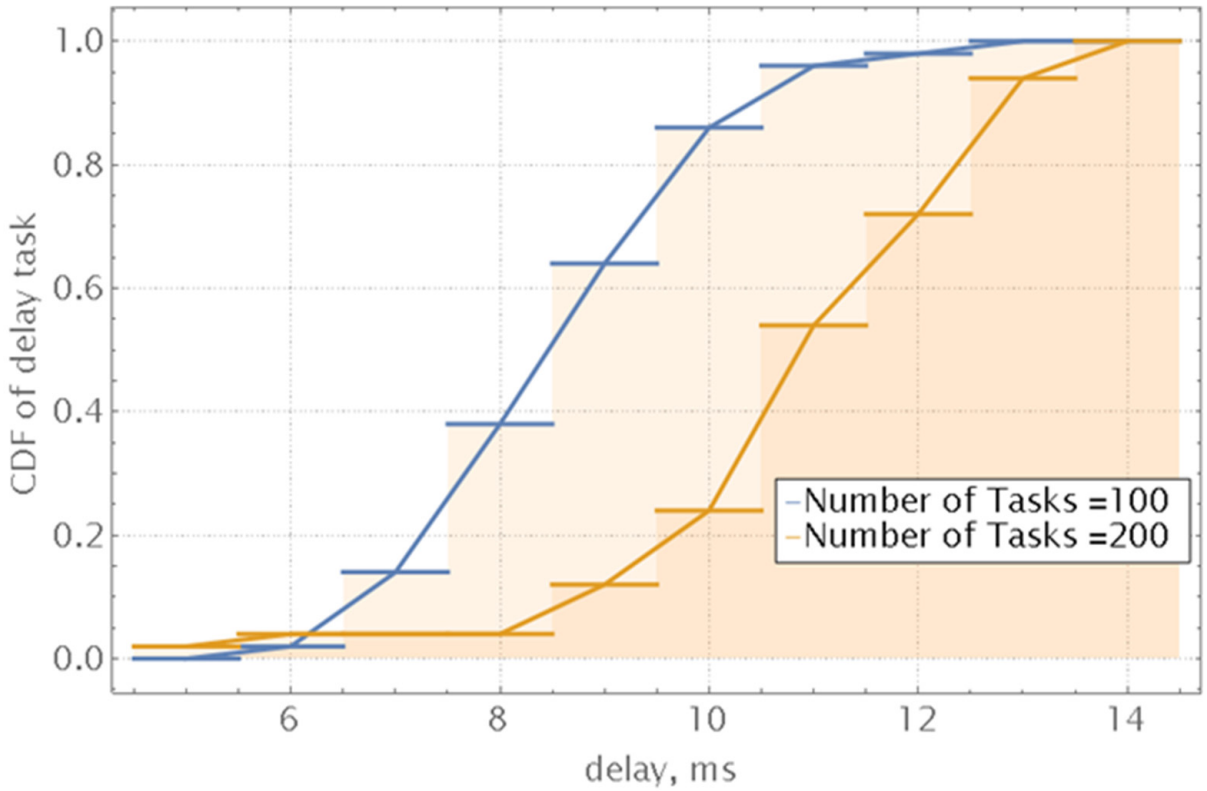


**Figure 12.** CDF of delay for task handling at the flying fog drone.

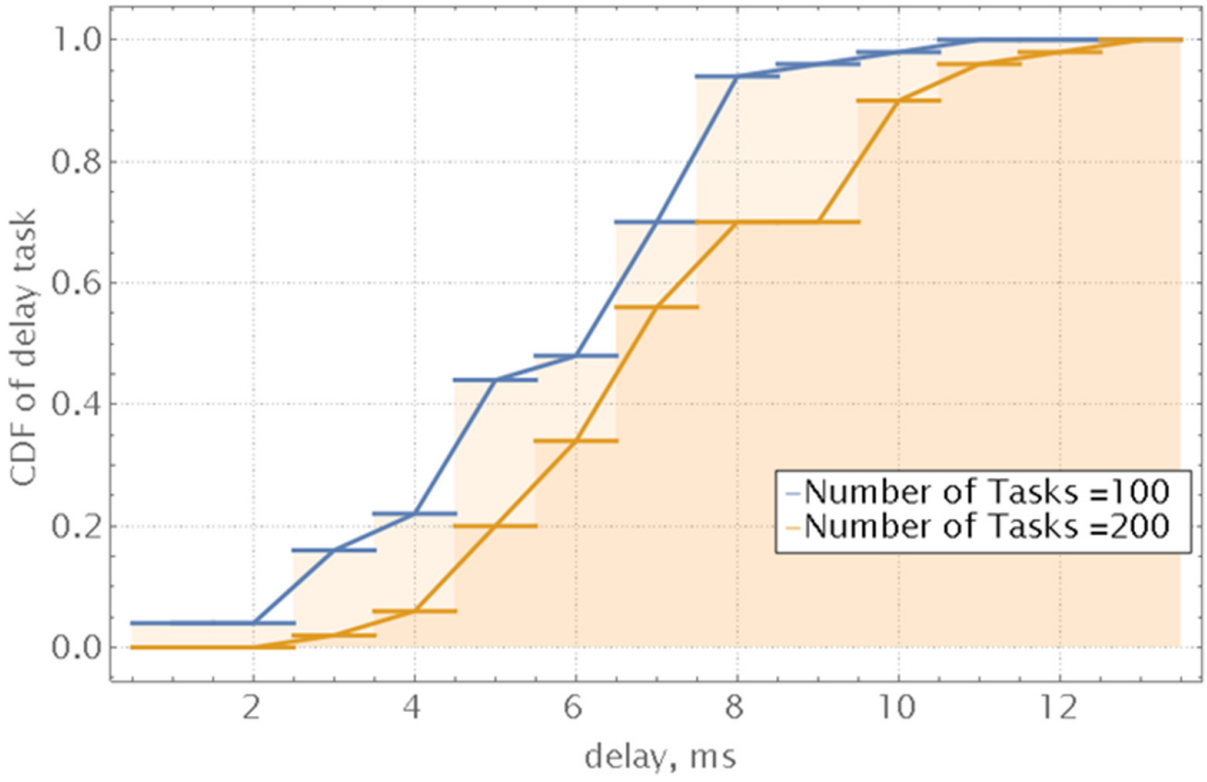


**Figure 13.** CDF of delay for task handling at MEC.

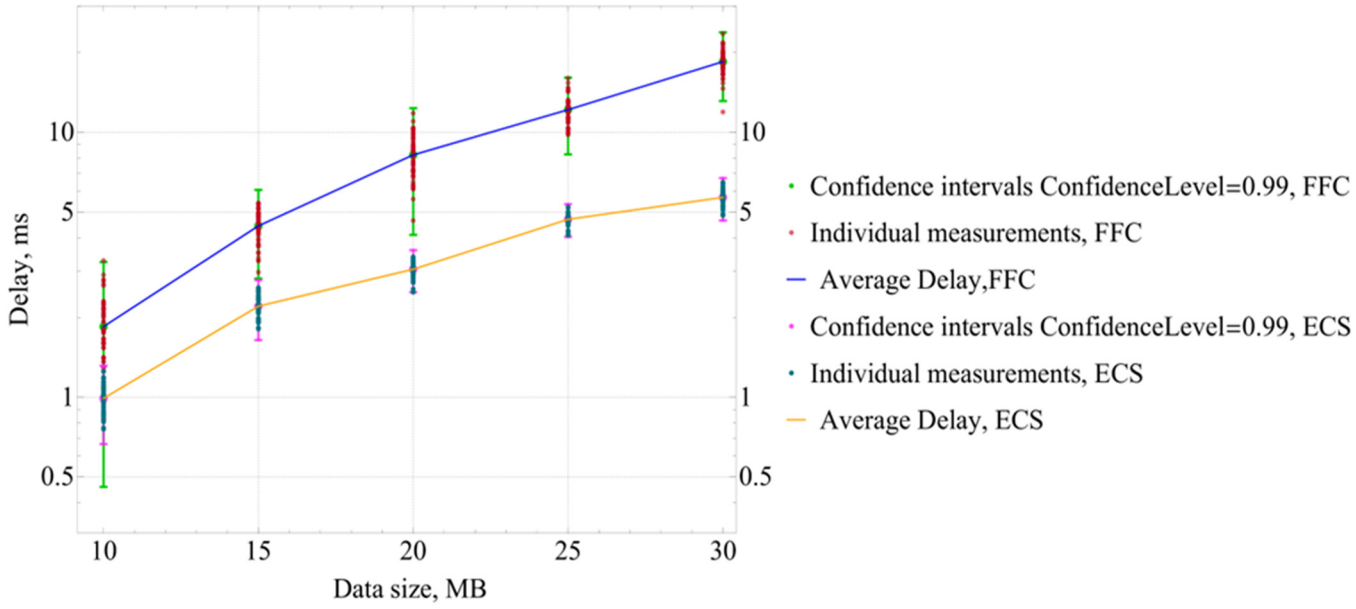


**Figure 14.** Delay vs. data size. Fly fog computing (FFC); edge–cloud server (ECS).

The mobility of the drone affects the data reliability of the system. This is due to the change in channel characteristics (e.g., the Doppler effect). Figure 15 presents the total packet loss at different drone speeds.

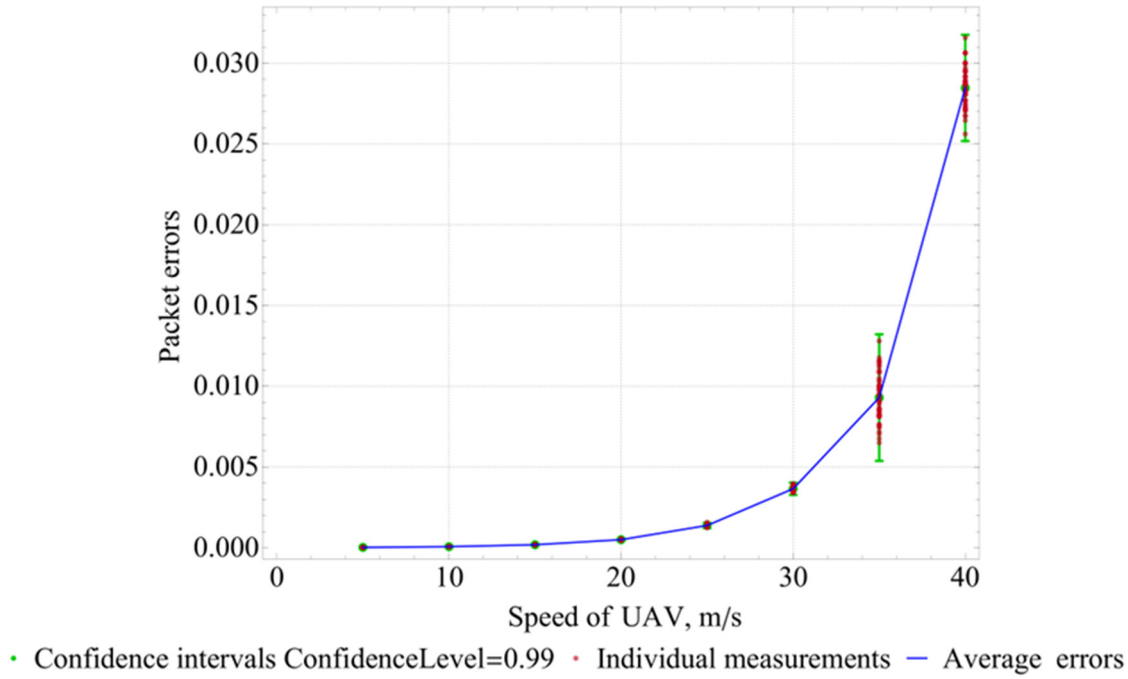


**Figure 15.** Packet loss at different speeds of the drone.

Both sets of results obtained from the simulation and emulation processes support each other. We considered the emulation after simulation processes to ensure the obtained results for near real scenarios. The results for latency and reliability matched in both considered evaluation methodologies. These results validate the developed model in terms of reliability and latency.

## 7. Conclusions

In this work, we addressed the growing demand for low-latency and high-throughput applications in the context of the rapid proliferation of IoT devices. We introduced the concept of flying fog computing, a promising solution that leverages drones equipped with fog nodes to enable data processing and storage closer to the network edge. This demonstrates various benefits over traditional static edge computing paradigms, including coverage improvement, enabling dense deployment and increasing availability and reliability within IoT networks. We developed a novel offloading model using dynamic programming to harness the full potential of flying fog-based IoT networks. The key factors considered in this decision-making process were the computational capacity of the fog nodes, the communication constraints of the IoT devices, and the latency requirements. Extensive simulations and experiments were conducted to evaluate our developed algorithm's performance. The proposed flying fog model outperformed traditional static edge computing. It reduced the average rate of task blocking and the latency required to handle IoT tasks.

**Author Contributions:** Conceptualization, A.A.A., A.M., M.E., and W.M.; methodology, A.A.A., A.K., A.M., and A.A.A.E.-L.; software, A.A.A., A.K., W.M., and A.M.; validation, A.A.A., M.S.A.M., A.K., W.M., and A.A.A.E.-L.; formal analysis, A.A.A., A.M., M.S.A.M., A.K., and W.M.; investigation, A.A.A., A.K., and A.M.; resources, A.A.A.E.-L., A.M., W.M., and A.A.A.; data curation, M.S.A.M., A.M., and A.A.A.E.-L.; writing—original draft preparation, A.A.A., A.K., A.M., and W.M.; writing—review and editing, M.S.A.M., A.A.A., A.M., and A.A.A.E.-L.; visualization, A.A.A., A.M., and A.A.A.E.-L.; supervision, W.M. and A.M.; project administration, A.A.A. and A.A.A.E.-L.; funding acquisition, W.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by The National Key Research and Development Program of China (2021YFB3301000) and The Chongqing Talent Plan Project (cstc2021ycjh-bgzxm0206).

**Data Availability Statement:** The data are contained within the article and/or available from the corresponding author upon reasonable request.

**Acknowledgments:** The authors would like to acknowledge the support of Prince Sultan University, Riyadh, Saudi Arabia, in part to acknowledge the support by the RUDN University Strategic Academic Leadership Program (recipient Abdukodir Khakimov).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Furstenau, L.B.; Rodrigues, Y.P.R.; Sott, M.K.; Leivas, P.; Dohan, M.S.; López-Robles, J.R.; Cobo, M.J.; Bragazzi, N.L.; Choo, K.-K.R. Internet of Things: Conceptual Network Structure, Main Challenges and Future Directions. *Digit. Commun. Netw.* **2023**, *9*, 677–687. [CrossRef]
2. Ateya, A.A.; Algarni, A.D.; Hamdi, M.; Koucheryavy, A.; Soliman, N.F. Enabling Heterogeneous IoT Networks over 5G Networks with Ultra-Dense Deployment—Using MEC/SDN. *Electronics* **2021**, *10*, 910. [CrossRef]
3. Jin, B.; Long, F.; Xia, F.; Chen, S.; Xu, H.; Zhan, W.; Feng, W.; Zhang, R. Advantages of 5G Slicing technology in the internet of things. In Proceedings of the 2023 IEEE International Conference on Integrated Circuits and Communication Systems (ICICACS), Raichur, India, 24–25 February 2023; pp. 1–5.
4. Qadir, Z.; Le, K.N.; Saeed, N.; Munawar, H.S. Towards 6G Internet of Things: Recent Advances, Use Cases, and Open Challenges. *ICT Express* **2023**, *9*, 296–312. [CrossRef]
5. Ateya, A.A.; Muthanna, A.; Koucheryavy, A.; Maleh, Y.; El-Latif, A.A.A. Energy Efficient Offloading Scheme for MEC-Based Augmented Reality System. *Cluster Comput.* **2023**, *26*, 789–806. [CrossRef]
6. Bhatia, S.; Mallikarjuna, B.; Gautam, D.; Gupta, U.; Kumar, S.; Verma, S. The Future IoT: The current generation 5G and next generation 6G and 7G technologies. In Proceedings of the 2023 International Conference on Device Intelligence, Computing and Communication Technologies, (DICCT), Dehradun, India, 17–18 March 2023; pp. 212–217.
7. Jamshed, M.A.; Ali, K.; Abbasi, Q.H.; Imran, M.A.; Ur-Rehman, M. Challenges, Applications, and Future of Wireless Sensors in Internet of Things: A Review. *IEEE Sens. J.* **2022**, *22*, 5482–5494. [CrossRef]
8. You, K.Y. A Summary on 5G and Future 6G Internet of Things. In *Advances in Wireless Technologies and Telecommunication*; IGI Global: Hershey, PA, USA, 2023; pp. 196–243. ISBN 9781799892663.
9. Fortino, G.; Savaglio, C.; Spezzano, G.; Zhou, M. Internet of Things as System of Systems: A Review of Methodologies, Frameworks, Platforms, and Tools. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 223–236. [CrossRef]

10. Sefati, S.S.; Halunga, S. Ultra-reliability and Low-latency Communications on the Internet of Things Based on 5G Network: Literature Review, Classification, and Future Research View. Trans. *Emerg. Telecommun. Technol.* **2023**, *34*, e4770. [CrossRef]
11. Ateya, A.A.; Ali Alhussan, A.; Abdallah, H.A.; Al duailij, M.A.; Khakimov, A.; Muthanna, A. Edge Computing Platform with Efficient Migration Scheme for 5G/6G Networks. *Comput. Syst. Sci. Eng.* **2023**, *45*, 1775–1787. [CrossRef]
12. Hazra, A.; Rana, P.; Adhikari, M.; Amgoth, T. Fog Computing for Next-Generation Internet of Things: Fundamental, State-of-the-Art and Research Challenges. *Comput. Sci. Rev.* **2023**, *48*, 100549. [CrossRef]
13. Apat, H.K.; Nayak, R.; Sahoo, B. A Comprehensive Review on Internet of Things Application Placement in Fog Computing Environment. *Internet Things* **2023**, *23*, 100866. [CrossRef]
14. Hayawi, K.; Anwar, Z.; Malik, A.W.; Trabelsi, Z. Airborne Computing: A Toolkit for UAV-Assisted Federated Computing for Sustainable Smart Cities. *IEEE Internet Things J.* **2023**, 10. [CrossRef]
15. Gupta, A.; Gupta, S.K. A Survey on Green Unmanned Aerial Vehicles-based Fog Computing: Challenges and Future Perspective. Trans. Emerg. *Telecommun. Technol.* **2022**, *33*, e4603. [CrossRef]
16. Devraj; Rao, R.S.; Das, S. Fog computing environment in flying ad-hoc networks. In *Cloud Computing Enabled Big-Data Analytics in Wireless Ad-Hoc Networks*; CRC Press: Boca Raton, FL, USA, 2022; pp. 31–48. ISBN 9781003206453.
17. Fernando, N.; Loke, S.W.; Avazpour, I.; Chen, F.-F.; Abkenar, A.B.; Ibrahim, A. Opportunistic Fog for IoT: Challenges and Opportunities. *IEEE Internet Things J.* **2019**, *6*, 8897–8910. [CrossRef]
18. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the Internet of Things. In Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; ACM: New York, NY, USA, 2012.
19. Verma, U.; Bhardwaj, D. Fog Computing paradigm for Internet of Things: Architectures, issues, challenges, and applications. In *Lecture Notes in Networks and Systems*; Springer Nature Singapore: Singapore, 2022; pp. 1–11. ISBN 9789811910173.
20. Hazra, A.; Adhikari, M.; Amgoth, T.; Srirama, S.N. Fog Computing for Energy-Efficient Data Offloading of IoT Applications in Industrial Sensor Networks. *IEEE Sens. J.* **2022**, *22*, 8663–8671. [CrossRef]
21. Kumari, N.; Yadav, A.; Jana, P.K. Task Offloading in Fog Computing: A Survey of Algorithms and Optimization Techniques. *Comput. Netw.* **2022**, *214*, 109137. [CrossRef]
22. Gasmi, K.; Dilek, S.; Tosun, S.; Ozdemir, S. A Survey on Computation Offloading and Service Placement in Fog Computing-Based IoT. *J. Supercomput.* **2022**, *78*, 1983–2014. [CrossRef]
23. Chen, Y.; Zhao, J.; Hu, J.; Wan, S.; Huang, J. Distributed Task Offloading and Resource Purchasing in NOMA-Enabled Mobile Edge Computing: Hierarchical Game Theoretical Approaches. *ACM Trans. Embed. Comput. Syst.* **2023**, 14. [CrossRef]
24. Wadhwa, H.; Aron, R. Optimized Task Scheduling and Preemption for Distributed Resource Management in Fog-Assisted IoT Environment. *J. Supercomput.* **2023**, *79*, 2212–2250. [CrossRef]
25. Ataie, I.; Taami, T.; Azizi, S.; Mainuddin, M.; Schwartz, D. D2FO: Distributed dynamic offloading mechanism for time-sensitive tasks in Fog-Cloud IoT-based systems. In Proceedings of the 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), Austin, TX, USA, 11–13 November 2022; pp. 360–366.
26. Tran-Dang, H.; Kim, D.-S. Dynamic Collaborative Task Offloading for Delay Minimization in the Heterogeneous Fog Computing Systems. *J. Commun. Netw.* **2023**, *25*, 244–252. [CrossRef]
27. Shahzad, H.; Szymanski, T.H. A dynamic programming offloading algorithm for mobile cloud computing. In Proceedings of the 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Vancouver, BC, Canada, 15–18 May 2016; pp. 1–5.
28. Wang, B.; Lv, B.; Song, Y. A Hybrid Genetic Algorithm with Integer Coding for Task Offloading in Edge-Cloud Cooperative Computing. *IAENG Int. J. Comput. Sci.* **2022**, *49*, 503–510.
29. Reddy, P.B.; Sudhakar, C. An Osmotic Approach-Based Dynamic Deadline-Aware Task Offloading in Edge–Fog–Cloud Computing Environment. *J. Supercomput.* **2023**, 1–23. [CrossRef]
30. Lakhan, A.; Mohammed, M.A.; Obaid, O.I.; Chakraborty, C.; Abdulkareem, K.H.; Kadry, S. Efficient Deep-Reinforcement Learning Aware Resource Allocation in SDN-Enabled Fog Paradigm. *Autom. Softw. Eng.* **2022**, *29*, 20. [CrossRef]
31. Thanedar, M.A.; Panda, S.K. A Dynamic Resource Management Algorithm for Maximizing Service Capability in Fog-Empowered Vehicular Ad-Hoc Networks. *Peer Peer Netw. Appl.* **2023**, *16*, 932–946. [CrossRef]
32. Hosseini, E.; Nickray, M.; Ghanbari, S. Optimized Task Scheduling for Cost-Latency Trade-off in Mobile Fog Computing Using Fuzzy Analytical Hierarchy Process. *Comput. Netw.* **2022**, *206*, 108752. [CrossRef]
33. Yao, J.; Ansari, N. Online Task Allocation and Flying Control in Fog-Aided Internet of Drones. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5562–5569. [CrossRef]
34. Serdaroglu, K.C.; Baydere, Ş.; Saovapakhiran, B.; Charnsripinyo, C. Location aware fog computing based air quality monitoring system. In Proceedings of the 2023 International Conference on Smart Applications, Communications and Networking (SmartNets), Istanbul, Turkiye, 25–27 July 2023.
35. Khuwaja, A.A.; Chen, Y.; Andreou, A.; Mavromoustakis, C.X.; Batalla, J.M.; Markakis, E.K.; Mastorakis, G.; Mumtaz, S. UAV Trajectory Optimisation in Smart Cities using Modified A* Algorithm Combined with Haversine and Vincenty Formulas. *IEEE Trans. Veh. Technol.* **2023**, *72*, 9757–9769.
36. Zhao, N.; Alouini, M.-S.; Dobbins, P. A Survey of Channel Modeling for UAV Communications. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2804–2821. [CrossRef]

37. Park, S.; Kim, H.T.; Lee, S.; Joo, H.; Kim, H. Survey on Anti-Drone Systems: Components, Designs, and Challenges. *IEEE Access* **2021**, *9*, 42635–42659. [CrossRef]
38. Khakimov, A.; Mokrov, E.; Poluektov, D.; Samouylov, K.; Koucheryavy, A. Evaluating the Quality of Experience Performance Metric for UAV-Based Networks. *Sensors* **2021**, *21*, 5689. [CrossRef]
39. Kafetzis, D.; Vassilaras, S.; Vardoulias, G.; Koutsopoulos, I. Software-Defined Networking Meets Software-Defined Radio in Mobile Ad Hoc Networks: State of the Art and Future Directions. *IEEE Access* **2022**, *10*, 9989–10014. [CrossRef]
40. USRP-2954 Specifications. Available online: https://www.ni.com/docs/en-US/bundle/usrp-2954-specs/page/specs.html (accessed on 25 July 2023).