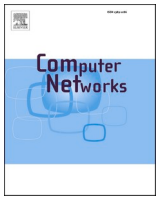
Computer Networks 263 (2025) 111233

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



An energy-efficient decentralized federated learning framework for mobile-IoT networks

Nastooh Taheri Javan [a,*], Elahe Zakizadeh Gharyeali [b], Seyedakbar Mostafavi [b,*]

[a] Computer Engineering Department, Imam Khomeini International University, Qazvin, Iran
[b] Department of Computer Engineering, Yazd University, Yazd, Iran

ARTICLE INFO

Keywords:
Clustering
Decentralized federated learning
Energy consumption
Mobile IoT
Sleep/wake-up scheduling

ABSTRACT

The Internet of Things (IoT) comprises a vast number of interconnected devices that generate and share enormous amounts of data. Traditional machine learning approaches, which rely on the exchange of raw data, are impractical for real-world applications with extremely high data volumes due to challenges such as energy constraints and node mobility. To mitigate these overheads in IoT, Federated Learning (FL) can be employed, decentralizing the learning process to various devices without the need for centralized data collection or sharing. In this paper, we propose a new energy-efficient decentralized federated learning framework aimed at reducing energy consumption in mobile IoT. This framework utilizes a Master/Slave clustering method and a dynamic sleep/wake-up strategy, ensuring that the Base Station (BS) does not interfere with the aggregation of learning models and only supervises the clustering process. To rigorously evaluate the results of the proposed approach, we initially present a Linear Programming (LP) mathematical model designed to optimize energy consumption costs. Simulation results demonstrate that the proposed scheme improves energy consumption by up to 52 % compared to the star scheme and 41 % compared to the hierarchical method. Additionally, the proposed approach achieves a high accuracy performance of approximately 98 %, significantly surpassing standard schemes. These quantitative results highlight the effectiveness of our approach in optimizing energy use and enhancing model performance in mobile IoT environments.

1. Introduction

The Internet of Things (IoT) is rapidly gaining traction, with forecasts indicating that approximately 125 billion smart devices will be interconnected by 2030 [1]. The pervasive integration of IoT across various applications has become so significant that meeting sustainable development goals, particularly in reducing energy consumption, is unattainable without incorporating IoT. As a result, vast amounts of data are generated and exchanged within these networks by devices like robots, sensors, smart devices, and mobile phones. This substantial volume of data must be analyzed and processed to serve the intended application [2].

In many applications, raw information gathered from the network is analyzed using intelligent, machine learning-based approaches to derive final results and data. Traditionally, data is sent from personal devices or organizational data centers to a server for processing, where the final analysis is conducted centrally [3]. This method incurs several costs, including increased risk of data leakage during transmission, creating network traffic, and significantly increasing the server's computational load [4].

To address these concerns, new alternative approaches are being developed in the IoT environment, aiming to perform as much of the learning process locally on the devices as possible. This minimizes the need to transfer data to a central server. One such innovative machine learning approach is federated learning [5].

In federated learning, the majority of the learning process is distributed and performed on the end devices, eliminating the need to exchange and share datasets. Instead, only "learning model updates" need to be exchanged [6]. In this approach, the learning model computations are conducted locally on the end devices, and the learning parameters are then sent to the server for aggregation. While federated learning initially seems like an excellent approach, its implementation in practical environments is not straightforward and comes with challenges [7]. These challenges include the need for computational power at the end stations, energy constraints at these nodes, the requirement for sufficient data samples to achieve the desired accuracy, and the

* Corresponding authors.
E-mail addresses: nastooh@eng.ikiu.ac.ir (N. Taheri Javan), a.mostafavi@yazd.ac.ir (S. Mostafavi).

https://doi.org/10.1016/j.comnet.2025.111233
Received 29 September 2024; Received in revised form 26 January 2025; Accepted 16 March 2025
Available online 17 March 2025

necessity to exchange learning parameters between devices and the server during each learning round [8].

In general, based on the strategies for information exchange between devices and the server, federated learning approaches are divided into two main types: Centralized Federated Learning (CFL) and Decentralized Federated Learning (DFL) [9]. In the centralized federated learning approach, the central server plays a relatively more active role in the learning process. Devices train their models with local data and then send their local models to the server. The server aggregates these models, updates the global learning model, and sends it back to the devices. The main challenge with this approach is that, in practice, nodes must continuously communicate with the server to exchange learning parameters, significantly increasing energy consumption in these networks. Additionally, this approach still relies on a central server with high reliability and processing power [10].

In contrast, to overcome the limitations of centralized federated learning methods, decentralized approaches have been introduced to eliminate dependency on a central server. These approaches use device-to-device (D2D) communications, exchanging learning parameters with neighboring devices without the need for a base station (BS), enabling the learning process to be distributed and completed within the network. In this approach, each node executes the learning process using local data and then aggregates its learning model by receiving updates from its neighbors [11].

The use of decentralized federated learning in mobile IoT also presents challenges, and given the nature of devices in these networks, reducing energy consumption remains the primary challenge in this approach [12]. IoT devices are generally mobile and use wireless communications, have limited energy, and thus are often reluctant to participate in the learning process. Additionally, factors such as the increasing number of devices participating in the learning process, unstable and weak bandwidth in wireless environments, node mobility, and the low speed of model parameter exchange disrupt the learning process in these methods, making it harder to achieve a final and universal learning model in decentralized federated learning methods.

To overcome the issues of decentralized federated learning in IoT networks, several solutions have been proposed. One approach focuses on reducing the communication overhead related to learning model parameters, where each device calculates its local updates and sends them to a central server based on conventional topologies [13]. Some approaches concentrate on improving D2D communications, enabling devices to exchange data directly without the support of a central coordinator [14]. Other approaches focus on compression schemes to reduce the volume of exchanged data [15]. Conversely, some methods aim to reduce the communication rounds needed for parameter exchange [16]. Additionally, several approaches focus on utilizing clustering ideas during the learning process [17].

In this paper, a new hybrid approach for utilizing decentralized federated learning in IoT-based networks is presented. In the proposed approach, the base station does not get involved in the aggregation of federated learning model parameters. Instead, the base station is solely responsible for the clustering process and the selection of cluster heads. The aggregation of learning model parameters is performed by edge devices in a decentralized manner, with the goal of minimizing the involvement of the base station and central server in the learning process. For the clustering process, the proposed approach employs a hybrid method based on the geographical location and availability time of the devices, unlike previous approaches. This means that devices located geographically close to each other and available within a certain timeframe are grouped into the same cluster.

The proposed approach uses an unscheduled scheme; thus, the tasks of the devices within the cluster are not predetermined. This approach offers devices substantial flexibility and adaptability, allowing them to respond dynamically to various conditions such as device availability, energy levels, and network connectivity. As a result, it enables the seamless integration of a broader array of devices into a federated learning system. This integration provides multiple benefits in terms of energy consumption and accuracy in learning models, which will be discussed in Section 5. Initially, the clustering process and selection of appropriate cluster heads for each cluster are performed. Subsequently, the clusters are divided into two main types: *master* and *slave*, based on the number of devices participating in the learning process within each cluster. The master cluster, having a higher number of participating devices and therefore collecting more data, provides better model parameters and becomes the core of the network. Slave clusters, using their local data and the parameters received from the master cluster, perform local learning and send their results back to the master cluster, enhancing and optimizing the final model.

To further reduce energy consumption, a sleep/wake-up mechanism is introduced for the clusters. In this mechanism, clusters with insufficient devices for effective collaborative learning can enter a sleep mode. During this time, they receive appropriate parameters from a master cluster, which has suitable conditions for learning. Additionally, the parameters generated by the devices involved in the learning process are transmitted to the cluster head using the local FedProx algorithm [18, 19].

An important point here is that in the initial phase of learning, processing the data in slave-clusters is inefficient due to the low number of participating devices in those clusters. Therefore, devices in slave clusters enter a sleep mode until new parameters are received from the master cluster. This mechanism also applies to devices in the master cluster, which remain in sleep mode until a new model needs to be trained. Asynchronous communications are used for intra-cluster device communication; in this mechanism, if a device moves out of its cluster during the learning process, other devices continue the learning process independently without waiting, reducing the wandering effect [20].

In this research, a mathematical model based on Linear Programming (LP) is used to formulate and evaluate the mentioned mechanisms, and mathematical and conceptual modeling is performed to describe all the variables in the problem-solving process. To precisely evaluate the proposed framework, its performance is compared with both the optimized mathematical model and existing approaches in Section 4.

The key contributions of this research can be summarized as follows:

- Proposing a decentralized federated learning framework for mobile IoT based on asynchronous D2D communications.
- Introducing a dynamic clustering method with the assistance of the base station, suitable for implementing decentralized federated learning in mobile IoT.
- Developing a sleep/wake-up mechanism to reduce energy consumption in clusters formed for implementing decentralized federated learning in mobile IoT.
- Modeling the proposed framework using linear programming to optimize energy consumption in mobile IoT.
- Evaluating the performance of the proposed framework in comparison with similar approaches and the optimized state based on a mathematical model.

The structure of this paper is organized as follows: The second chapter reviews related research. The third chapter outlines the system model, while the fourth chapter discusses the proposed framework in detail. In the fifth chapter, numerical results are presented. Finally, the sixth chapter offers a summary and conclusions of the study.

2. Related work

In this chapter, we will first review the concept of federated learning. Subsequently, recent studies utilizing centralized federated learning in the Internet of Things will be examined, followed by an exploration of recent research on decentralized federated learning in IoT. Finally, to provide a comprehensive view of the research path in this field and to compare the achievements of previous works with the proposed

approach, a general comparison of the existing approaches has been presented.

2.1. Introduction to federated learning

Federated learning, also known as collaborative learning, is a distributed machine learning approach where model parameters are shared only between network devices and a central server [21]. In this paradigm, the process unfolds in three stages: each device initially receives an initial model from the server, trains it with its private data, and sends the updated model parameters back to the central server. Once these parameters are aggregated, a global model is created by the server. This iterative process continues until the global model achieves the required accuracy for the subsequent training rounds across all nodes. This approach allows computations to be performed closer to the data, addressing critical issues such as privacy preservation and data locality efficiently [22].

Federated learning finds diverse applications such as autonomous vehicles [23], traffic prediction [24], telecommunications [25], IoT [26] and Industrial IoT (IIoT) [27], AI-driven healthcare [28] and security issues [29]. Its implementation in IoT offers advantages such as preserving privacy, reducing data exchange latency, improving scalability, enhancing learning models, and reducing energy consumption [30].

Today, federated learning in IoT is broadly categorized into two types: centralized federated learning with a central server and decentralized federated learning without a server [31]. Centralized federated learning involves a central server and a set of devices that communicate concurrently with the server to aggregate and update learning parameters. In contrast, decentralized federated learning typically operates without a central server to mitigate existing bottlenecks. In this approach, devices connect peer-to-peer (P2P) and receive the aggregated model through neighboring devices. Thus, each node performs local learning based on its data and aggregates its model based on updates received from its peers. The key advantage of this approach lies in enhancing scalability for applications where access to a central server is either unavailable or impractical.

2.2. Centralized federated learning in IoT

Several studies explore advanced techniques and algorithms to optimize centralized federated learning across diverse IoT and mobile networks. In [32], a two-step communication protocol and dynamic resource allocation strategy enhance centralized federated learning by maximizing bandwidth allocation among clients, selecting participants, and appointing leaders to improve global model accuracy. This approach outperforms FedAvg in terms of communication cost and model accuracy.

In [33], Li et al. introduce a data compression scheme, FT-LSGD-DB, tailored for edge wireless devices, which integrates local stochastic gradient algorithms and gradient reduction strategies. This scheme significantly reduces communication rounds and sizes, achieving substantial energy savings. Additionally, hierarchical federated learning is explored in [34], demonstrating EARA's effectiveness in resource allocation for heterogeneous IoT systems, improving classification accuracy by accommodating non-iid. data distributions.

Luo et al. [35] propose a time-sharing scheduling scheme for federated learning in edge mobile networks, optimizing client participation and local iterations to reduce training time and energy consumption. Furthermore, MUCSC, introduced in [36], utilizes model compression techniques like B-MUCSC to enhance communication efficiency by grouping minor updates within super-clusters, thereby reducing overall communication traffic.

In [37], a dual-level incentive mechanism mitigates node failures through evolutionary game theory, allowing clusters to reward user participation and compete for model services. Yang et al. [38] address energy and computational resource efficiency in wireless federated learning with algorithms optimizing time, bandwidth, power, and frequency allocation.

In [39], a weighted Proximal learning-based algorithm minimizes energy consumption and completion time for federated learning in wireless IoT, demonstrating superior efficiency compared to traditional methods.

The studies by Wu et al. [40,41], Gong et al. [42,43], and Yu et al. [44] introduce various approaches such as semi-asynchronous federated averaging, Semi-SynFed for Internet of Vehicles, clustered federated learning, Hierarchical Federated Edge Learning (HFEL), and ELASTIC algorithm for wireless IoT, each focusing on specific challenges and optimizations in federated learning environments.

2.3. Decentralized federated learning in IoT

Recently, extensive research has been conducted on the application of decentralized federated learning in the Internet of Things, and we will review some of these studies below. In [45], a Distributed Federated Learning (DBFL) approach is introduced, aiming to achieve scalable and energy-efficient exchange of trained models. This approach focuses on the heterogeneity of data and their feature spaces, employing autoencoders for model aggregation. Reported results demonstrate that this approach outperforms centralized federated learning approaches in terms of accuracy and energy efficiency.

In [46], a new Federated Learning approach called TT-HF is proposed as a semi-decentralized federated learning method. In this approach, Device-to-Device (D2D) communications are utilized for federated learning between end devices and a server. During each global aggregation round, devices send their local model parameters via D2D communications within local clusters. Experiments show that TT-HF offers improvements in model accuracy and network energy consumption under various statistical heterogeneity scenarios.

In [47], a FL-EOCD framework is proposed for decentralized federated learning, aiming to reduce energy consumption using D2D communications and overlapping clustering. In this method, a cluster is defined as a coverage area for an end device, and devices overlapping in cluster areas are termed Bridge Devices (BDs). Clusters are interconnected via BDs either in a star or hierarchical topology, allowing for decentralized deployment of cluster-specific models without the need for a central server. Results indicate that the proposed approach enhances energy consumption and learning time compared to similar hierarchical and star-based approaches.

In [48], focused and distributed learning approaches for Internet of Drones (IoD) have been proposed using graph theory computations. The proposed federated learning approach for IoD utilizes a decentralized distribution of local parameters among drones in network overlap regions to enable aggregation of a global model. Results show that the decentralized approach provides comparable performance in terms of privacy preservation and energy consumption compared to centralized approaches among drones.

In [49], leveraging edge-to-cloud distributed training using Federated Edge Learning (DFEL), an intelligent model training process is distributed across nodes from edge devices to cloud servers. This method considers a multi-layered heterogeneous device framework with added local network topology structures, synchronized via D2D communications. This approach is expected to be beneficial for latency-sensitive applications such as intelligent factory automation, transitioning from star topologies to distributed topologies to significantly reduce network resource costs. Horizontal/vertical communication optimization, resource allocation, clustering issues, and network dynamics are not adequately addressed in this scheme.

In [50], a clustered federated learning approach is presented for vehicular networks. This approach utilizes Vehicle-to-Vehicle (V2V) communication to overcome federated learning communication bottlenecks. In each round, a subset of vehicles is selected to act as a cluster head, and other vehicles are matched with them. Non-iid is employed to

converge learning models, creating new models for non-participating individuals and new vehicles. Finally, a greedy algorithm for selecting and allocating cluster head resources and a two-part matching algorithm with maximum weight for cluster formation are proposed.

In [51], a hybrid approach combining Decentralized Federated Energy Learning (DFEL) and Multi-Principal Single-Agent (MPOA) contract-based methods is proposed for Electric Vehicle (EV) networks to increase profit for Charging Stations (CSs). The proposed approach allows CSs to locally train their energy transactions for accurate demand prediction while ensuring data privacy preservation. On the other hand, using MPOA, maximizing the profitability of CS is formulated as a non-collaborative energy contract problem, allowing each desirable CS to increase its appeal under common constraints of the Smart Grid Provider (SGP). The results show that the proposed method increases energy demand prediction accuracy and reduces communication overhead.

2.4. Comparison of existing approaches

In this section, to clarify the position of the proposed approach in the field of research, we've made an effort to provide a straightforward analysis of previous research that relates to the proposed solution in this paper. To achieve this and offer a more comprehensive view, earlier centralized and decentralized approaches are briefly summarized in Table 1. Through this comparison, the contributions and innovations of the proposed approach are highlighted, particularly in combining the sleep/wake mechanism with the federated learning process, which helps improve energy efficiency in a meaningful way. Moreover, the table provides a good overview of recent research trends in using federated learning for IoT and makes the connection between the proposed approach and earlier studies more clear. The last row of this table compares the proposed approach of this paper based on the same comparative framework.

3. System model

In this chapter, the system model and assumptions considered for implementing the proposed framework are delineated, including network model, communication model, mobility model, and learning model. The system model outline is presented in Fig. 1.

3.1. Network model

In this proposed framework, the edge network consists of IoT devices along with a base station. End devices include smart sensors, smartphones, etc., within the IoT environment, providing raw data for machine learning algorithms. These devices are typically mobile and have specific constraints such as processing power, memory, and battery life. On the other hand, the base station plays a crucial role in clustering and assigning primary and secondary roles based on the number of participating devices in a cluster. Selecting an appropriate device as a cluster head is based on factors like device energy and mobility.

Table 1
Summary of key contributions and frameworks for improving FL in IoT and mobile networks.

Approache	Refs.	Key Solutions Compression	Clustering	D2D	S/W	Mobility	Evaluation
Centralized	[32]	×	✓	✓	×	✓	*Addressing scalability issues in federated learning by enhancing model accuracy and communication efficiency.*
	[33]	×	✓	×	×	×	*Focusing on energy efficiency in edge computing environments, ensuring relevance for real-world applications with resource constraints.*
	[34]	×	×	×	×	✓	*Emphasizing user assignment in heterogeneous settings to broaden the applicability of federated learning across various scenarios.*
	[35]	×	✓	×	×	×	*Balancing client participation and resource management to improve efficiency in mobile federated learning systems.*
	[36]	✓	✓	×	×	×	*Minimizing bandwidth usage through compression strategies, addressing communication bottlenecks in federated learning.*
	[37]	×	✓	×	×	×	*Enhancing user engagement and participation through an incentive mechanism, while exploring practical implementation challenges in diverse environments.*
	[38]	×	×	×	×	✓	*Reducing energy consumption to enable more sustainable federated learning systems.*
	[39]	✓	×	×	×	✓	*Addressing real-world constraints with a pragmatic approach, contributing to environments with limited resources.*
	[40]	×	×	×	×	✓	*Advancing client-server dynamics management through the semi-asynchronous federated averaging (SAFA) protocol.*
	[41]	×	×	×	×	✓	*Improving efficiency in IoV systems, showcasing the importance of adaptive federated learning protocols in transport applications.*
	[42]	×	✓	×	×	✓	*Solving data heterogeneity challenges with AdaCFL to enhance model performance in recommender systems.*
	[43]	×	✓	×	×	×	*Addressing communication challenges in edge computing with the HFEL framework to improve federated learning efficiency.*
	[44]	×	×	×	×	✓	*Optimizing resource management and energy efficiency in IoT with the ELASTIC algorithm while maintaining model performance.*
Decentralized	[45]	×	✓	×	×	×	*Tackling data heterogeneity while enhancing accuracy and energy efficiency with DBFL, suitable for real-world applications.*
	[46]	×	×	✓	×	×	*Improving accuracy and energy utilization by incorporating D2D communications in federated learning systems.*
	[47]	×	✓	✓	×	✓	*Enabling decentralized model distribution through cluster connections via bridge devices, reducing energy consumption and learning time.*
	[48]	×	✓	✓	×	✓	*Offering a balanced solution by combining centralized and decentralized methods for IoD applications.*
	[49]	×	✓	✓	×	×	*Coordinating D2D communications in heterogeneous devices with a multi-layer hybrid learning structure for latency-sensitive applications.*
	[50]	×	✓	×	×	✓	*Enhancing communication efficiency by selecting cluster heads based on update similarity, accelerating learning in Non-IID settings.*
	[51]	×	✓	×	×	✓	*Maximizing utility for charging stations while ensuring data privacy through local training, improving energy demand forecasting and reducing communication overhead.*
	Our paper	×	✓	✓	✓	✓	***Utilizing a dynamic sleep/wake mechanism and asynchronous communications to address energy and mobility challenges in federated learning.***

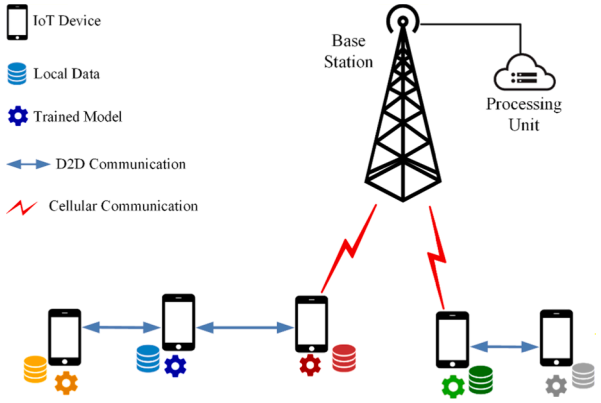


Fig. 1. An overview of the proposed system model.

The base station and IoT devices are essential components in this system, where devices form a set $D = \{D_1, D_2, ..., D_N\}$ within the coverage area of the base station denoted by B. $D_B \subseteq D$ represents devices communicating with BS. Each device d_i in D has a geographical location defined by coordinates (x_i, y_i), indicating its position in the environment.

The base station leverages clustering informed by device statuses, combining geographic location approaches and device availability. Clusters are formed with devices that are simultaneously available and physically close to each other, facilitating possible D2D links. Similar to the approach in research [47], it is assumed that each device cannot transmit and receive data simultaneously, making the D2D channel half-duplex. RRBs are employed as communication protocol models for parameter transmission, minimizing bandwidth interference and link failures caused by mobility.

3.2. Communication model

In this study, both Device-to-Device (D2D) and cellular communications are considered. D2D communication is utilized when devices are within each other's coverage range, improving energy efficiency compared to traditional infrastructure-based communications. Conversely, if devices are outside each other's coverage, cellular communications are used, providing connectivity beyond D2D coverage limits. Finally, parameters are relayed to respective cluster heads through neighboring devices (relays), enabling devices outside the cluster coverage to transmit their parameters for final aggregation.

3.3. Mobility model

In this framework, we use a simple mobility model to make it easier to understand while still capturing the essential dynamics relevant to our study. The mobility model used in this system assumes that the movement of each device is determined solely based on its geographical position, without considering the device's speed [52]. These assumptions allow us to focus on the key aspects of federated learning optimization without introducing unnecessary complexity.

At each time step, their positions are updated according to their current coordinates and a simplified mobility model. The new position of device d_i at time t is represented as $(x'i.\ y'i)$, calculated based on (1) and (2):

$$x'i = x_i + \Delta t \tag{1}$$

$$y'i = y_i + \Delta t \tag{2}$$

where Δt represents the time interval between updates. The mathematical description of the mobility model is also defined according to (3):

$$(x_i.y_i) \rightarrow (x'i.y'i) = (x_i + \Delta t.y_i + \Delta t) \tag{3}$$

where (x_i, y_i) represents the initial position of the device, and (x'_i, y'_i) denotes its updated position.

3.4. Learning model

In the proposed framework, collected data are not sent to a central location; instead, final devices, aided by federated learning principles, train their models based on local data. This approach employs a federated learning algorithm called FedProx, particularly advantageous for heterogeneous data environments compared to other algorithms like FedAvg, especially when dealing with non-IID data distributions [19].

The FedProx algorithm is an improved version of the FedAvg algorithm, specifically designed to handle and address non-IID data and heterogeneous models, especially in IoT environments [53]. This algorithm introduces a "proximal term," which adds a regularization term proportional to the distance between the updated local model and the global model, to keep devices closer to the global model. This approach prevents devices with highly heterogeneous data from diverging too far from the global model and ensures better convergence [54].

The regularization parameter λ controls the permissible deviation between local and global models; a higher λ results in more restricted updates, leading to greater homogeneity, while a lower λ allows for more heterogeneity. Generally, λ should be carefully adjusted to balance between fitting local data and maintaining proximity to the global model [55]. The value of λ will be determined and discussed in the performance evaluation section.

4. Proposed framework

In this section, the details of the proposed framework are outlined, including the clustering mechanism, the sleep/wake-up mechanism, and the federated learning mechanism. Subsequently, the proposed framework is formulated. Table 2 summarizes the main notations used in this part.

4.1. Clustering mechanism

The clustering operations in the proposed framework are centered around the base station. In this approach, the base station first identifies the set of devices within a specific time slot t_{sb} and a geographical radius r_b; this resulting set is represented as follows:

Table 2
Summary of symbols.

Notation	Description
M_d	Mobility of a device, indicating how much it has moved
$M_{threshold}$	A threshold value to assess low mobility
M	Devices in the master cluster
H_M	Cluster head of the master cluster
S_i	Devices in the slave cluster i
H_{S_i}	Cluster head of the slave cluster i
X_M	Data set in the master cluster
X_{S_i}	Data set in the slave cluster i
S_j	Device state in sleep/wake mode
M_g	Global parameter/model
$f\ (w)$	Initial model based on the master cluster data
$g\ (w)$	Learning process with the device dataset and creating a local parameter
$h\ (w)$	Aggregation of local parameters by the master cluster head
$\Delta\ (Mg)$	Difference between new and previous model/parameters
$threshold$	Control of changes in parameters
M'_g	Previous model/parameter
$f(x;\theta)$	Global model across the dataset
θ_i	Current local parameter for device
λ	Regularization parameter

$$C\ (ts_b.\ r_b) = \{d_i | ts_b\ \in\ A_i\ \&\&\ distance\ (d_i.\ (x_b.y_b)) < r_b\} \quad (4)$$

where (x_b, y_b) are the coordinates of the base station. In this approach, each device d_i has a set of available time slots denoted as A_i, which is represented as $A_i = \{ts_1, ts_2, \ldots\}$, where ts_k is a specific time slot available to d_i. Based on this, the base station defines a cluster for each time slot ts_k, referred to as C_k, which includes the devices for which that time slot is available, defined as follows:

$$C_k\ (ts_k)\ = \{d_i \mid ts_k \in A_i\} \quad (5)$$

After this, based on the defined cluster, a location-based cluster is established. This cluster includes all devices located within a specified geographic radius R, defined as follows:

$$C_l = \{d_i | distance\ (d_i.d_j) <\ R.\ \forall\ d_i.d_j\ \in\ C_l\} \quad (6)$$

A location-based cluster C_l is defined, encompassing devices within a specified geographic radius R, each device has a limited coverage area, representing its service area as a circle with radius R. Subsequently, by intersecting C_a and C_l as $C = C_a\ (ts_k) \cap C_l$, a cluster C is formed, comprising devices that are both temporally and geographically proximate. Ultimately, all devices are divided into k clusters using this method: $C = \{C_1, C2, \ldots, C_k\}$.

In the proposed approach, the clustering details are handled using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [56]. This method can identify points that are close to each other by considering the updated positions of devices due to mobility [57]. To achieve this, the density of data points within a specified radius is denoted by e, and the minimum number of points required to form a cluster is denoted by m. Specifically, e is a distance parameter that sets the maximum permissible distance between two data points for them to be considered neighbors, indicating how close devices must be to each other to be grouped in the same cluster. Additionally, m determines the minimum number of devices that must exist within the e-neighborhood for a point to be considered a core point.

DBSCAN is chosen here over other clustering methods, such as k-means, because of its robustness to noise and its ability to identify clusters without prior knowledge of their number. This flexibility is critical in environments where device distribution and density can vary significantly. In addition, DBSCAN is particularly effective for mobile devices because it can adapt to real-time changes in location and density, making it an ideal choice for implementing our proposed clustering approach in the federated learning framework [56,57].

4.1.1. Determining master and slave clusters

After completing the first stage of the clustering process and identifying the set of clusters as $C = \{C_1, C_2, \ldots, C_K\}$, the cluster with the most members is designated as the master cluster. Formally, this is expressed as:

$$Master = C_x where\ C_{x\_devicesize}\ = \max\left(C_{i\_devicesize}\right) for\ all\ C_i\ in\ C \quad (7)$$

For subsequent periods, if the cluster still has the most devices, it will continue to be selected as the Master due to the higher participation of devices. If the cluster loses its devices due to mobility or weak connections (based on a threshold T, which is defined as a percentage of the master cluster size D_m and given by $T = \gamma \times D_m$, where $0 < \gamma < 1$), then dynamically, the cluster with the highest score will be selected as the new Master.

In the meantime, the remaining clusters are considered as slave clusters, defined as $slaves(t) = C - \{master(t)\}$, until a slave cluster grows and becomes the master cluster, and the former master cluster takes on the role of a slave. Overall, the cluster with the most devices is always selected as the master cluster. It is worth noting that during the learning process, the devices within a cluster are allowed to communicate with each other to collect local parameters for updating the global model.

4.1.2. Selecting master and slave cluster heads

After performing the clustering operation and determining the primary and Slave clusters, the next step is to designate the cluster head for each cluster. To select an appropriate cluster head for the master cluster, the proposed approach first considers the remaining energy levels of the devices, followed by their mobility.

In the first step, the device with the highest remaining energy among all devices is selected as the cluster head. This ensures that the chosen device has sufficient energy to function effectively as the cluster head. Once a device is selected as the cluster head, its mobility is assessed based on the parameter M_d. If $M_d < M_{threshold}$, it indicates that the device has low mobility and can continue to serve as the cluster head.

For selecting the slave cluster heads, the connection to the master cluster head is of significant importance. A device is chosen as the slave cluster head based on its ability to act as a bridge to the master cluster. Specifically, the distance between each device in the slave cluster and the master cluster head is calculated, and the device with the minimum distance is selected as the bridge device and the slave cluster head. By incorporating periodic checks of the cluster heads, it can be ensured that slave cluster heads are updated in case of device mobility or unstable connections. This approach guarantees that the cluster heads operate effectively and maintain a robust network topology.

4.2. Sleep/wake-up mechanism

The sleep/wake-up mechanism operates as follows:

1. Initially, all devices in the slave clusters are in sleep mode, while all devices in the master cluster are awake. In other words, $S_j = 0$ for all $j \in S_i$ and $S_j = 1$ for all $j \in M$.
2. The master cluster head creates an initial model by collecting and aggregating parameters from the devices within the cluster according to (8):

$$M_g = f\ (X_M[H_M]) \quad (8)$$

3. The initial model is sent to all the slave cluster heads within its overlap region according to (9):

$$H_{S_i} \cap H_M \neq \varnothing \quad (9)$$

H_{S_i} receives M_g from H_M. This condition in the sleep/wake mechanism is implemented considering that the slave cluster heads overlap with the master cluster head.

4. The slave cluster heads wake up their devices and send the initial model to their member devices. Specifically, H_{S_i} sends M_g to S_i, and S_j is set to 1 for all $j \in S_i$.
5. Each device in the slave clusters performs the learning process with its dataset, creating a local parameter as described in (10):

$$S_i\ = g\ (X_{S_i}) \quad (10)$$

6. Each slave cluster head sends its local parameter to the master cluster head, meaning H_M receives S_i from H_{S_i}.
7. The master cluster head aggregates the parameters to create a new global model (for each slave cluster) as described by (11), denoted by M_g.

$$M_g = h\ (S_i) \quad (11)$$

8. According to (12), if the new global model significantly differs from the previous model, the master cluster head sends the new model to all slave cluster heads, and the process repeats from step 4:

$$\Delta(Mg) > \ threshold.\ H_{S_i} \cap H_M \neq \varnothing \tag{12}$$

where H_{S_i} receives M_g through H_M.

9. Conversely, if the new global model does not show significant differences, the master cluster head sends a sleep signal to all devices in the system, as described in (13):

$$\Delta(Mg) \leq threshold. \forall j \in M => S_j = 0 \tag{13}$$

In steps 8 and 9, the threshold Δ is a parameter that determines when the global model parameters have changed sufficiently to wake up the devices in the system. It is defined as a function of the difference between the current global model and the previous global model, as described in (14):

$$\Delta = \alpha \times ||Mg \ - M'g|| \tag{14}$$

In this framework, the threshold Δ is a parameter that determines when the global model parameters have changed sufficiently to wake up the devices in the system. It is defined as a function of the difference between the current global model (M_g) and the previous global model (M'_g), with $||x||$ representing the L2 norm, indicating the magnitude of change between the two models. The parameter α controls the sensitivity of the threshold and can be set to a high value (e.g., 0.1) to ensure rapid convergence, considering the energy constraints of the devices. This approach measures convergence based on the difference between the current and previous models. If the difference is below the threshold, devices can enter a sleep state since there is no significant new data to learn from. Conversely, if the difference exceeds the threshold, devices should remain awake to learn from the new data. This strategy balances energy efficiency and learning performance.

10. If a device moves out of its current cluster and joins another cluster due to mobility, the mechanism must adapt accordingly. This means that once it is assigned to a new slave cluster or master cluster, it sends the parameters to the adjacent device in the master cluster (M) or slave cluster (S_i). Then, the process resumes from step 4, updating the parameters using the new device's data, as shown in (15).

$$M_g = h\ (S_i|M.\ X_{new}) \tag{15}$$

where X_{new} represents the dataset of the new device. The process repeats from step 3, ensuring that the new device can participate in the learning process.

11. If a device exits the system due to depleted energy or unstable connectivity, its tasks are redistributed among the remaining devices in the corresponding cluster. When the device becomes available again, it receives the current parameters from a neighboring device. Consequently, depending on whether it is a member of a primary or slave cluster, it will repeat steps 5 or 7 respectively.

4.3. Federated learning mechanism

In the proposed framework, each device D at the edge of the network has a local dataset ($\{x_i.\ y_i\}$), where x_i is the input sample i and y_i is the output sample i. It is assumed that X represents the data distribution across a set of devices $D = \{D_1, D_2, \ldots, D_k\}$. Each device D_i maintains a subset X_i of the data, as described by (16).

$$X \ = \cup_i X_i \tag{16}$$

4.3.1. Learning method in the master cluster

The objective of this task is to train a global model $f\ (x;\ \theta)$ across the entire dataset X, where θ represents the model parameters. Here, the process is carried out in a decentralized manner. Devices in the master cluster $M = \{M_1, M_2, \ldots, M_k\}$ train their local parameters $f_i\ (x;\ \theta_i)$ using their local data X_i. Through communication with each other, they update their local parameters and coordinate the training of the global model. Each device sends its parameters to its neighbor, or if it is close to the cluster head, it sends them directly to the cluster head for aggregation. The aggregated model is then sent to the slave cluster heads and finally to the member devices of the master cluster. The steps for this process are as follows [58]:

- $f_i\ (x;\ \theta_i)$ is considered the local model parameter of device M_i at time t.
- In each iteration t, the devices in the master cluster update their local parameters according to (17):

$$\theta_{i.\ t+1} \ = \ argmin_\theta \left[\frac{\lambda}{2} \ \times \ ||\theta \ - \ \theta_{i.\ t}||^2 + \left(\frac{1}{n}\right) \times \sum_{j=1}^{n} L_M \left(f_j\ (x;\ \theta).\ X_j\right) \right] \tag{17}$$

where λ is the regularization parameter, θ_i is the current local parameter for a device D_j trained on the corresponding local dataset X_j. θ represents the global model parameter, which embodies the collective knowledge of all devices. n is the number of devices, and L_M is the local loss function for device D_j, measured with $f_j\ (x;\ \theta)$ and local data X_j. To obtain the global loss function (a sum of local loss functions from slave clusters and the master cluster) on the device's dataset, common examples like linear regression are used, as described in (18) [59]:

$$L_G = \left(\frac{1}{N}\right) \times \Sigma \left[\left(\frac{1}{n_i}\right) \times \Sigma\left(y_j - - \hat{y}_j\right)^2 \right] + \left(\frac{1}{M}\right) \times \Sigma \left[\left(\frac{1}{m_k}\right) \times \Sigma(y_k - - \hat{y}_k)^2 \right] \tag{18}$$

and y_j is the actual target value, $\hat{y}_j$ is the predicted value, n_i is the total number of training samples, and N is the number of slave clusters. Similarly, in the second term, M represents the total number of devices in the master cluster, m_k is the total number of training samples, y_k is the actual target value, and $\hat{y}_k$ is the predicted value. The overall accuracy calculation based on the loss function is given by (19):

$$\in_{G(t)} = \ 1 \ - \ (L_G + \ L_M) \tag{19}$$

- After updating the parameters, the devices in this cluster sequentially send and receive the updated parameters. Device M_i sends its parameter to the neighboring device M_j and device M_i receives it to aggregate their local parameters.
- Finally, the local parameters of the devices are aggregated at the cluster head using (20):

$$f\ (x;\ \theta) = \ \left(\frac{1}{|M|}\right) \times \ \sum_{\{i=1\}}^{k} f_i(x;\ \theta_i.\ t) \tag{20}$$

- In this equation, $|M|$ represents the number of devices in the master cluster. The aggregated parameters are sent to the slave cluster heads and their member devices for further training.

4.3.2. Learning method in the slave cluster

The initial global model $f\ (x;\ \theta)$ is sent by the master cluster head to

the slave cluster heads, following these steps:

- Let $S = \{S_1, S_2, \ldots, S_k\}$ be a set of slave clusters. Each slave cluster S_j has a local dataset X_j and a local model $f_j \quad (x; \theta_j)$ at time t. The local loss function is given by:

$$L_j = \left(\frac{1}{n}\right) \times \Sigma\left(y_j - - \hat{y}_j\right)^2 \tag{21}$$

Additionally, (22) computes the local accuracy [60]:

$$\in_{l(t)} = 1 - L_j \tag{22}$$

- In each iteration t, the devices in each slave cluster S_j update their local parameters using (23):

$$\theta_{\{j.\ i.\ t+1\}} = argmin_\theta \left[\frac{\lambda}{2} \times \left|\left|\theta - \theta_{\{j.\ i.\ t\}}\right|\right|^2 + \left(\frac{1}{N}\right) \times \sum_{k=1}^{N} L_k\left(f\left(x;\ \theta\right).\ X\left\{j.\ k\right\}\right) \right] \tag{23}$$

where λ is the regularization parameter, N is the number of devices, L_k is the local loss function for device D_k, and $X_{\{j.k\}}$ is the local data set for device D_k in the slave cluster.

- Each device D_k in slave cluster S_j sends its local model $\theta_{j.k.t+1}$ to a neighboring device if it is nearby. The slave cluster head aggregates the parameters from the devices using (24).

$$\theta_{\{j.\ t+1\}} = \left(\frac{1}{N}\right) \times \sum_{\{i=1\}}^{N} \theta_{\{j.\ i.\ t+1\}} \tag{24}$$

in which $\theta_{j.t+1}$ represents the aggregated model parameters for slave cluster S_j at time $t+1$. $f_j\ (x;\ \theta_{j.t+1})$ is sent to the master cluster head for aggregation with parameters from other slave clusters and the previous global model [58]. The master cluster head aggregates the parameters from the slave clusters according to (25).

$$f'(x;\ \theta') = \left(\frac{1}{(m+1)}\right) \times \left[f\ (x;\ \theta) + \sum_{j=1}^{m} f_{j(x;\theta_j.t+1)}\right] \tag{25}$$

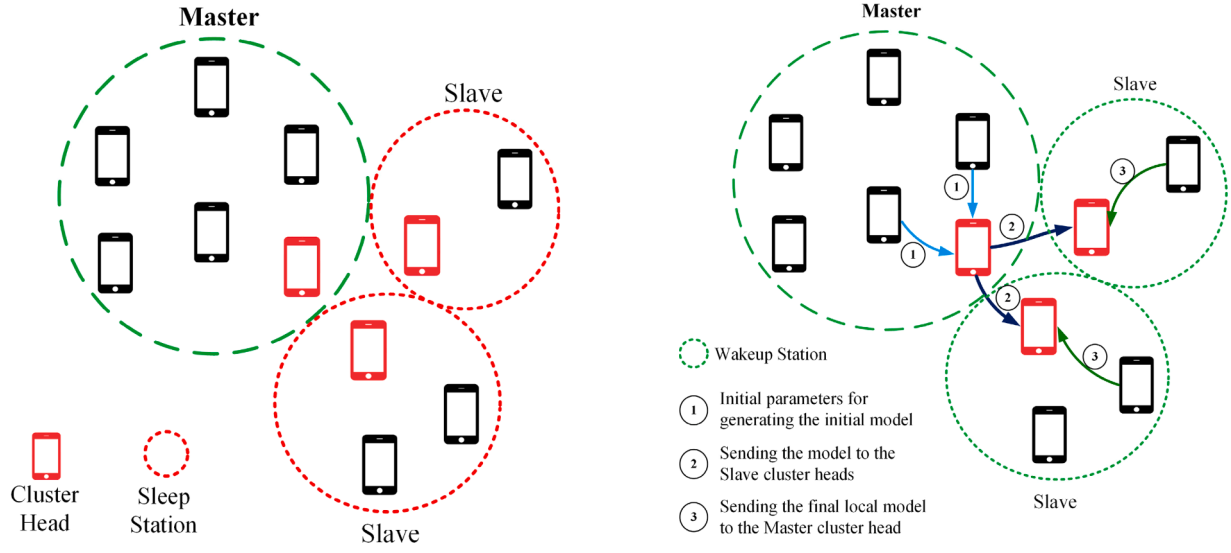
m is the number of slave clusters, and θ' represents the aggregated parameters, which are returned to the slave clusters for further training in the next iteration and distributed among the devices. Similarly, in the master cluster, this process continues until the convergence criteria outlined in Section 5 and the desired accuracy are met. Generally, in this approach, the stages of updating local parameters, transmission, reception, and aggregation are repeated so that over time, the local parameters converge to a common global model.

Eqs. (17) and (23) also represent an optimization problem, where the goal is to minimize the regularization term and the average loss from the local datasets of devices in the clusters. Fig. 2 presents a schematic view of the proposed federated learning approach to provide a better understanding of the clusters, the base station, and the key features of decentralized learning. In this figure, part (a) illustrates the initial clustering and the sleep/wake mechanism in the proposed system, while part (b) depicts the proposed federated training method for global iteration.

Fig. 3 provides an overall view of the steps in the proposed learning process from start to finish, while Fig. 4 summarizes this approach in more detail using pseudocode.

4.4. Formulation

In this section, the proposed framework will be examined from the perspective of energy consumption in computations (energy expended by the device during parameter aggregation) and communications (parameter exchange with neighboring devices or cluster heads). The total energy consumption and execution time will be formulated. It is noteworthy that in the proposed framework, the issue of improving energy consumption in the use of federated learning in the Internet of Things is considered. This includes clustering, selecting appropriate cluster heads, and managing computational and communication energy consumption, practically implemented using concepts like primary/Slave clusters and the sleep/wake mechanism.



a) Initial clustering and sleep/wake mechanism in the proposed system **b) Proposed FL training method for One global iteration**

Fig. 2. Schematic view of the proposed approach.

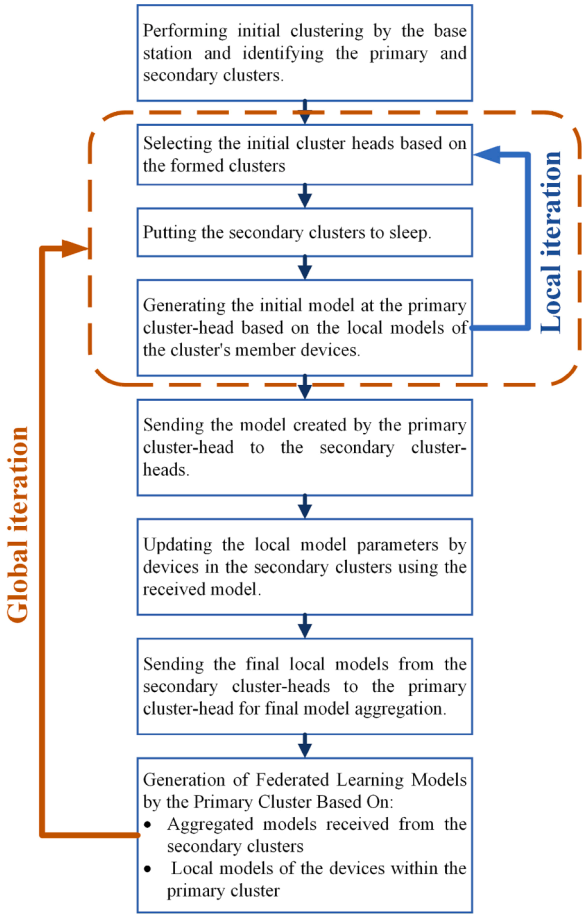


Fig. 3. Proposed FL training method.

4.4.1. Computational and communication energy consumption

In the proposed framework, each device belongs to a cluster (primary or secondary). The Slave clusters are responsible for collecting local parameters from their devices and sending them to the master cluster for aggregation and updating the global model. The following equations present the computational and communication energy consumption for each component in the proposed scheme, along with the total energy consumption.

4.4.1.1. Active devices in the master cluster energy consumption. Computational energy consumption: The active devices in this cluster perform computations on the collected data according to [61], which is calculated using (26):

$$E_{comp_i} = P_{comp_i} \times A_i \tag{26}$$

where P_{comp_i} is the computational power of device i and A_i is its active time.

Communication energy consumption: Devices that communicate with neighboring devices within the same cluster. Based on [47,61], it is calculated using (27):

$$E_{comm_{i,\ j}} = \left(P_{tx_{ij}} \times D_{ij} \times T_{ij}\right) + \left(P_{rx_{ij}} \times D_{ij} \times T_{ij}\right) \tag{27}$$

where P_{tx} (transmission power) and P_{rx} (reception power) represent the parameters between two active devices i and j, and D_{ij} is the distance, and T_{ij} is the transmission time between them. Any device that communicates with the master cluster head uses (28) to determine its energy consumption:

$$E_{comm_{i,\ head}} = \left(P_{tx_{i_{head}}} \times D_{i_{head}} \times T_{i_{head}}\right) + \left(P_{rx_{i_{head}}} \times D_{i_{head}} \times T_{i_{head}}\right) \tag{28}$$

This equation also calculates the transmission and reception power between device i and the master cluster head. Additionally, $D_{i_{head}}$ head represents the distance and $T_{i_{head}}$ head the transmission time between them.

4.4.1.2. Master cluster head energy consumption. Computational Energy Consumption: The master cluster head performs computations on the received parameters. Its energy consumption is determined by (29):

$$E_{comp_{head}} = P_{comp_{head}} \times A_{head} \tag{29}$$

where $P_{comp_{head}}$ is the power required for computations and A_{head} is the active time.

Communication Energy Consumption: The cluster head communicates with its member devices to send updated parameters. The energy consumption is determined by (30):

$$E_{comm_{head_i}} = \left(P_{tx_{head_i}} \times D_{head_i} \times T_{tx_{head_i}}\right) + \left(P_{rx_{head_i}} \times D_{head_i} \times T_{rx_{head_i}}\right) \tag{30}$$

This equation is similar to (27); however, the process occurs between the cluster head and device i. Subsequently, the energy consumption calculation for communication between the cluster head and the slave cluster heads is provided by (31):

$$\mathrm{E_{comm_{head_{slave}}}} = \left(\mathrm{P_{tx_{head_{slave}}}} \times \mathrm{D_{head_{slave}}} \times \mathrm{T_{head_{slave}}}\right) + \left(\mathrm{P_{rx_{head_{slave}}}} \times \mathrm{D_{head_{slave}}} \times \mathrm{T_{head_{slave}}}\right) \tag{31}$$

where $P_{rx_{head_{slave}}}$ and $P_{tx_{head_{slave}}}$ represent the power required for parameter exchange between cluster heads, and $D_{head_{slave}}$ and $T_{head_{slave}}$ denote the distance and time between them, respectively.

4.4.1.3. Slave cluster heads energy consumption. Computational Energy Consumption: Each slave cluster head performs computations on the received parameters and its own local parameters. The energy consumption is calculated using (32):

$$E_{comp_{slave_{j_i}}} = P_{comp_{slave_{j_i}}} \times A_{slave_{j_i}} \tag{32}$$

where $P_{comp_{slave_{j_i}}}$ is the power of device i and $A_{slave_{j_i}}$ is the active time of device i in cluster j.

Communication Energy Consumption: Each device sends its local parameters to the cluster head, and its energy consumption is calculated using (33):

$$E_{comm_{slave_{j_i}}} = \left(P_{tx_{slave_{j_i}}} \times D_{slave_{j_i}} \times T_{slave_{j_i}}\right) + \left(P_{rx_{slave_{j_i}}} \times D_{slave_{j_i}} \times T_{slave_{j_i}}\right) \tag{33}$$

where $P_{tx_{slave_{j_i}}}$ is the transmission power and $P_{rx_{slave_{j_i}}}$ is the reception power. The distance and time between them are represented by $D_{slave_{j_i}}$ and $T_{slave_{j_i}}$ respectively. To calculate the energy consumption for sending parameters from device i to its neighboring device, (34) is used:

$$E_{comm_{i_j}} = \left(P_{tx_{i_j}} \times D_{i_j} \times T_{i_j}\right) + \left(P_{rx_{i_j}} \times D_{i_j} \times T_{d2d_{i_j}}\right) \tag{34}$$

where $P_{tx_{i_j}}$ and $P_{rx_{i_j}}$ represent the transmission and reception power, respectively, for the communication, while D_{i_j} denotes the distance and T_{i_j} the time between the devices. Equation (35) shows the energy consumed by each slave cluster head for sending parameters to the master cluster head:

$$E_{comm_{slave_{j_{head}}}} = \left(P_{tx_{slave_{j_{head}}}} \times D_{slave_{j_{head}}} \times T_{slave_{j_{head}}}\right)$$

Input:
M: Master cluster devices, S: Slave cluster devices, X: Dataset, Threshold value, α: Sensitivity parameter
Output:
M_g: Final global model parameters
Procedure:
// Initializing:
1. Assign devices to Master Cluster M and Slave Clusters S, ensuring overlap between cluster heads
2. Set sleep state $S_j = 0$ for all devices in Slave Clusters and awake state $S_j = 1$ for all devices Master Cluster
// Master Cluster Learning:
3. Collect the parameters from devices in the Master cluster head
4. Initialize the global model M_g at the Master cluster head:
5. **For** each iteration:
6. Calculate the gradient difference: $\Delta g = \lambda * \left(M_j - M_g\right)$
7. Update the global model: $M_g = M_g - \eta * \left(g_j + \Delta g\right)$
8. Send the initial M_g to all overlapping Slave cluster heads
9. Slave cluster heads wake up their devices and send the initial M_g to their member device
// Local Learning in Slave Clusters:
10. **For** each device in Slave Clusters:
11. Perform the learning process with its own data using the received global M_g
12. Send the updated local parameters to the Master cluster head
13. **End for**
// Global Model Aggregation
14. Master cluster head aggregates the local parameters to create a new global model
15. Compute the difference between the new and previous global models: $\Delta = \alpha \times \left|\left|M_g - M_g'\right|\right|$
// Sleep/Wake Mechanism Updates:
16. **If** $\Delta >$ threshold:
17. Send the new model M_g to all Slave cluster and go to step **5**
18. **Else If** $\Delta \leq$ threshold:
19. Send a sleep signal to all devices in the system: $S_j = 0$ for all $j \in M$ and waiting for new parameters
20. **End if**
// Sleep/Wake Mechanism Updates (Continued):
21. **If** a device joins a cluster:
22. Send its parameters to neighboring devices in master or slaves
23. Repeat from step **5** to update parameters
24. **End if**
25. **If** a device leaves a cluster:
26. Distribute its work among the remaining devices in the corresponding cluster (Slave or Master)
27. Repeat according to the appropriate cluster (step **10** for Slave, step **3** for Master)
28. **End if**
29. **If** a previously disconnected device becomes available again:
30. Receive the current model from nearby devices
31. Re-learn with its own dataset and send the generated parameters to the corresponding adjacent devices
32. Repeat according to the appropriate cluster (step **10** for Slave, step **3** for Master)
33. **End if**
34. Repeat steps **5-9** until the learning process converges
35. **Return** M_g

Fig. 4. Learning process in clusters with sleep/wake-up mechanism.

$$+\left(\mathrm{P}_{\mathrm{rx}_{\mathrm{slave}_{j_{\mathrm{head}}}}} \times \mathrm{D}_{\mathrm{slave}_{j_{\mathrm{head}}}} \times \mathrm{T}_{\mathrm{slave}_{j_{\mathrm{head}}}}\right) \tag{35}$$

4.4.1.4. Total energy consumption. By analyzing the energy consumption of each computational and communicational component and summing them up, the total energy consumption of the proposed system (excluding the energy consumption of the base station) is obtained using (36):

$$\begin{aligned} E_{total} = {} & \Sigma\left(E_{comp_i}\right) + E_{comp_{head}} + \Sigma\left(E_{comp_{slave_j}}\right) + \Sigma\left(E_{comp_{slave_{j_i}}}\right) + \Sigma\left(E_{comm_{i_j}}\right) \\ & + \Sigma\left(E_{comm_{i_{head}}}\right) + \Sigma\left(E_{comm_{head_{slave}}}\right) + \Sigma\left(E_{comm_{slave_{j_{head}}}}\right) \\ & + \Sigma\left(E_{comm_{slave_{j_i}}}\right) \end{aligned} \tag{36}$$

With this introduction, to minimize the total energy consumption of the system, an optimization problem has been formulated by adding constraints and an objective function, which will be discussed in Section IV-D-4.

4.4.2. Federated learning time

The time for Federated Learning includes the computation time for training local parameters and the time to send them to the cluster head or neighboring devices in each iteration. In the learning process, device d trains its local parameters until it reaches a local accuracy (ε_l) and updates them [47]. It is assumed that C_d represents the number of CPU cycles required to process a data sample on device d. The total number of CPU cycles required for one local iteration across all data samples is $C_d \times D_d$. Thus, the computation time for one local iteration on an IoT device is calculated using (37):

$$T_d^{comp} = T_l \frac{C_d\ D_d}{f_d} \tag{37}$$

where T_l is the number of local iterations required to achieve local accuracy, and f_d is the CPU computational speed of the device in cycles per second [62].

The time required for each global iteration is also estimated using (38), considering the longest duration for receiving parameters among all devices and the time for sending them within the clusters.

$$T_d^{comm} = \frac{s}{R_{d_n.z}^d} \tag{38}$$

where s is the size of the data transferred from one device to another during parameter transfer. $R_{d_n.z}^d$ denotes the transfer rate between neighboring devices in the z-th RRB.

Additionally, the system requires applying a time constraint to ensure that the entire learning process, including computations, communications, timely model updates, and coordination between components, is completed within the specified time frame [63]. This issue is known as the federated learning time constraint, defined as T_{FL}, and is formulated according to (39):

$$T_{FL} \geq T_{comp} + T_{comm} \tag{39}$$

where T_{comp} is the total time required for model computations within the clusters, and T_{comm} calculates the overall time spent on communication between cluster devices and between the cluster heads.

4.4.3. Optimizing energy consumption

The optimization problem revolves around resource allocation to minimize energy consumption while meeting performance requirements. The goal is to strike a balance between computational tasks and communication needs to achieve energy efficiency. However, there are constraints that must be considered. For further details, the objective function and its constraints are provided based on works:

P_0: Minimize: E_{total}

s.t.:

$$\mathbf{C1}: \ \Sigma(A_i) \leq R_{total}, \ \forall i \in \text{compute set}$$

$$\mathbf{C2}: \Sigma\left(T_{i_j} \times D_{i_j}\right) \leq B_{total}, \ \forall i.\ j \in S_{i|j}$$

$$\mathbf{C3}: \ A_i \geq P_{comp_{i_{min}}}, \forall i \in \text{compute set}$$

$$\mathbf{C4}: \ A_{head} \geq P_{comp_{head_{min}}}$$

$$\mathbf{C5}: \ A_{slave_j} \geq P_{comp_{slave_{j_m}}}, \ \forall j \in H_{S_i}$$

$$\mathbf{C6}: \ A_{slave_{j_i}} \geq P_{comp_{slave_{j_{i_{min}}}}}, \ j \in H_{S_i}.\ \forall i \in H_{S_j}$$

$$\mathbf{C7}: \ T_{i_j} \geq T_{i_{j_{min}}}, \ \forall i.\ j \in S_{i|j}$$

$$\mathbf{C8}: \ T_{i_{head}} \geq T_{i_{head_{min}}}, \ \forall i \in M$$

$$\mathbf{C9}: \ T_{head_{slave}} \geq T_{head_{slave_{min}}}, \ \forall j \in H_{S_i}$$

$$\mathbf{C10}: \ T_{slave_{j_i}} \geq T_{slave_{j_{i_{min}}}}, \ \forall j \in H_{S_i}, \ \forall i \in H_{S_j}$$

$$\mathbf{C11}: \ T_{i_j} \geq 0, \ \forall i.\ j \in M|S_i$$

$$\mathbf{C12}: \ T_{i_j} = T_{j_i}, \ \forall i.\ j \in M|S_i$$

$$\mathbf{C13}: \ T_{d2d_{i_j}} \leq T_{i_j}, \ \forall i.\ j \in M|S_i$$

$$\mathbf{C14}: \ T_{d2d_{i_j}} + T_{i_j} \leq T_{i_{head}}, \ \forall i \in H_M$$

$$\mathbf{C15}: T_{d2d_{i_j}} + T_{i_j} \leq T_{head_{slave}}, \ \forall i \in S_i \tag{40}$$

The problem P_0 in (40) is recognized as a Linear Programming (LP) problem, where both the objective function and constraints are linear, as they involve linear relationships between variables (the linear combination of energy consumption terms and constraints of inequalities or equalities).

The optimization problem P_0, defined in (40), is identified as a linear programming (LP) problem. This classification is based on the characteristics of LP problems, which include an objective function, decision variables (values to optimize), and constraints. In this case, the goal is to minimize the total energy consumption (E_{total}), which is expressed in (36) as the sum of various energy components. These components can be represented as linear functions of the decision variables: A_i for computational energy and T_{ij} for communication time, defined as (41):

$$E_{total} = \sum_i C_i A_i + c_{head} A_{head} + \sum_j C_{slave_j} A_{slave_j} + \sum_{i,j} d_j T_{i_j} + d_{head} T_{head} + d_{head_{slave}} T_{head_{slave}} \tag{41}$$

Here, c_x and d_x are constants that represent the energy costs associated with each decision variable. These variables are continuous and can take any non-negative value, aligning with the requirements of an LP problem. Additionally, the constraints in this problem, which will be discussed later, are represented as linear inequalities or equalities. For example, constraints C_1 to C_{10} are all inequalities involving the decision variables A_i and T_{ij}, ensuring they are greater than or equal to certain minimum values. They also enforce the non-negativity of the corresponding decision variables. As a result, the presented optimization problem satisfies all the criteria for classification as a linear programming problem. It is worth noting that this problem is solved using PuLP, an open-source linear programming modeling library in Python, and the results will be presented in the performance evaluation section.

The above constraints ensure that each computational component and communication link meets or exceeds the minimum performance requirements, thereby maintaining the expected quality of service. The expressions specified as C1 to C10 impose constraints on the allocation of computational and communication resources and C11 to C15 pertain to issues related to semi-duplex and asynchronous D2D communications. Below, we provide a brief explanation of these conditions.

C1 ensures that the total computational load assigned to all computational components does not exceed the available system capacity. C2 stipulates that the total communication resources used by devices within the cluster should not exceed the available communication bandwidth. This constraint accounts for both the duration and distance between devices. C3 ensures that the power of each computational component is greater than or equal to the minimum required power, guaranteeing that each component operates within the specified power range without energy deficiency.

C4 indicates that the power of the master cluster head is greater than or equal to the minimum required power, setting a lower bound for the energy consumption of the cluster head. C5 guarantees that each slave

cluster head performs computations equal to or greater than its minimum required performance. C6 ensures that the power of each device in the cluster head is greater than or equal to the minimum required power, ensuring that each device consumes at least the minimum amount of energy.

C7 sets the minimum communication time between devices within a cluster and ensures that communication between devices in the cluster occurs for the specified duration. C8 establishes the minimum communication time between devices and the master cluster head, ensuring that communication between devices lasts for the specified duration. C9 specifies the minimum communication time between the master cluster head and slave cluster heads. C10 determines the communication time between a device in the Slave clusters and its adjacent device, which should be equal to or greater than the minimum required performance.

C11 ensures that communication time is non-negative. C12 guarantees that communication delay between two devices is the same regardless of the direction of transmission. C13 ensures that direct communication time between devices is equal to or less than communication time via alternative methods (through a centralized entity), demonstrating that D2D communication can be faster. Finally, C14 and C15 ensure that the total communication time should not exceed the communication time with the main or Slave clusters.

5. Simulation and evaluation

This section presents the evaluation results of the proposed framework. The implementation was carried out using Python 3.9 in the PyCharm development environment. Libraries such as Scikit-learn were used for clustering algorithms, and TensorFlow was employed for tasks related to federated learning. The MNIST dataset and a Deep Neural Network (DNN) model were utilized for data classification tasks. Additionally, the PuLP library was used to solve the LP optimization problem of the mathematical model.

5.1. Simulation setup

5.1.1. Simulation environment and settings

In this simulation, a base station is considered at the center of a circle with a radius of 1 km, accompanied by 40 devices randomly distributed within this area with varying geographical positions and access times (between 1 to 120 s). Each device has a random coverage radius between 10 to 500 m, defining its service area.

For the implementation of the D2D communication channel, parameters such as path loss with a value of $148+40\log_{10}$ (dis. [km]) and cellular path loss with a value of $128.1+37.6\log_{10}$ (dis. [km]) are configured. Table 3 presents the allocated values for the network model parameters and the federated learning model parameters.

5.1.2. Comparison approaches

In this section, five approaches are compared from different perspectives:

- **Star Topology**: In this method, devices send their models directly to the base station for global aggregation.
- **Hierarchical Topology**: In this method, the transfer and reception of model parameters first occur at the intermediate cluster head, and then they are sent to the base station for final aggregation. After global aggregation, each device receives the global model through the same intermediate cluster head.
- **FL-EOCD** approach presented in [47].
- **Proposed Framework (Optimal)**.
- **Proposed Framework (Simulation)**.

Table 3
Summary of parameters for the simulation environment.

Parameter	Value
Time Slot for BS	1–120 sec
IoT Device Coverage Radius	10–500 m
Number of IoT Devices	10–70
Radius of BS	1 km
Maximum Power of IoT Device	3 w
Maximum Bandwidth of BS	10 Mb/s
Number of data samples	200 - 1000
Learning rate	0.001
Convergence threshold rate (α)	0.1
Learning algorithm	Fedprox
Regularization parameter (λ)	0.01
FL time threshold	2 sec
Total Energy Capacity of IoT Devices	500–2000 j

5.2. Results and discussion

In this section, numerical results obtained from the simulation are presented. To compare the proposed framework, we selected three performance metrics: (1) Energy Consumption, (2) Learning Accuracy, (3) Federated Learning Time.

5.2.1. Energy consumption

Fig. 5 shows the total energy consumption of the network as the number of Slave clusters increases in the proposed framework. In this scenario, 40 devices are distributed across the environment. As can be observed, with an increase in the number of Slave clusters (and consequently a decrease in the size of each cluster), the energy consumption in the proposed framework decreases.

Fig. 6 illustrates the energy consumption of the five evaluated approaches as a function of the number of devices in the network. As shown in the figure, for lower numbers of devices, the FL-EOCD approach performs slightly better than the proposed framework. However, in networks with a higher number of devices, the proposed framework demonstrates superior performance in terms of energy consumption.

Fig. 7 shows the energy consumption versus the number of data samples for the five compared approaches. As the sample size increases, the energy consumption also rises. The proposed framework demonstrates lower energy consumption compared to existing methods for all sample sizes. This is primarily because the proposed approach reduces communication overhead due to the connections between neighboring devices. In contrast, the star and hierarchical approaches generally consume higher energy due to higher communication and computational costs. However, the hierarchical approach, compared to the star approach, benefits from its structure, providing more efficient parameter aggregation and thereby reducing energy consumption.

5.2.2. Learning accuracy

Fig. 8 illustrates the accuracy of the proposed learning approach

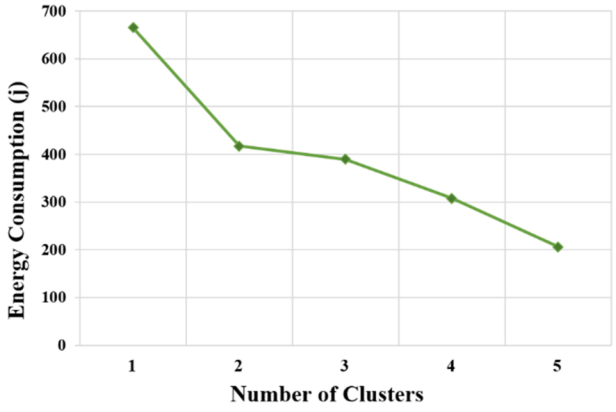


Fig. 5. Energy consumption analysis relative to the number of slave-clusters.

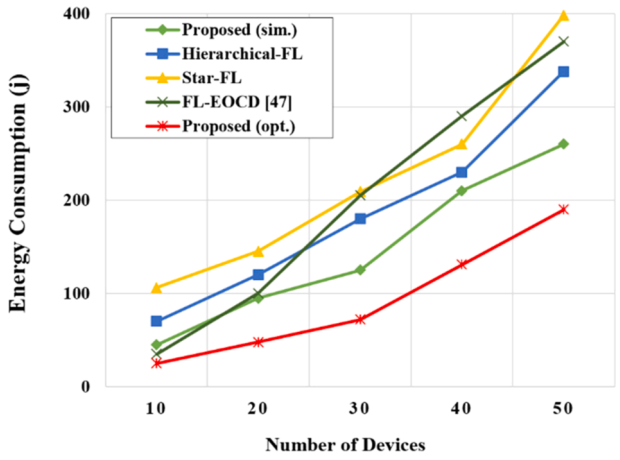


Fig. 6. Comparison of energy consumption in a global iteration.

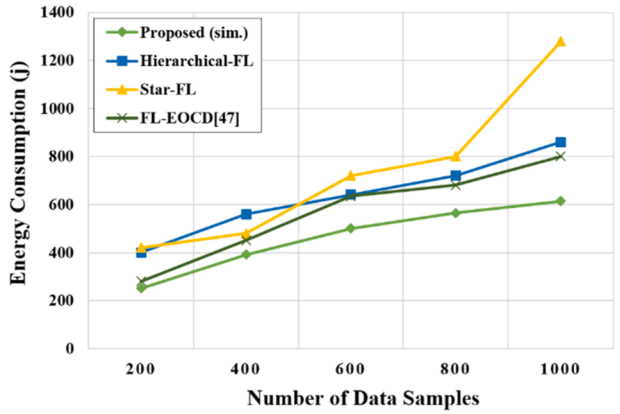


Fig. 7. Comparison of energy consumption against the number of devices.

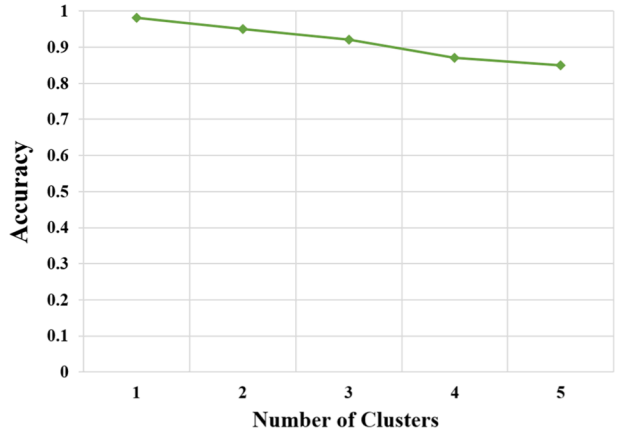


Fig. 8. Evaluation of learning accuracy against the number of slave-clusters.

against different numbers of slave clusters. As shown, the accuracy of the proposed approach decreases with an increase in the number of clusters. This decline in accuracy can be attributed to two main reasons. First, as the number of clusters increases and consequently the size of each cluster decreases, the amount of training data within each cluster also decreases. Second, there is an increase in feature diversity among diff8erent clusters, which can negatively impact the overall accuracy.

Fig. 9 displays the accuracy of the compared approaches. It can be observed that the main/slave clustering mechanism used in the proposed framework results in improved accuracy. In contrast, the star and

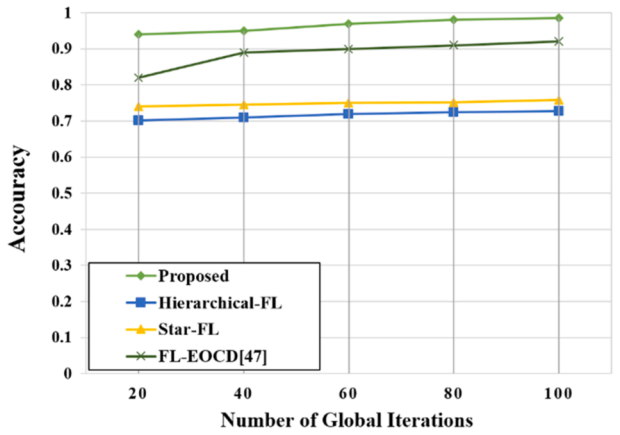


Fig. 9. Comparison of learning accuracy against the number of global iterations.

hierarchical approaches with centralized structures demonstrate lower accuracy. The FL-EOCD approach, with its chain-like clustering and use of SGD, achieves higher accuracy compared to the star and hierarchical methods. However, it still falls short of the accuracy achieved by the proposed approach.

The observed drop in the accuracy of existing approaches can be linked to several key factors, such as suboptimal model configurations and experimental setups (as shown in Table 3). Specifically, decisions related to hyperparameters (like learning rate, convergence threshold, and regularization techniques) and even insufficient training iterations play a big role in influencing model performance. These factors often lead to challenges like poor convergence or overfitting. Additionally, the centralized structures commonly used in star and hierarchical models may disrupt effective communication between devices, worsening issues related to data heterogeneity, which overall impacts model accuracy. On the other hand, the proposed approach takes a decentralized structure and incorporates a specific clustering mechanism. This facilitates better collaboration within clusters, effectively addressing the limitations of centralized models. As a result, it improves communication efficiency, speeds up model convergence, and ultimately enhances accuracy and overall performance.

5.2.3. FL time

In Fig. 10, the learning times of the methods are compared. Here, the FL-EOCD method requires the least amount of time for a lower number of devices. This is due to the presence of scheduled devices and coordinated RRB resource allocation, which streamline the learning process. These features are not relied upon by the three approaches of the

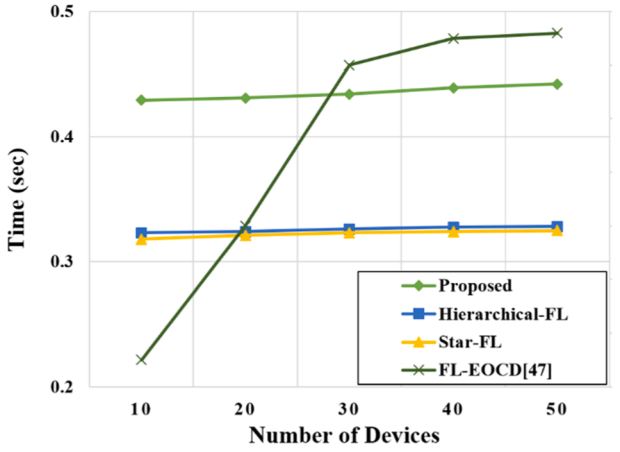


Fig. 10. Comparison of FL time in a global iteration.

proposed method, star topology, and hierarchical topology, where communications are asynchronous. However, as the number of devices increases, due to differences in the clustering approach between the FL-EOCD method and the proposed method, the learning time in the proposed approach slightly improves. In summary, the unscheduled design in the proposed scheme may cause initial inefficiencies in learning time, but it ultimately fosters greater adaptability to changing network conditions and allows devices to adjust their contributions based on real-time availability.

6. Conclusions and future works

In this research, a comprehensive framework has been proposed to leverage the benefits of decentralized federated learning in the Internet of Things (IoT) environment, focusing on improving energy consumption. In this approach, devices can exchange model information and learning parameters by utilizing Device-to-Device (D2D) communications without the need for a central server. In the proposed framework, by implementing a sleep/wake-up mechanism, clusters with a number of devices not suitable for collaborative learning can enter a sleep mode and receive suitable parameters from a cluster with appropriate conditions, referred to as the master cluster. Subsequently, they update their local parameters and send them to the master cluster. Overall, these mechanisms contribute to energy efficiency and can address mobility challenges in this environment. Finally, to evaluate the proposed approach, the results obtained from implementing and running the method are compared with the optimal values obtained from LP, as well as star, hierarchical, and FL-EOCD methods in terms of energy efficiency, learning time, and accuracy. The results indicate that the proposed method, with its distinct structure and parameters, effectively reduces energy consumption and demonstrates superiority in accuracy compared to other evaluated methods. However, its learning time increases based on specific features in the given scenario.

For future work on optimizing energy consumption within the proposed framework, two main areas have been identified:

- Compression Methods: Future research should investigate advanced IoT-focused compression techniques, such as quantization and model compression algorithms like Distillation. Efficient data compression can substantially reduce communication energy demands and enhance overall energy efficiency.
- Energy-Aware Learning Parameters: Optimizing learning parameters for energy efficiency is crucial. Future studies could develop energy-aware optimization techniques that adjust learning parameters based on energy availability and device constraints. For example, algorithms could dynamically modify the learning rate, convergence threshold, or regularization parameters to minimize energy consumption. Addressing these areas could lead to reduced energy usage and shorter learning times.

Additionally, future research should address the limitations of the proposed framework, including potential scalability challenges in large-scale deployments, such as expanding the mobility model, and the need for robust security measures, to gain a better understanding of the practical implications of this approach.

CRediT authorship contribution statement

Nastooh Taheri Javan: Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology. **Elahe Zakizadeh Gharyeali:** Writing – original draft, Investigation, Data curation. **Seyedakbar Mostafavi:** Investigation, Formal analysis, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] D.C. Nguyen, et al., 6G Internet of Things: a comprehensive survey, IEEE Internet Things J. 9 (1) (2022) 359–383.
[2] A. Hakiri, A. Gokhale, S.B. Yahia, N. Mellouli, A comprehensive survey on digital twin for future networks and emerging Internet of Things industry, Comput. Netw. 244 (2024) 110350.
[3] S. Messaoud, A. Bradai, S.H.R. Bukhari, Ph.T.A. Quang, O.B. Ahmed, M. Atri, A survey on machine learning in Internet of Things: algorithms, strategies, and applications, Internet Things 12 (2020) 100314.
[4] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, A.S. Avestimehr, Federated learning for the internet of things: applications, challenges, and opportunities, IEEE Internet Things Mag. 5 (1) (2022) 24–29.
[5] H. Zhou, Y. Zheng, X. Jia, Towards robust and privacy-preserving federated learning in edge computing, Comput. Netw. 243 (2024) 110321.
[6] L.U. Khan, W. Saad, Z. Han, E. Hossain, C.S. Hong, Federated learning for Internet of Things: recent advances, taxonomy, and open challenges, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1759–1799.
[7] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H. Vincent Poor, Federated learning for Internet of Things: a comprehensive survey, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1622–1658.
[8] E.T. Martínez Beltrán, et al., Decentralized federated learning: fundamentals, state of the art, frameworks, trends, and challenges, IEEE Commun. Surv. Tutor. 25 (4) (2023) 2983–3013.
[9] L. Palmieri, Ch. Boldrini, L. Valerio, A. Passarella, M. Conti, Impact of network topology on the performance of decentralized federated learning, Comput. Netw. 253 (2024). Art. no. 110681.
[10] L. Yuan, Z. Wang, L. Sun, P.S. Yu, C.G. Brinton, Decentralized federated learning: a survey and perspective, IEEE Internet Things J. (2024), https://doi.org/10.1109/JIOT.2024.3407584.
[11] H. Xie, M. Xia, P. Wu, S. Wang, K. Huang, Decentralized federated learning with asynchronous parameter sharing for large-scale IoT networks, IEEE Internet Things J. (2024), https://doi.org/10.1109/JIOT.2024.3354869.
[12] B. Li, W. Gao, J. Xie, M. Gong, L. Wang, H. Li, Prototype-based decentralized federated learning for the heterogeneous time-varying IoT systems, IEEE Internet Things J. 11 (4) (2024) 6916–6927.
[13] A. Salama, A. Stergioulis, S.A.R. Zaidi, D. McLernon, Decentralized federated learning on the edge over wireless mesh networks, IEEE Access 11 (2023) 124709–124724.
[14] T. Chen, X. Zhang, M. You, G. Zheng, S. Lambotharan, Federated learning enabled link scheduling in D2D wireless networks, IEEE Wirel. Commun. Lett. 13 (1) (2024) 89–92.
[15] D. Wu, W. Yang, H. Jin, X. Zou, W. Xia, B. Fang, FedComp: a federated learning compression framework for resource-constrained edge computing devices, IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 43 (1) (2024) 230–243.
[16] O.R.A. Almanifi, Ch.O. Chow, M.L. Tham, J.H. Chuah, J. Kanesan, Communication and computation efficiency in federated learning: a survey, Internet Things 22 (2023) 100742.
[17] J. Fan, K. Wu, G. Tang, Y. Zhou, S. Huang, Taking advantage of the mistakes: rethinking clustered federated learning for IoT anomaly detection, IEEE Trans. Parall. Distribut. Syst. 35 (6) (2024) 862–876.
[18] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proc. Mach. Learn. Syst. 2 (2020) 429–450.
[19] M. Sharma, P. Kaur, Reliable federated learning in a cloud-fog-IoT environment, J. Supercomput. 79 (14) (2023) 15435–15458.
[20] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.
[21] Y. Liu, et al., Vertical federated learning: concepts, advances, and challenges, IEEE Trans. Knowl. Data Eng. 36 (7) (2024) 3615–3634.
[22] B. Liu, N. Lv, Y.G.Y. Li, Recent advances on federated learning: a systematic survey, Neurocomputing 597 (2024) 128019.
[23] W. Ali, I.U. Din, A. Almogren, J.J.P.C. Rodrigues, Federated learning-based privacy-aware location prediction model for internet of vehicular things, IEEE Trans. Veh. Technol. (2024), https://doi.org/10.1109/TVT.2024.3368439.
[24] S. Lee, J. Sung, M.-K. Shin, Layer-wise personalized federated learning for mobile traffic prediction, IEEE Access 12 (2024) 53126–53140.
[25] S. Niknam, H.S. Dhillon, J.H. Reed, Federated Learning for wireless communications: motivation, opportunities, and challenges, IEEE Commun. Mag. 58 (6) (2020) 46–51.
[26] S. Durand, K. Khawam, D. Quadri, S. Lahoud, S. Martin, Federated learning game in IoT edge computing, IEEE Access 12 (2024) 93060–93074.
[27] I. Kavasidis, E. Lallas, G. Mountzouris, V.C. Gerogiannis, A. Karageorgos, A federated learning framework for enforcing traceability in manufacturing processes, IEEE Access 11 (2023) 57585–57597.

[28] M. Abaoud, M.A. Almuqrin, M.F. Khan, Advancing federated learning through novel mechanism for privacy preservation in healthcare applications, IEEE Access 11 (2023) 83562–83579.
[29] S.I. Manzoor, S. Jain, Y. Singh, H. Singh, Federated learning based privacy ensured sensor communication in IoT networks: a taxonomy, threats and attacks, IEEE Access 11 (2023) 42248–42275.
[30] M. Gecer, B. Garbinato, Federated learning for mobility applications, ACM Comput. Surv. 56 (5) (2024) 1–28.
[31] H. Gao, M.T. Thai, J. Wu, When decentralized optimization meets federated learning, IEEE Netw. 37 (5) (2023) 233–239.
[32] T.V. Nguyen, N.D. Ho, H.T. Hoang, C.Danh Do, K.-S. Wong, Toward efficient hierarchical federated learning design over multi-hop wireless communications networks, IEEE Access 10 (2022) 111910–111922.
[33] L. Li, D. Shi, R. Hou, H. Li, M. Pan, Z. Han, To talk or to work: flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices, in: Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM 2021), Vancouver, BC, Canada, 2021, pp. 1–10.
[34] A.A. Abdellatif, et al., Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data, Future Gener. Comput. Syst. 128 (2022) 406–419.
[35] B. Luo, X. Li, S. Wang, J. Huang, L. Tassiulas, Cost-effective federated learning in mobile edge networks, IEEE J. Sel. Areas Commun. 39 (12) (2021) 3606–3621.
[36] L. Cui, X. Su, Y. Zhou, Y. Pan, Slashing communication traffic in federated learning by transmitting clustered model updates, IEEE J. Sel. Areas Commun. 39 (8) (2021) 2572–2589.
[37] W.Y.B. Lim, et al., Decentralized Edge Intelligence: A dynamic resource allocation framework for hierarchical federated learning, IEEE Trans. Parall. Distribut. Syst. 33 (3) (2022) 536–550.
[38] Z. Yang, M. Chen, W. Saad, C.S. Hong, M. Shikh-Bahaei, Energy efficient federated learning over wireless communication networks, IEEE Trans. Wirel. Commun. 20 (3) (2021) 1935–1949.
[39] V.D. Nguyen, S.K. Sharma, T.X. Vu, S. Chatzinotas, B. Ottersten, Efficient federated learning algorithm for resource allocation in wireless IoT networks, IEEE Internet Things J. 8 (5) (2021) 3394–3409.
[40] W. Wu, L. He, W. Lin, R. Mao, C. Maple, S. Jarvis, SAFA: a semi-asynchronous protocol for fast federated learning with low overhead, IEEE Trans. Comput. 70 (5) (2021) 655–668.
[41] F. Liang, Q. Yang, R. Liu, J. Wang, K. Sato, J. Guo, Semi-synchronous federated learning protocol with dynamic aggregation in internet of vehicles, IEEE Trans. Veh. Technol. 71 (5) (2022) 4677–4691.
[42] B. Gong, T. Xing, Z. Liu, J. Wang, X. Liu, Adaptive clustered federated learning for heterogeneous data in edge computing, Mob. Netw. Appl. 27 (4) (2022) 1520–1530.
[43] S. Luo, X. Chen, Q. Wu, Z. Zhou, S. Yu, HFEL: joint edge association and resource allocation for cost-efficient hierarchical federated edge learning, IEEE Trans. Wirel. Commun. 19 (10) (2020) 6535–6548.
[44] L. Yu, R. Albelaihi, X. Sun, N. Ansari, M. Devetsikiotis, Jointly optimizing client selection and resource management in Wireless federated learning for Internet of Things, IEEE Internet Things J. 9 (6) (2022) 4385–4395.
[45] S.A. Khowaja, K. Dev, P. Khowaja, P. Bellavista, Toward energy-efficient distributed federated learning for 6G networks, IEEE Wirel. Commun. 28 (6) (2021) 34–40.
[46] F.P.-C. Lin, S. Hosseinalipour, S.S. Azam, C.G. Brinton, N. Michelusi, Semi-decentralized federated learning with cooperative D2D local model aggregations, IEEE J. Sel. Areas Commun. 39 (12) (2021) 3851–3869.
[47] M.S. Al-Abiad, M. Obeed, Md.J. Hossain, A. Chaaban, Decentralized aggregation for energy-efficient federated learning via D2D communications, IEEE Trans. Commun. 71 (6) (2023) 3333–3351.
[48] M.S. Al-Abiad, Md.J. Hossain, Coordinated scheduling and decentralized federated learning using conflict clustering graphs in fog-assisted IoD networks, IEEE Trans. Veh. Technol. 72 (3) (2023) 3455–3472.
[49] S. Hosseinalipour, C.G. Brinton, V. Aggarwal, H. Dai, M. Chiang, From federated to fog learning: distributed machine learning over heterogeneous wireless networks, IEEE Commun. Mag. 58 (12) (2020) 41–47.
[50] A. Taïk, Z. Mlika, S. Cherkaoui, Clustered vehicular federated learning: process and optimization, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 25371–25383.
[51] Y.M. Saputra, D.N. Nguyen, D.T. Hoang, T.X. Vu, E. Dutkiewicz, S. Chatzinotas, Federated learning meets contract theory: economic-efficiency framework for electric vehicle networks, IEEE Trans. Mob. Comput. 21 (8) (2022) 2803–2817.
[52] K.N. Dattatraya, K.R. Rao, Hybrid based cluster head selection for maximizing network lifetime and energy efficiency in WSN, J. King Saud Univ. Comput. Inf. Sci. 34 (3) (2022) 716–726.
[53] S. Zahri, H. Bennouri, A. Chehri, A.M. Abdelmoniem, Federated learning for IoT networks: enhancing efficiency and privacy, in: Proceedings of the 9th IEEE World Forum on Internet of Things (WF-IoT), Aveiro, Portugal, 2023, pp. 1–6.
[54] J. Pei, W. Liu, J. Li, L. Wang, C. Liu, A review of federated learning methods in heterogeneous scenarios, IEEE Trans. Consum. Electr. (2024), https://doi.org/10.1109/TCE.2024.3385440.
[55] H. Ye, L. Liang, G.Y. Li, Decentralized federated learning with unreliable communications, IEEE J. Sel. Top. Signal Process. 16 (3) (2022) 487–500.
[56] S. Regilan, L.K. Hema, Optimizing environmental monitoring in IoT: integrating DBSCAN with genetic algorithms for enhanced clustering, Int. J. Comput. Appl. 46 (1) (2024) 21–31.
[57] D. Vorobyova, et al., IoT network model with multimodal node distribution and data-collecting mechanism using mobile clustering nodes, Electronics 12 (6) (2023) 1410.
[58] X. Yu, L. Li, X. He, S. Chen, L. Jiang, Federated learning optimization algorithm for automatic weight optimal, Comput. Intell. Neurosci. 2022 (2022) 8342638.
[59] Y. Ruan, C. Joe-Wong, FedSoft: soft clustered federated learning with proximal local updating, Proc. AAAI Conf. Artif. Intell. 36 (7) (2022) 8124–8131.
[60] M.H. Mahmoud, A. Albaseer, M. Abdallah, N. Al-Dhahir, Federated learning resource optimization and client selection for total energy minimization under outage, latency, and bandwidth constraints with partial or No CSI, IEEE Open J. Commun. Soc. 4 (2023) 936–953.
[61] M.S. Al-Abiad, M.Z. Hassan, M.J. Hossain, Energy efficient distributed learning in integrated fog-cloud computing enabled IoT networks, in: Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops), Seoul, Korea, 2022, pp. 872–877.
[62] Y. Li, et al., Energy-constrained D2D assisted federated learning in edge computing, in: *Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '22)*, 2022, pp. 33–37.
[63] X. Zhou, J. Zhao, H. Han, C. Guet, Joint optimization of energy consumption and completion time in federated learning, in: Proceedings of the IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), Bologna, Italy, 2022, pp. 1005–1017.