Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in ICT Internet and Multimedia Engineering

Resource Allocation through Auction-based Incentive Scheme for Federated Learning in Mobile Edge Computing

Master Candidate

Jawad Asif

Student ID 2013002

Supervisor

Prof. Frank Michael Schleif

Technical University of Applied Sciences Würzburg-Schweinfurt

Co-supervisor

Prof. Pietro Zanuttigh

Universita di Padua

Academic Year
2022/2023

Abstract

Organizations are increasingly engaging in collaborative networks, pooling resources, and managing supply chains. When it involves the transportation of small quantities of physical goods, sharing downtime among companies within the sharing economy, always presents logistical challenges. This thesis has been dedicated for MEC devices acquiring multi-dimensional resources to work in an AI-driven logistics network that autonomously enhances intralogistics in industrial areas, facilitating the efficient movement of goods between companies using aerial means.

Given the heterogeneous nature of the environment and the varying characteristics of devices, this necessitates the utilization of a federated learning framework, allowing each device to train its local model on its proprietary dataset and select the top-performing nodes based on the quality of their shared resources to perform a task. To make sure every nodes receives a fair distribution of reward, we have used auction based incentive schemes that assure the optimal payment for each node taking part in the bidding process.To achieve this goal, I conducted a comprehensive assessment of our proposed auction mechanisms and bidding strategies using numerical analysis. Our approach has demonstrated superior performance compared to the traditional federated learning strategy, FedAvg, by achieving earlier convergence in loss and reducing the number of required rounds

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

GT	Game Theory
BR	Best Response
NE	Nash Equilibrium
BNE	Bayesian Nash Equilibrium
ML	Machine Learning
FL	Federated Learning
DTS	Driver-less transport system
UAVs	Unnamed aerial vehicles
ANN	Artificial Neural Network
SGD	Stochastic Gradient descent
FedAvg	Federated Averaging
FedDyn	Federated Learning with Dynamic Regularization
IID	Independent and Identical distribution
RA	Resource Allocation
HetNets	Heterogeneous Networks
IR	Individual Rationality

1 Introduction

Advancement in businesses and digitalization in industry, todays logistics sector is increasingly confronting with the need to transport small size goods with low cost. Conventional logistics systems, costs more when shipping small size packages. The design of logistic system is always based on the principle of minimum effort. Therefore, it is needed to develop a logistics network based on Artificial Intelligence (AI) approaches, that organize itself and optimize the intralogistics production by means of micro-mobility of goods across companies, by land and by air.

Modern technologies like driverless transport systems (DTS) and unmanned aerial vehicles (UAVs) have great potential, but they're not well-connected. Companies work together more and more, sharing resources and managing supply chains across their boundaries. This sharing economy creates logistical challenges when it comes to sharing small physical goods. Therefore, the aim of the thesis is to establish an relaiable autunomous system where devices can share their resources to perform a task.

1.1 Background and Research Problem

Currently, the percentage of truck which runs empty and minimal loads is well over 37.1%, which force to customize logistics systems. Therefore, industries requires to develop collaboratively, cooperation of different transport units such as DTS and UAVs to enable the determine use of these tools in heterogenous network and it should allow logistics units to communicate with each other and determine the most suitable candidate for the transportation task, at hand. The problem is quite complex and can be solved with the help of Artificial intelligence. Due to the autonomously and decentrally acting transport units, are trained by a machine learning model executed on the device level, which they share with other network participants in order to improve the overall model. To be able to technically map these requirements we need an approach which should be able to leverage large amounts of decentrally organized data and process it locally on each node performing task.

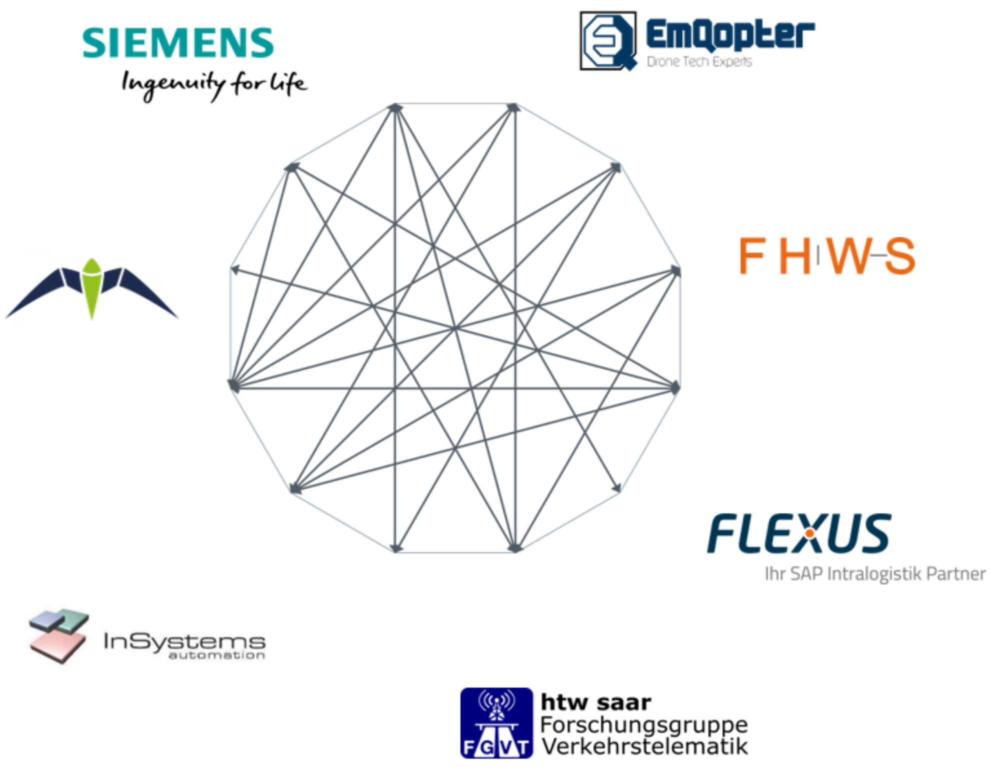


Figure 1.1: Companies Involved.

The optimal solution can be divided into two parts. Frist applying machine learning approaches to train the units in heterogenous environment and secondly, applying game theory concept to encourage nodes to participate in the network to provide quality resources in the multi-dimensional environment to do a hand-on task. Fortunately, edge nodes equipped with powerful computing capability, sufficient Flash storage, etc., accelerate the adoption of local data processing. This allows us to adopt the Mobile Edge Computing (MEC) architecture, that enables edge nodes to locally collect and process various data with the remote cloud coordination, which specially appeals to the Internet of Things

(IoT), social networking, 5G etc.

1.2 Novel solution

Ongoing research at the Technical University of Applied Sciences, known as FlowPro, explores AI-supported inspection drones [37, 44] and the development of a logistics network for autonomous ground and air-based transport units within 5G test environments. This network facilitates cross-company production and streamlined goods movement in future supply chains, particularly for small packages. Concurrently, a team is creating a secure, decentralized AI system that autonomously manages adaptive bidding in logistics through federated learning from various data sources. This thesis, part of the FlowPro initiative, focuses on establishing bidding strategies to fully automate and cost-effectively helps to operate the logistics system.

Therefore, our approach employs a bidding model designed to incentivize high-quality edge nodes to participate in collaborative learning efficiently and cost-effectively, ultimately enhancing overall performance. To realize this objective, we expanded upon the multi-dimensional procurement auction model and introduced a novel Incentive Mechanism tailored for multi-dimensional auctions integrated with federated learning.

For practical implementation of this solution, we chose to utilize the Flower framework[1]. This framework simplifies the implementation of a single aggregator responsible for initializing the process and requesting resources from currently available transportation unit nodes. As a result, we can optimize both local and global logistics objectives, achieving cost reduction, emission reduction, faster delivery times, and increased system capacity.

1.3 Thesis organization

In this chapter, we have provided a introduction to the incentive solution for the existing problem and federated learning. Additionally, we offered a concise overview of the scope of this work. The subsequent chapters are structured as follows:

[1]`https://flower.dev/`

- In Chapter 2, we'll delve into foundational concepts essential to our thesis: auction theory, resource allocation, federated learning, and game theory . This chapter lays the groundwork for our subsequent research..
- In Chapter 3, we will conduct a thorough review of the academic literature and research pertinent to our current thesis. We will analyze and synthesize the significant findings, methodologies, and theories from previous studies in this comprehensive examination.
- In Chapter 4, we will showcase our approach's ability to attract these valuable edge nodes to share their quality resources to perform the task announced by server.
- In Chapter 5, we put the previously discussed strategy into action by implementing it on both synthetic and MNIST datasets. This chapter provides an in-depth exploration of these datasets, elucidates the preprocessing steps employed, defines performance metrics, and offers a detailed analysis of the results derived from the implementation.
- Chapter 6 addresses the limitations of our work, explores potential avenues for future research, and concludes the thesis.

2 Background

In this chapter, basic concept related to current thesis will be explained. Since game theory, auction and federated learning are taking major part in the thesis, so we will break down the key terms related to these topics. Understanding these concepts will help to set the groundwork, which will set the stage for the rest of the thesis, where we will see how combining game theory and federated learning has led to our research findings.

2.1 Game Theory

Game theory (GT) is a branch of mathematics that studies strategic interactions between multiple decision-makers, known as players, who aim to optimize their outcomes. It provides a framework to analyze various scenarios and model the behavior of rational agents in situations of conflict and cooperation. A game consists of finite players (N) who interact based on rules. Players can be individuals, groups, or devices. The game is described by the players and their strategies (S), representing their possible actions ($a \in A$). In book [18], the author define the game theory as: The subject of game theory are situations, where the result for a player does not only depend on his own decisions, but also on the behaviour of the other players.

In our case, the players represent nodes participating in the training process. Each node's objective is to maximize its payoff, which can be represented by various metrics such as model accuracy, convergence speed, or resource utilization. To achieve this, each node selects strategies during the training process. [19].

2.1.1 Utility Function and Best Response

In game theory, a payoff or *utility function* represents the outcome that a player receives as a result of their chosen strategy in a game in numerical form. It allows us to evaluate the desirability of different outcomes and to understand the players' motivations for choosing specific strategies. If the number of node is limited to two and if their sets of strategies consist of only a few elements, the outcome of the payoff function can be represented in a matrix, this so-called *payoff matrix* [19].

A *Best Response* (BR) refers to a strategy that maximizes a player's payoff given the strategies chosen by the other players in the game. By definition it is the strategy (or strategies) which produces the most favorable outcome for a player, taking other players' strategies as given. For utility function: $a_i \in A_i$ is a best response to $a_{-i} \in A_{-i}$ if:

$$u_i(a_i, a_{-i}) \geq u_i(a'_i, a_{-i}) \quad \forall a_i \in A_i \tag{2.1}$$

In GT, incentives are motivations that drive individuals or entities to make certain decisions or take specific actions within a strategic interaction. These incentives can influence how players behave in a game, affecting the outcomes and strategies they choose. Game theory seeks to understand how different incentive structures lead to various equilibrium points in games [31]. One of the common type of incentive in GT is *Nash Equilibrium*.

2.1.2 Bayesian Games: Imperfect Information

In GT, when players are uncertain about each other's characteristics or resources, the situation can be modeled as a Bayesian game. In a Bayesian game, players' uncertainty is captured using probability measures over a set of possible states. The concept of Bayesian games allows for more realistic modeling of real-world situations where players lack complete information about each other [35].

For instance, consider a scenario where nodes in a network are involved in a resource-sharing problem, but they are uncertain about the resources possessed by other nodes. This uncertainty can be effectively captured using a Bayesian game framework. Each node represents a player with private information (such as the amount of available resources), and their actions depend on their beliefs about other players' characteristics [23]. In this setting, the notion of Nash equilibrium still plays a crucial role. An Imperfect Information Nash Equilibrium represents a solution concept in which players' strategies are optimal given their beliefs and the strategies of others.

2.1.3 Nash Equilibrium

Bayesian Nash equilibrium (BNE) captures the strategic interactions in games with incomplete information, where players make decisions based on their private knowledge and beliefs about others, leading to more nuanced and realistic outcomes in real-world situations where information is uncertain or imperfect [17]. The type of a player may include private values, beliefs, or preferences that are not known to others. Players have beliefs about the distribution of other player's types based on available information or observations.

Let (S, f) be a game with n players, where S_i is the strategy set for player i, $S = S_1 \times S_2 \cdots \times S_n$ is the set of strategy profiles and $f = (f_1(x), \ldots, f_n(x))$ is the payoff function for $x \in S$. Let x_i be a strategy profile of player i and $x_{-}i$ be a strategy profile of all players except for player i. When each player $i \in 1, \ldots, n$ chooses strategy x_i resulting in strategy profile $x = (x_1, \ldots, x_n)$ then player i obtains payoff $f_i(x)$. Note that the payoff depends on the strategy profile chosen, i.e., on the strategy chosen by player i as well as the strategies chosen by all the other players. A strategy profile $x^* \in S$ is a Nash equilibrium (NE) if no unilateral deviation in strategy by any single player is profitable for that player [33, 32], that is

$$\forall i, x_i \in S_i, x_i \neq x^*{}_i : f_i(x^*{}_i, x^*_{-i}) \geq f_i(x_i, x^*_{-i}) \tag{2.2}$$

In brief, in a NE of a Bayesian game, each player chooses the best action available to him given the signal that he receives and his belief about the state and the other players actions that he deduces from this signal. In short NE is a choice of moves a^* where no player can improve their position by choosing a different move while the others pick the same strategy.

Pareto-Dominated & Pareto-Optimal

One idea is to concentrate on Pareto-optimal Nash equilibrium only. A Nash equilibrium is *Pareto-dominated* by another Nash equilibrium if every players payoff in the first one is smaller or the same as in the second one. Nobody would object to move to the second Nash equilibrium. A Nash equilibrium is *Pareto-optimal* if it is not Pareto-dominated by any other Nash equilibrium, except maybe by some having exactly the same payoffs [39].

2.1.4 Incentives or Rewards

Incentives or rewards are mechanisms used to motivate individuals or entities to take specific actions or make certain decisions. They are designed to influence behavior by offering benefits, advantages, or positive outcomes in exchange for performing desired actions. Incentives can take various forms, such as monetary rewards, bonuses, recognition, privileges, discounts, or any other valuable outcome that encourages individuals to align their actions with a particular goal or objective [5].

2.1.5 Rational Player

A rational player, often known as a rational agent, makes decisions based on logical and self-interested evaluations of information and outcomes. In game theory, they aim to maximize personal gain by assessing options and their probabilities, considering costs and benefits, and aligning choices with preferences.In many scenarios, incentives or rewards are used to influence the decisions of rational players, aligning their choices with desired outcomes or objectives.

2.2 Auction

An auction is a market mechanism or process where goods, services, or assets are bought and sold through competitive bidding by multiple participants called bidders. It allows to compete bidders against each other to win the item being auctioned, and the winner is typically the bidder who submits the highest bid. The bid submitted by a bidder represents the price they are willing to pay for the item. However, the objective is not only win the auction but also to maximize their utility, which depends on the valuation they have for the item [39]. Auction GT explores various types of auctions, one of which is First Price auction, which we will use in our thesis.

2.2.1 First Price (sealed bid) Auction

first price auction, bidders submit sealed bids $b_1, \ldots, b_n$.The bidders who submits the highest bid is awarded the object, and pays his bid. According to sealed bid, it should be clear player does not want to share their information to others and it should only be to auctioneer. For this purpose Envelope theorem is use for solving for symmetric equilibrium bidding strategies.

Envelope Theorem

A closely related, and often convenient, approach to identify necessary conditions for a symmetric equilibrium is to exploit the envelope theorem. To this end, suppose $b(s)$ is a symmetric equilibrium in increasing differentiable strategies [39]. Then $i^{'}s$ equilibrium payoff given signal s_i is a best-response in equilibrium:

$$U(s_i) = \max_{bi}(s_i\text{-}b_i)F^{n\text{-}1}(b^{\text{-}1}(b_i)) \tag{2.3}$$

2.2.2 Independent Private Model

An independent private value (IPV) model, refers to a scenario where each bidder's valuation for the item being auctioned is private and independent of the valuations held by other bidders. In simpler terms, each bidder assigns a subjective value to the item based solely on their own preferences and information, without knowing how other bidders value the item [30].

2.2.3 A model of Multidimensional Auction

A model of a multidimensional auction is a framework that describes the rules, strategies, and outcomes of an auction where multiple attributes or dimensions are involved in the bidding and allocation process. In traditional auctions, such as single-item auctions, participants bid on a single parameter (usually price) to determine the winner. In multidimensional auctions, bidders consider multiple attributes, parameters, or criteria in their bids, which can include both price and other characteristics of the items being auctioned.The goal is to allocate items to bidders in a way that maximizes efficiency, fairness, or other specific objectives [6]. This relates to thesis, where different nodes having different characteristics or resources.

2.3 Federated learning

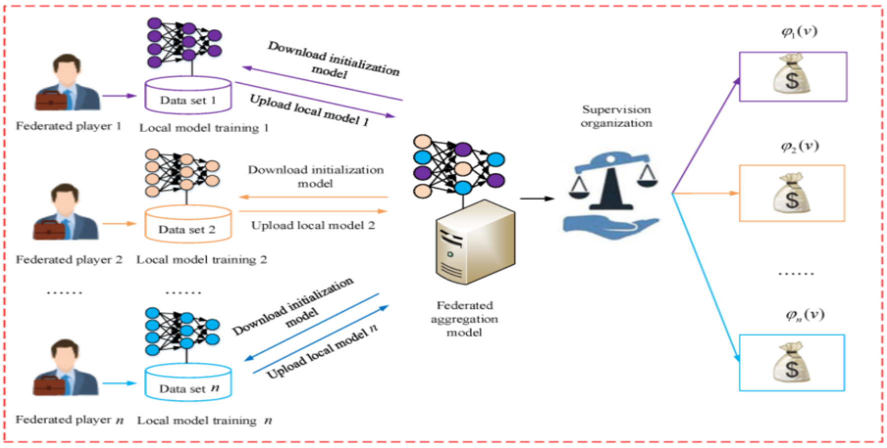


Figure 2.1: FL Incentive Model [42]

Federated learning is a machine learning approach that allows models to be trained across multiple decentralized devices or servers while keeping the data localized on those devices or servers. In this approach, the model is sent to individual devices, such as smartphones, IoT devices, or local servers, where they perform training using their local data, which involves abundant, often confidential, data accessibility. Models constructed from the data hold which potentially enhanced usability in advanced applications. However, the sensitive nature of the data and model introduces risks and obligations when stored centrally.[18, 16].

Federated Learning is an innovative approach to machine learning that enables devices to collectively train a shared model while maintaining data privacy and decentralization. In this paradigm, devices like smartphones, IoT devices, and edge servers independently perform local model updates using their own data, eliminating the need to share raw data. These local updates are then sent to a central server, which aggregates them to create a global model that benefits from the insights of all participating devices. This decentralized nature not only safeguards data privacy but also enhances privacy preservation by avoiding data centralization. Additionally, federated learning optimizes bandwidth usage as communication involves sharing model parameters rather than transmitting raw data. This approach finds applications in diverse fields where data distribution, privacy, and connectivity constraints are prevalent, offering an efficient and secure way to collaboratively train machine learning models.

The learning process involves a decentralized network of cooperating devices, termed "clients," led by a central server. Each client retains a local training dataset, avoiding the need to transmit it to the server. Rather than transmitting extensive data, clients calculate updates for the central server-held model. This approach aligns with the principle of data minimization, focusing on collecting only the necessary targeted information. [34]. As these updates exclusively enhance the existing model, there's no requirement to retain them post-application.

A principal advantage of this approach is the decoupling of model training from the need for direct access to the raw training data. Clearly, some trust of the server coordinating the training is still required. However, for applications where the training objective can be specified on the basis of data available on each client, federated learning can significantly reduce privacy and security risks by limiting the attack surface to only the device, rather than the device and the cloud.

In this work, our emphasis is on the non-IID and unbalanced properties of the optimization, as well as the critical nature of the communication constraints.

2.4 Neural Networks

A neural network is a computational model inspired by the structure and functioning of the human brain, consisting of interconnected nodes (neurons) organized into layers. It is a fundamental component of modern machine learning and can automatically learn from data to solve a wide range of tasks, such as image recognition, natural language processing, and more. Neural networks excel at capturing complex patterns and representations from data by adjusting the connections between neurons (weights) during the training process.[38]

The main idea of such networks is (to some extent) inspired by the way the biological neural system works, to process data, and information in order to learn and create knowledge. The key element of this idea is to create new structures for the information processing system. The Artificial neural network architecture is shown in the figure 2.

The system is made up of a large number of highly interconnected processing elements called neurons that work together to solve a problem and transmit

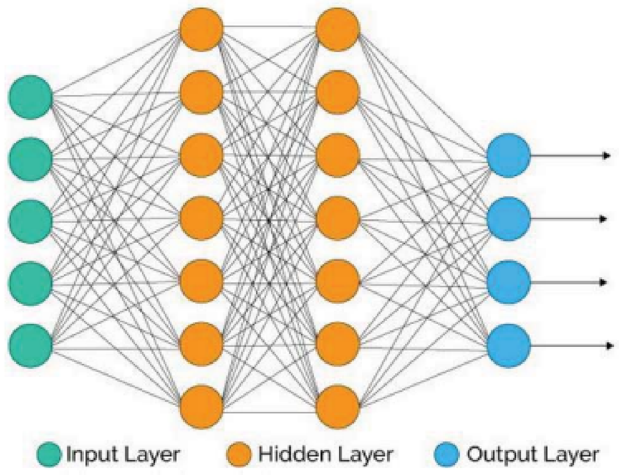


Figure 2.2: Artificial neural network architecture [7]

information through synapses (electromagnetic connections). The neurons are interconnected closely and organized into layers. The input layer receives the data, while the output layer generates the final result. Between the two, one or more secret layers are typically sandwiched. This arrangement makes predicting or knowing the exact flow of data difficult. Each connection has a connection weight, and each neuron has a threshold value and an activation function [2].

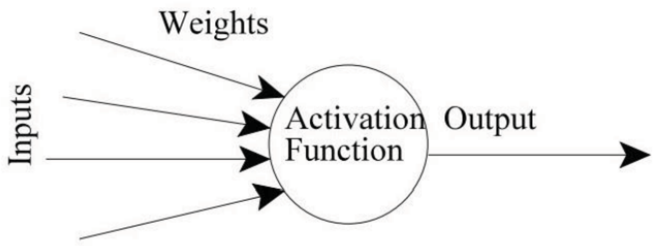


Figure 2.3: Weight, input and output of the ANN. [27]

It is calculated if each input has a positive or negative weight based on the sign of the input's weight. The weight affects the signal intensity at a connection [27]. Neurons which have a threshold above which a signal is only transmitted if the aggregate signal exceeds it. The Activation Value is the weighted sum of the summing unit, and the output is generated based on the signal from this activation value. The learning rate determines how large the model's corrective steps are in adjusting for errors in each observation. A high learning rate reduces training time but reduces overall accuracy, while a low learning rate takes longer but has the potential for greater accuracy.

3 Related Work

In this chapter, we will analyze some works which are related to the current thesis. It's important to mention that not many studies have looked deeply into our topic, which is about how to share resources and encourage devices to give rewards using federated learning. However, there exist research efforts provide a foundation which we can use in our thesis.we decided to partition it in four main topic: distributed learning, Federated learning and its strategies, bidding framework and resource allocation .

3.1 Distributed Learning Incentive Scheme

Distributed learning is a collaborative approach in machine learning where the process of training models is distributed among multiple devices, servers, or nodes. Instead of centralizing data and computations in a single location, each participant holds its own data and performs local model training. In reality each node consider as selfish and incur costs during the computation [11], and each node has its own private information[12] [28].

In distributed learning, the concept of providing incentives is incorporated. This approach involves multiple computation rounds, where the platform receives a decodable subset of results in each round. Upon receiving these results, the platform notifies other workers to cease computation, thus preventing any wastage [1]. Following the computation of each round, the platform rewards the participating workers.

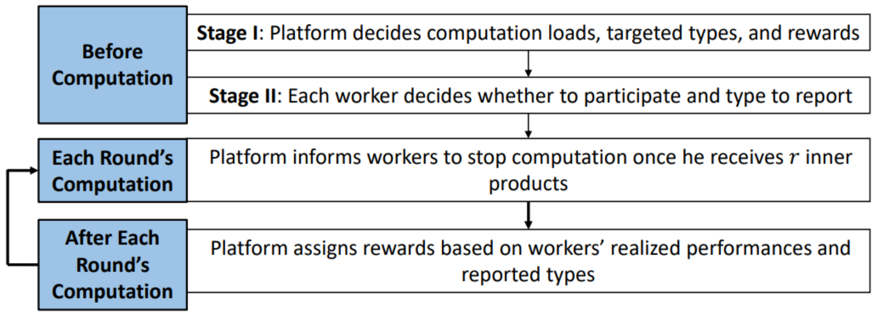

Before Computation
Stage I: Platform decides computation loads, targeted types, and rewards
Stage II: Each worker decides whether to participate and type to report
Each Round's Computation
Platform informs workers to stop computation once he receives r inner products
After Each Round's Computation
Platform assigns rewards based on workers' realized performances and reported types

Figure 3.1: Workflow of coded machine learning with incentives [13].

In paper [13], they introduced incentive mechanisms designed for devices operating in scenarios with both complete and incomplete information. These mechanisms address the multi-dimensional heterogeneity of workers in terms of computation performances and costs. We will focus on the scenario of incomplete information, where the nodes lack knowledge about each other's computation costs. This scenario relatable to our resource sharing problem.

The figure 3.1 describe in brief the steps used to reward incentives in distributed learning environment. In *Stage I*, the platform communicates the specific types of workers targeted, denoted as $S \subseteq M$, the computation loads for each computation round, and the rewards allocated for completing a computation task in each round. In essence, the platform establishes the anticipated rewards for workers. Moving on to *Stage II*, once informed about the platform's choices, each worker determines whether to take part and which type to declare. This decision is influenced by the fact that rewards are based on the actual performances achieved by workers and the types they declare subsequent to the computation process.

Workers Payoff: Because of the random nature of computation time, each worker's goal is to maximize their expected payoff. This is calculated by subtracting their anticipated computation cost from the expected reward. Within each round, the expected time taken by all workers is represented by $\mathbb{E}$, which stands for the overall expected runtime. Despite a worker completing their task early, they're still required to wait for others to finish within the same round, preventing them from switching tasks. Consequently, for a worker of type m, the anticipated computation cost is $c_m\mathbb{E}[T]$. As a result, the expected earnings for a type-m worker in each round are as follows:

$$\mathbb{E}[U(c_m, \mu_m, a_m, \bar{m})] = p_{\bar{m}} c_m \mathbb{E}[T] \tag{3.1}$$

Workers Participation Decisions in Stage II: Based on the load assignment and rewards, a worker decides to participate, he can only choose to report a type that is within the targeted worker type set announced by the platform. if a type-m worker participates and reports his type as $\bar{m}$, his expected payoff in this case is.

$$\mathbb{E}[U(c_m, \mu_m, a_m, \bar{m})] = p_{\bar{m}} c_m \frac{r}{\sum_{m \in S} N_m \frac{\mu_m}{1+\mu_m \lambda_m}} \tag{3.2}$$

Each worker will participate in the computation once he expects a non-negative payoff in equation . $E[U(c_m, \mu_m, a_m, \bar{m})] \geq 0$.

Platforms Strategies in Stage I: Taking workers decisions into consideration, the platform determines the optimal targeted worker type set and the optimal rewards for workers. Since the platform does not know each workers type, the platform needs to ensure a non-negative payoff for each targeted worker type and make sure that all workers do not misreport their type.

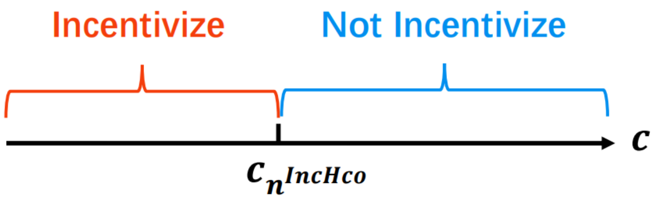


Figure 3.2: Incomplete-Information scenario.

Figure 3.2,demonstrates that the platform gives all workers the same expected reward, which ensures that worker types in the desirable set S^{IncHco} will participate, and types not in S^{IncHco} will not participate

Experimental evidence confirms that, in cases of incomplete information, the platform assigns higher rewards to workers demonstrating efficient computation capabilities. These reward-performance patterns remain consistent regardless of the distribution of workers within each type. Notably, the platform's additional costs stemming from incomplete information diminish as the number of workers increases significantly. However, this reduction in costs doesn't exhibit a steady decline as the worker count increases; rather, it follows a more complex trend..

3.1.1 Limitation of Distributed Learning

Distributed learning encounters challenges like communication overhead, privacy concerns from raw data sharing, lack of centralized control leading to inconsistent updates and model performance, and limitations in heterogeneous environments and scalability [40]. In contrast, federated learning can handle data that is non-identically distributed across devices, making it more adaptable to real-world scenarios. It leverages localized data to enhance privacy and employs a decentralized approach for improved scalability and adaptability, addressing these drawbacks and offering a more efficient and privacy-conscious method for collaborative model training across devices. Certainly, we will provide a comprehensive analysis of these aspects in the upcoming section.

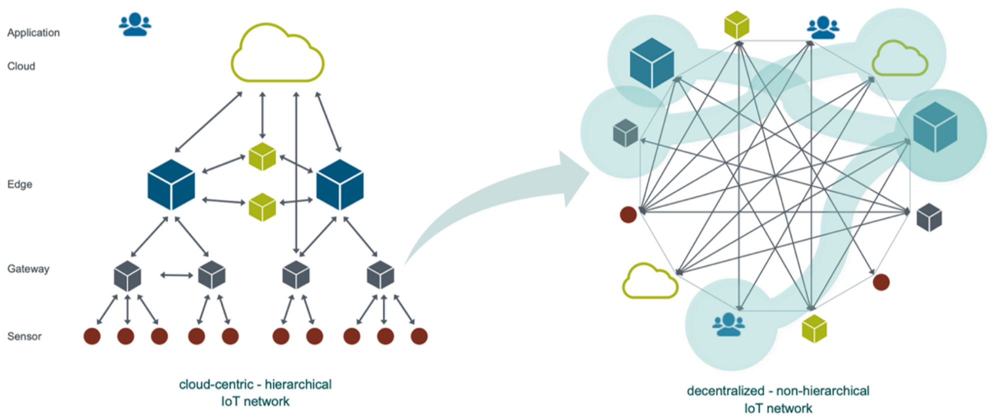

Application
Cloud
Edge
Gateway
Sensor
cloud-centric - hierarchical
IoT network
decentralized - non-hierarchical
IoT network

Figure 3.3: Difference Between Centralize and Decentralize

3.2 Federated Learning (FL)

Federated learning is a cutting-edge technique, that recently introduced in ML, enabling collaborative model training across decentralized devices, it was firstly proposed by McMahan in [29]. It is designed for privacy-concerning scenarios where local nodes would not like to upload and share their private data. [25]. This technique involves adjusting the model's architecture and loss function to enhance compatibility with gradient-based optimization methods [15]. While computationally efficient, it demands numerous rounds of single batch gradient calculations for effective model training [21]. They adopt a strategy where a fraction of clients is chosen during each round, and their combined data is used to compute the loss gradient. Here, the value of fraction of client determines the global batch size, with $C = 1$ representing full-batch gradient descent. This fundamental algorithm is referred to as FedSGD.

In the upcoming discussion, we will explore the strategies of FedAvg and FedProx, which are pivotal in optimizing the collaborative training process and addressing challenges related to communication, privacy, and convergence in federated learning setups.

3.2.1 Federated Averaging (FedAvg)

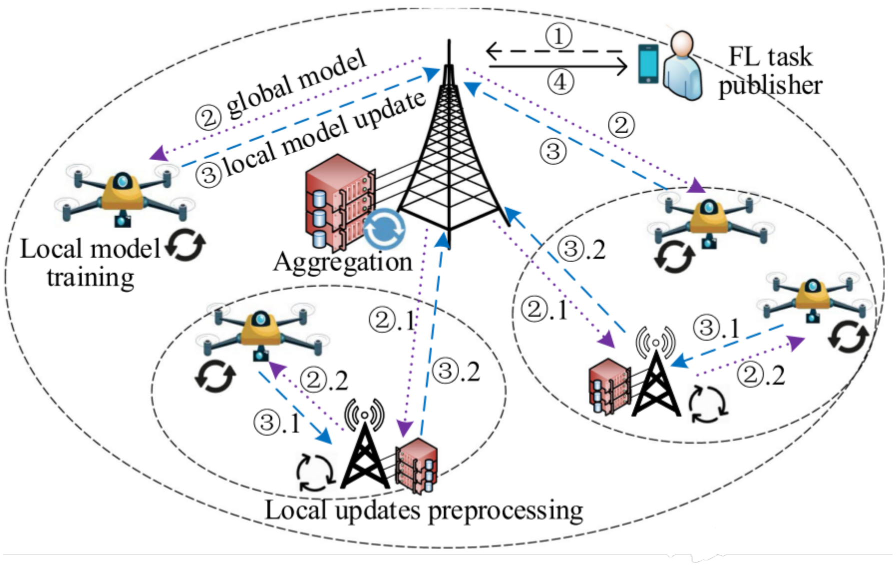


Figure 3.4: Illustration of federated learning services for UAV-aided crowdsensing in 5G HetNets.

Federated Averaging is a central algorithm in FL,a decentralized approach for training machine learning models across multiple devices or nodes. FedAvg is baseline strategy and there also other version are there, It begins with initializing global model that is shared among all devices. In each training round, a subset of devices is chosen, and these devices independently update the global model using their local data through local model training. The updated models are then aggregated by averaging their parameters on a central server, creating a new global model. This iterative process of local updates, aggregation, and sharing the global model continues over multiple rounds. FedAvg addresses challenges posed by heterogeneous data distributions and limited communication capabilities in federated environments, striking a balance between model performance and privacy preservation by leveraging collaborative knowledge without sharing sensitive data. Complete pseudo-code is given in Algorithm 1.

Algorithm 1 *FederatedAveraging*. The K clients are indexed by $k; B$ is the local minibatch size, E is the number of local epochs, and η is the learning rate [16].

```
Server Executes:
initialize $w_0$
for each round t = 1 , 2 . . . do
    $m \leftarrow \max(K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
        $w^k_{t+1} \leftarrow$ ClientUpdate $(k, w_t)$
        $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w^k_{t+1}$
    end for
end for
ClientUpdate $(k, w)$ {Run on client side $k$}
$\beta \leftarrow$ (split $P_k$ into batches of size $B$)
for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \beta$ do
        $w \leftarrow w - \eta \nabla l(w; b)$
    end for
end for
return $w$ to server
```

It is the extended version of FedSGD with $C = 1$ and a fixed learning rate η, has clients $(k \in K)$, computes the average gradient on its local data at the current model w_t, then central server aggregates these gradients and applies the update $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} n_k \frac{n_k}{n} g_k$. An equivalent update is given by $\forall K, w^k_{t+1} \leftarrow w_t - \eta g_k$ then $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w^k_{t+1}$. Each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. This can be extended to more clients and add more computation to each client by iterating the local update $w^k \leftarrow w^k - \eta \nabla F_k(w^K)$, multiple times before the averaging.

Computation is governed by three parameters: C (fraction of clients performing computation), E (number of training passes per client), and B (local minibatch size). The impact of the number of local epochs on convergence is significant. In scenarios with heterogeneous local objectives, increasing local epochs can direct each device towards its local rather than global objective, possibly leading to convergence issues. It is essential to select an appropriate local epoch count that balances communication reduction and robust convergence, accounting for varying factors like local data and resource availability.

Advance federated learning uses various strategies to avoid divergence like distillation. A more natural approach than mandating a fixed number of local epochs is to allow the epochs to vary according to the characteristics of the network, and to carefully merge solutions by accounting for this heterogeneity and introduced below.

3.2.2 Federated Proximal (FedProx) strategies

FedProx deals the situations where the distribution of data across clients is non-uniform or have different amounts of data, this can lead to slower convergence or even divergence of the global model. FedProx solved this by adding a regularization term to the loss function during model updates. This regularization term encourages the local models to stay close to a central model, promoting convergence and improving generalization across clients.

The regularization term is based on the proximal operator, which is a mathematical concept used in optimization [26]. Instead of assuming a uniform γ for all devices throughout the training process, FedProx implicitly accommodates variable γs for different devices and at different iterations. In particular, instead of just minimizing the local function $F_k(\cdot)$, device k uses its local solver of choice to approximately minimize the following objective h_k

$$\min_{w} h_k(w; wt) = F_k(w) + \frac{\mu}{2}||w - w^t||^2 \tag{3.3}$$

The proximal term is beneficial in two aspects: (1) It addresses the issue of statistical heterogeneity by restricting the local updates to be closer to the initial (global) model without any need to manually set the number of local epochs. (2) It allows for safely incorporating variable amounts of local work resulting from systems heterogeneity. Summarize steps of *FedProx* algorithm can be seen in Algorithm 2.

An important distinction of the FedProx usage is that of tackling heterogeneity in federated networks and can help in solving objectives in a distributed setting with: (1) non-IID partitioned data, (2) the use of any local solver, (3) variable inexact updates across devices, and (4) a subset of devices being active at each round. These assumptions are critical to providing a characterization of such a framework in realistic federated scenarios.

Algorithm 2 *Federated Proximal.*

```
Require: Input $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
  for t = 0 ,1 , . . . T-1 do
    Server Selects a subset $S_t$ of $K$ devices at random (each device $k$ is chosen with probability $p_k$)
    Server sends $w^t$ to all chosen devices
    Each chosen device $k \in S_t$ finds a $w_k^{t+1}$ which is a $\gamma_k^t$ inexact minimizer of :
    $w_k^{t+1} \approx \arg\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}||w - w^t||^2$
    Each device $k \in S_t$ sends $w_k^{t+1}$ back to server.
    Server aggregates the $w's$ as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
  end for
```

3.3 Bidding Approach for Task-Offloading with Auction Mechanism

Bidding is a key topic in our thesis, and we'll now discuss recent research that focuses on this area. Specifically, these studies are all about creating a system where participants bid for tasks to be done in Mobile Edge Computing (MEC) for task offloading. We'll look into these studies to see how their ideas can help us with our resource sharing problem

Resource Allocation for Task-Offloading with Auction Mechanism involves a technique in MEC settings for transferring computational tasks from devices to edge servers. Tasks can also be offloaded to nearby idle users using technologies like Device-to-Device communication [14], also known as collaborative task offloading [10].

In [36], an auction mechanism is utilized to efficiently allocate edge server computational resources to offloaded tasks. Collaborative execution of computing tasks can occur on local devices, neighboring devices, and remote MEC servers. However, task execution consumes device energy, and limited battery capacity may discourage idle users from performing others' tasks. Hence, incentivizing idle users to engage in collaborative computing becomes crucial. They proposed a computing resource sharing auction (CRSA) algorithm to motivate idle users to participate in task offloading.

It start by collecting the bids from idle user or node from base station. They use ε and G to denote the set and the number of the winning idle users in the auction, respectively. Then, they choose the bids with the minimum price density $l_{e,n}/f_{e,n}$ as the winning bids. The total amount of computing resource provided by the winning bids is at least equal to the amount of computing resource that the BS intends to collect. The reward obtained by the winning idle user e is the product of the price density of the idle user who will win when the winning idle user e does not participate in the auction and the amount of computing resource provided by the winning idle user ε.

The detailed description of the CRSA algorithm is shown in Algorithm 1.

Algorithm 3 CRSA Alogrithm

```
for $e \in \mathcal{E}$ do
    Collect bids $L_e \triangleq (l_{e,1}, f_{e,1}), \dots, (l_{e,N}, f_{e,N})$ from idle user $e$.
end for
Set $Q = \max_{e \in \mathcal{E}, n \in N} l_{e,N}/f_{e,N}$
Further set $C = 0, G = 0, \mathcal{E}^{'} := 0$.
while $U > C$ do
    $e^*, n^* = \arg\min_{e \in \mathcal{E}, n \in N} l_{e,n}/f_{e,n}$
    $x^{'}_{e^*,n^*} = 1$ and $x^{'}_{e^*,n^*} = 0, \forall \in N \backslash n^*$
    $C = C + f_{e^*,n^*}.G = G + 1, \mathcal{E}^{'} := \mathcal{E}^{'} \cup e^*$.
    Set $l_{e^*,n}/f_{e^*,n} = Q$ for all $n \in N$
end while
for do
    Compute the reward $r_{e^*}$
end for
```

Idle users can enhance their odds of winning by lowering the price. The victorious idle user (e) sets the highest price for the corresponding computing resource, which is established by identifying the idle user who would secure victory if idle user e did not partake in the auction. The utmost value of the price density for the winning idle user e is no greater than the price density of the idle user who would triumph if idle user e abstained from the auction. Since the reward for the winning idle user e is determined as the critical price, the algorithm adheres to truthfulness [8].

The algorithm gathers bids from E idle users and selected winning bids determine rewards for idle users using a reward scheme that calculates the price density of the idle user who would win without the winning idle user e. The process requires a maximum of two interactions per idle user. Once the base station identifies winning bids and provides rewards to those idle users, busy users can offload tasks to the MEC server or the winning idle users, or complete tasks locally. Using the set $M \triangleq 0, \mathcal{E}^{'}$, task executors are denoted by the MEC server (0) and G idle users. The allocated computing resources by winning idle users to busy users, match the declarations in the winning bids.

3.4 FL Based Resource Allocation by Reward Distribution

Following the bidding approach, our thesis also addresses the allocation of resources to servers by computing rewards or incentives. This section represents latest work done by behmand [3], on a policy training approach for resource allocation (RA) that offloads the policy training task from the server.

This approach aiming to address the key challenge related to distributed reward computation. Unlike conventional Deep Learning based Resource Allocation methods where a centralized reward function is readily available at the server, FL approach seeks to achieve decentralized reward computation. This decentralization ensures that each user can independently calculate its own reward without requiring access to the reward functions or policy parameters of other users, denoted as θj with $j \neq i$. The pseudo code of the algorithm can be observed in the provided algorithm 4

Reward Distribution: In this approach, the issue of *decentralized reward* computation in Federated Learning (FL) was effectively addressed by employing a gradient-free approach. Instead of directly communicating gradients, users independently estimated gradients through their local function evaluations and parameter adjustments. This novel strategy eliminates the need for extensive inter-user communication. The estimated gradients are then used to sequentially update parameters.

To enhance these updates without requiring direct communication between users, a distributed reward term $(\bar{y}_i + \tilde{y}_i)$ is incorporated. This clever integration of the distributed reward term contributes to the enhancement of parameter updates while minimizing the need for inter-user communication. Consequently, this approach ensures the efficiency of FL by significantly reducing communication requirements and associated delays.

When user j updates θ_j^k, user i tracks its local reward change $f_\iota(\theta_\iota^k, \theta_{\setminus\iota}^k)$. This change updates $i's$ Whereas the server aggregates the parametes every I

Algorithm 4 *FL* based Resource Allocation [3].

```
**Initialize** $\theta_\iota^0 = \theta^0, \forall \iota \in [N], \alpha > 0, \beta > 0$ for batchsize $B \geq 1$
*Output:* $\theta^K$
**for** k = 1 , 2 . . . , K **do**
    $u^l \sim N(0, \sum)\forall l \in [L]$ in all users
    $l_i = k \forall \iota \in [N]$
    **for** i = 1, 2, . . . , N sequentially **do**
        user i
        estimates: $g_\iota^\iota(\theta_\iota^k, \theta_{\setminus\iota}^k)$
        performs: $\theta_\iota^{k+1} = \theta_\iota^k + \gamma^k g_\iota^\iota(\theta_\iota^k, \theta_{\setminus\iota}^k)$
        $l_i = K + 1$
        user $j, \forall j \in [N] \setminus \iota$
        **in parallel and synchronous with user** $\iota$
        estimates: $g_j^\iota(\theta_j^{l_j}, \theta_{\setminus j}^{l_j})$
        performs: $\theta_j^{l_j} \leftarrow \theta_j^{l_j} + \gamma^{l_j} g_j^\iota(\theta_j^{l_j}, \theta_{\setminus j}^{l_j})$
        **if** $(kN + \iota)$ is a multiple of $I$ **then**
            all users send $\theta_l^{I_l} \quad \forall l \in [N]$ to the server
            all users update $\theta_l^{I_l} \leftarrow \bar{\theta} \quad \forall l \in [N]$
        **end if**
    **end for**
**end for**
```

iterations by.

$$\bar{\theta} = \frac{1}{N} \sum_{l=1}^{N} \theta_l^{I_l} \tag{3.4}$$

The training is done in a distributed manner, where the users communicate with a computationally simple server once in a while. More importantly, each user needs to probe only its own data rate, as the distributed reward function, to update its local DNN. Furthermore, the users do not need to share their local measurements with the server during the training. All these features offload the training computational load from the server and guarantee the possibility of real-time modelfree policy (re)training in time-varying environments without necessarily requiring a computationally complex server.

4 Federated Bidding Approach

In this chapter, we will explore how we have put into action the methods outlined in Chapter 2 and Chapter 3. The aggregator starts by asking connected clients or nodes to bid on a task. Clients calculate the value of their resources and the reward they expect, sending this information back to the aggregator. The aggregator then selects the top two nodes, figures out how much to pay each client, and begins the first round of training, combining the loss values. Each round follows a similar process, with different clients winning bids and undergoing training. This ensures that different clients get a chance to participate in each training round.

4.1 Considered Problem

Companies are increasingly collaborating in networks, sharing resources and overseeing their supply chains across organizational borders. Sharing downtime among companies within the sharing economy creates logistical hurdles, especially when it comes to distributing small amounts of physical goods. Despite these challenges, there is a clear demand for such solutions. For instance, at the Audi plant in Ingolstadt Germany,are exploring ways to utilize the largely unused airspace to address the high utilization of intralogistics delivery traffic. Audi finds multicopters highly beneficial for "emergency ordering," cutting delivery times from 15 to 5 minutes compared to AGVs. This is particularly advantageous in production environments with costly downtimes, offering potential cost savings [20].

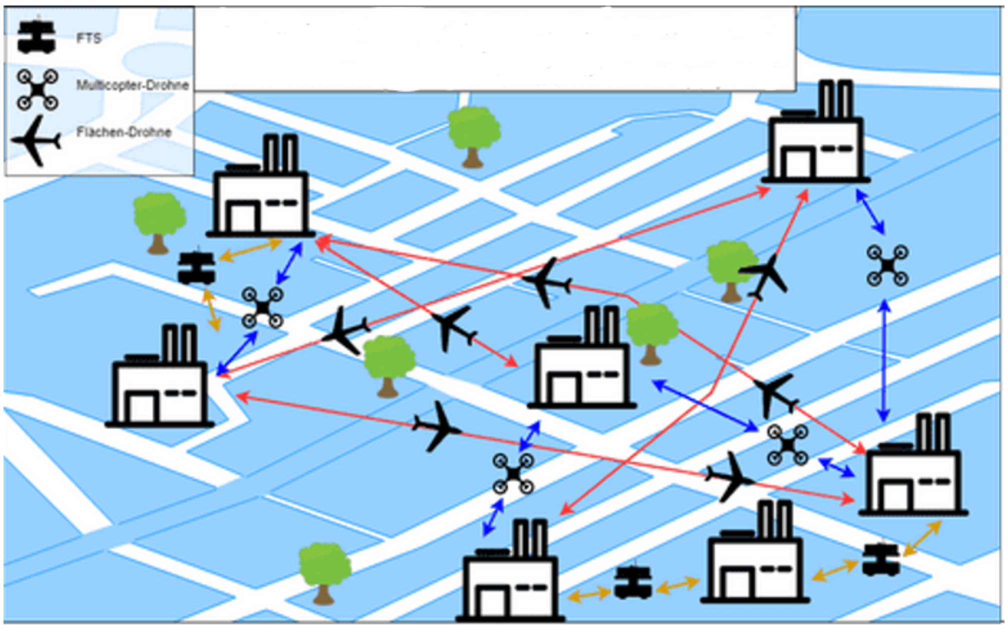


Figure 4.1: DTS Representation

There is therefore a need to combine the respective advantages of DTS, multicopter and area drone technologies in the best possible way via an intelligent overall solution for companies and logistics centers.

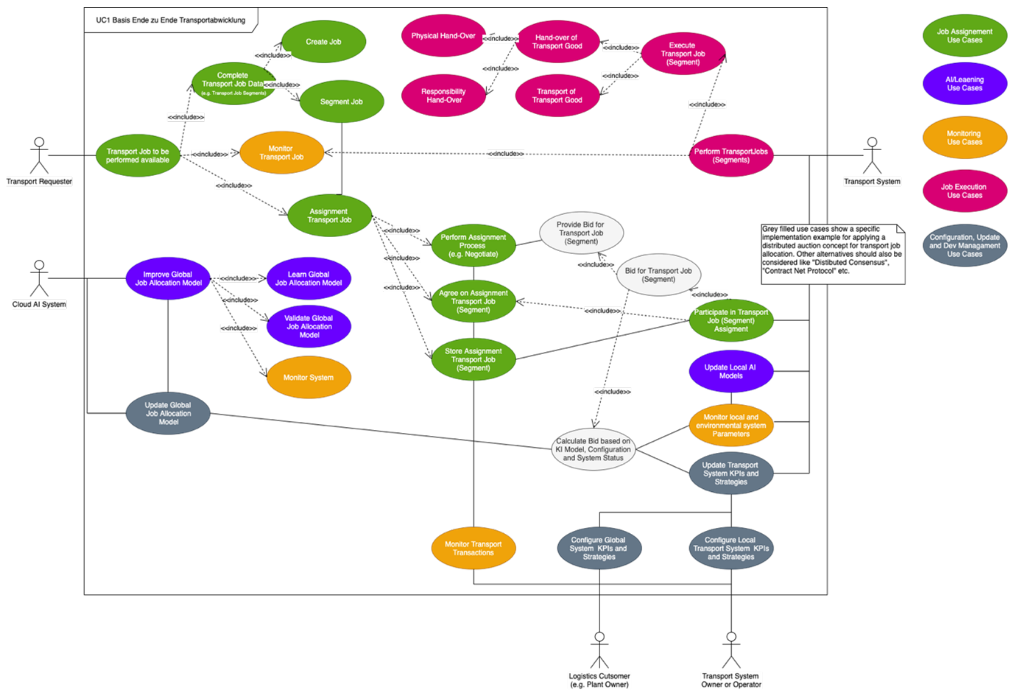
At THWS, research on inspection drones with AI support is being carried on in self-organizing AI models[37, 44]. They are developing a secure, decentralized, self-organized AI system that implements an automatic, adaptive bidding process in the logistics system. The learning of the bidding model [22] is realized by means of federated learning [4] from heterogeneous streaming data [37, 44]. This research project is name as FlowPro, whereas the thesis in only focused on bidding part.

4.1.1 FlowPro Project

The FlowPro project is creating an AI-driven logistics network that self-organizes and enhances intralogistics in industrial areas. It also aims to facilitate the efficient movement of goods between companies using both land-based and aerial means. The FlowPro system brings together job-specific and external data sources, offers optimization for all participants using artificial intelligence, and communicates with them via a platform-independent, decentralized framework. Three autonomous urban-capable systems are connected. The collaborative cooperation of different transport units such as driverless transport systems

(DTS), multicopter and area drones is developed based on an AI service, which enables a heterogeneous network to overcome the individual, technological system boundaries and purposeful use. The entire schema of the project can be seen in the figure 4.2.

The requirements for the project FlowPro system do not allow classical machine learning approaches, because the autonomously and decentrally acting transport units are trained by a machine learning model executed on the device level, which they then share with other network participants in order to improve the overall model. To be able to technically map these requirements, the "federated learning" approach is relied upon [4], as it is conceptually designed to leverage large amounts of decentrally organized data.


UC1 Basis Ende zu Ende Transportabwicklung
Create Job
Complete Transport Job Data (e.g. Transport Job Segments)
Segment Job
Physical Hand-Over
Hand-over of Transport Good
Execute Transport Job (Segment)
Responsibility Hand-Over
Transport of Transport Good
Transport Requester
Transport Job to be performed available
Monitor Transport Job
Perform TransportJobs (Segments)
Transport System
Assignment Transport Job
Perform Assignment Process (e.g. Negotiate)
Provide Bid for Transport Job (Segment)
Bid for Transport Job (Segment)
Cloud AI System
Improve Global Job Allocation Model
Learn Global Job Allocation Model
Validate Global Job Allocation Model
Monitor System
Agree on Assignment Transport Job (Segment)
Participate in Transport Job (Segment) Assigment
Store Assignment Transport Job (Segment)
Update Local AI Models
Update Global Job Allocation Model
Calculate Bid based on KI Model, Configuration and System Status
Monitor local and environmental system Parameters
Update Transport System KPIs and Strategies
Monitor Transport Transactions
Configure Global System KPIs and Strategies
Configure Local Transport System KPIs and Strategies
Logistics Cutsomer (e.g. Plant Owner)
Transport System Owner or Operator
<<include>>
Job Assignement Use Cases
AI/Leaening Use Cases
Monitoring Use Cases
Job Execution Use Cases
Configuration, Update and Dev Managament Use Cases

Grey filled use cases show a specific implementation example for applying a distributed auction concept for transport job allocation. Other alternatives should also be considered like "Distibuted Consensus", "Contract Net Protocol" etc.

Figure 4.2: Proposed working Scheme.

This project will allow logistics units to communicate with each other and determine the most suitable candidate for the transportation job at hand through an AI-powered bidding system. The AI service will be structured in such a way that all parameters in the logistics network, be it traffic data, environmental influences such as the weather as well as the capabilities of the individual transport units and the product data as well as the strategies of each company located in the network, can be included and thus local and global logistics goals (costs, emission reduction, delivery times, capacity utilization increase,) can be achieved. It will also allow to integrate additional transport units at any time, to provide own units for a fee and thus to optimize logistical goals for all partners involved. In FlowPro, the focus is primarily on AI-supported decision-making based on a decentralized agent system.

So as part of FLowPro, this thesis aims to develop a bidding strategies that helps to fully automate with the help of AI, the logistic system to operate with minimal cost. Therefore, Based on this we figure out with help of Federated learning and Incentive schemes we can achieve this goal, As we have already explain the concept and mathematical formulation foundation in previous sections, in the next section we will explain our proposed methodology as the solution for bidding strategies.

4.2 Proposed Method

To accomplish this objective, we draw inspiration from and build upon the multi-dimensional procurement auction model introduced by Che in [9] and further extended in [43]. We introduce the incentive mechanism for resource allocation, which is founded on the principles of multi-dimensional procurement auctions. In this section, we will elucidate the reasoning behind each step of the design. To explicitly illustrate our results, we have made a prototype that shows our approach works better than the previous methods and also fits on problem where drones are required for transport logistics:

Beginning with the aggregator: It kicks off the process by sending out bid requests along with the criteria for selection. Subsequently, participants individually submit sealed bids, including information about the quality of their resources and the payment they expect. After this, the aggregator chooses a certain number of winners, represented as K (where $K \geq 1$), using a ranking of the scores. We address two notably intricate tasks while devising the incentive scheme. First, we devise a unique Nash equilibrium strategy for each node, aimed at maximizing the anticipated profit. Additionally, we offer guidance to the aggregator on acquiring the expected resources, both of which pose significant challenges.To demonstrate the effectiveness of our approach, we've created a sophisticated simulator. We verify the performance of our proposal through extensive testing using different datasets and a range of learning models. Furthermore, we've gone beyond by putting our ideas into practice with a real system that involves multiple nodes. Our approach delivers three key contributions.

a. We introduce a multi-dimensional incentive plan designed for federated learning. It covers different ways of scoring and shows efficiency in specific situations. We use ideas from game theory to find the best strategies for edge participants. Moreover, we apply expected utility theory to help the aggregator get the resources they need effectively.

b. The proposed scheme is not only lightweight but also Incentive Compatible (IC). In real-world usage, the computational load and communication costs are very low. The IC principle ensures that there's no advantage for edge nodes to provide incorrect resource quality information in our proposed framework.

c. Extensive simulation results clearly show that the approach effectively speeds up federated training by decreasing the necessary number of training rounds.

4.3 FL Loss Function

Federated learning is designed to work together in a distributed manner to train a single global model. The goal is to decrease the combined global loss function $F(w)$ [29]. Ultimately, the aim of federated learning is to find the model parameters w^* that satisfy:

$$w^* = \arg\min F(w). \tag{4.1}$$

Usually, the training process requires several rounds to reach convergence. In each round, the aggregator selects K nodes randomly from the total of N edge nodes. Subsequently, the aggregator shares the global parameter $w(t)$, where $t = 0, 1, \ldots, T-1$ indicates the iteration number, with the chosen nodes. Using the global parameter $w(t)$, the selected node proceeds to train the shared model using its local data.

$$w_i(t+1) = w(t) \quad - \quad \eta \nabla F_i(w_i(t)), \tag{4.2}$$

The parameter η represents the step size. Following the local training, these nodes upload their model parameters to the aggregator. Subsequently, the aggregator computes the global parameters for iteration $t+1$ as follows:

$$w(t+1) = \frac{\sum_{i=1}^{N} D_i w_i(t+1)}{\sum_{i=1}^{N} D_i} \tag{4.3}$$

Here, D_i represents the dataset size of node i. Then, the aggregator will initialize the next round of training by randomly choosing K nodes. When the accuracy of global model satisfies the requirement or the training time exceeds the predefined threshold, this training process terminates. Briefly, federated learning consists of many iterations of global aggregation and local training that can be seen on the left part of the Fig. 4.3. Furthermore, both model accuracy and the number of training rounds hold significant importance as performance metrics. Lastly, it's noteworthy to mention that our suggested approach can be employed not only within the framework of the traditional federated learning [29], but also within other paradigms like those discussed in [24].

4.4 Proposed Mechanism Stages

The proposed incentive strategy comprises six sequential stages can be seen on the right side of figure 4.3: bid ask, bid collection, winner determination, task assignment, local training, and global aggregation. These phases repeat in each training round. The final three phases bear resemblance to the conventional federated learning approach, RandFL, discussed in [29]. The computational and communication demands arise primarily in the initial three phases. Let's delve into a comprehensive description of each stage. The proposed alogorithm can be seen in the alogrithm 5.

4.4.1 Bid Ask:

In every round of federated learning, the process begins with the aggregator broadcasting a scoring rule $S(q_1, \ldots, q_m, \quad p)$, where $q = (q_1, \ldots, q_m)$ is the quality vector of resources, and p is the expected payment that the edge node bids with the provision of q. In our approach for simulating drones, the considered resources include battery capacity and the weight capacity of the device. Furthermore, the aggregator uses the scoring function to select participants. We

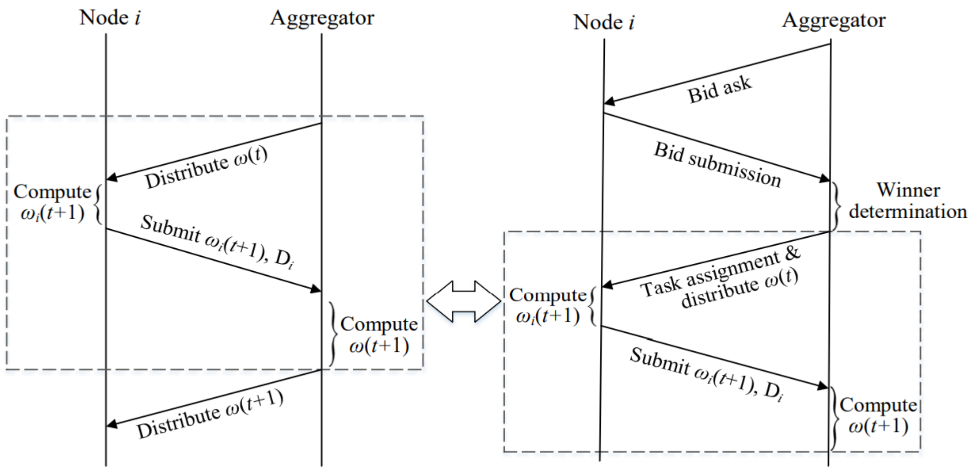


Figure 4.3: Difference Between RandFL and our approach

express $S(q_i, p_i)$ as a quasi-linear function.

$$S(q_{i1}, q_{i2}, \dots, q_{im}, p_i) \quad = \quad s(q_{i1}, q_{i2}, \dots, q_{im})p_i \tag{4.4}$$

Here, the subscript $i = 1, 2, \dots, N$ represents the node index. When compared to the size of the model parameters, the communication overhead in this step is disregarded. This is because only a score function and basic criteria are transmitted from the aggregator to the edge nodes, resulting in a data size of just a few bytes.Some classic utility functions include the perfect substitution utility function, the perfect complementary function, and the general Cobb-Douglas function, which are separately denoted as:

$$s(\cdot) = \alpha_1 q_1 + \cdots + \alpha_m q_m, \tag{4.5}$$

$$s(\cdot) = \min\{\alpha_1 q_1, \dots, \alpha_m q_m\} \tag{4.6}$$

where $\alpha_1, \dots, \alpha_m$ are coefficients. Many scoring functions can also be included. For our approach, $s(\cdot)$,we set as the utility function $U(q_1, \dots, q_m)$ of the aggregator. We added the constraint $\alpha_i = 1$, but it is not imperative. From above mentioned functions, The additive form (eq 4.5) is favored over perfect substitution resources, much like how distinct drone features, such as battery life and payload capacity, hold unique importance instead of being interchangeable. Meanwhile, the perfect complementary form (eq 4.5) could be the optimal choice for situations where both flight range and payload capacity are jointly taken into account.

After the Bid ask phase, the aggregator proceeds to collect bids from nodes, a process that will be further explained in the upcoming section.

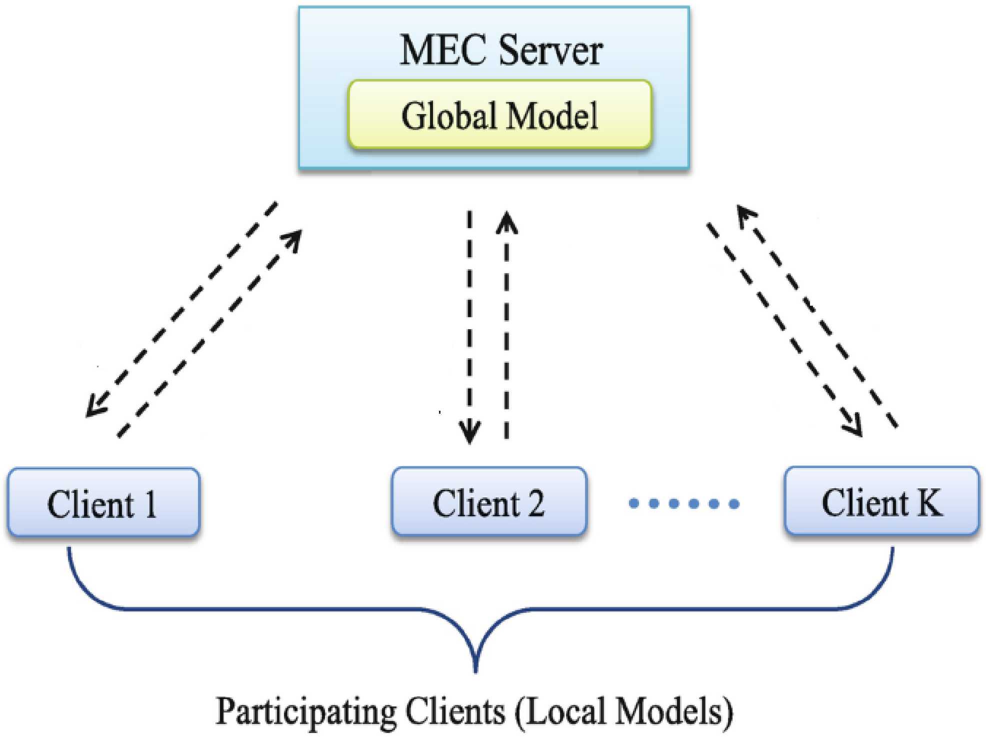


Figure 4.4: Illustration of communication between aggregator and client

4.4.2 Bid Collection:

Upon receiving a bid ask accompanied by the scoring function $S(\cdot)$, edge nodes independently determine whether to place a bid or abstain based on their available resources. As per the private value model outlined in [41], edge node i possesses an individual cost parameter denoted as θ_i, which enables it to establish a private cost function $c(q_1, \ldots, q_m, \theta_i)$. Note that the cost function $c(\cdot)$ is an increasing function of q_i. In our proposed method, we assume single crossing conditions $c_{qq} \geq 0, c_{q\theta} > 0$, and $_{cqq\theta} \geq 0$, which mean the marginal cost increases with the parameter θ. As for now we don't have realistic dataset but after gathering realistic data before bidding, each node should learns its private cost parameter θ and gets the Cumulative Distribution Function (CDF) $F(\theta)$ from the historical data. It is assumed that θ_i is independently and identically distributed over the range of $[\underline{\theta}, \bar{\theta}]$ $(0 < \underline{\theta} < \bar{\theta} < \infty)$. There also exists a positive and continuously differentiable density function $f(\theta)$. Now there is a question for the nodes, how much to bid? As a rational edge node, node i needs to choose q_i and p_i to maximize the following profit function:

$$\pi_i(q_1, \ldots, q_m, p_i) = p_i - c(q_1, \ldots, q_m, \theta_i). \tag{4.7}$$

In this optimization problem, one of the constraints is Individual Rationality (IR), which implies that any node will not participate in federated learning when its profit is negative. In other words, $\pi_i(qi, p_i) \geq 0$. Let $\pi_i(q_i, p_i) = 0$ denote that node i will not join in the training. When edge node i submits its bid (q_i, p_i) to the aggregator, the technique of sealed-bid auction is adopted, indicating that this bid is only known to the aggregator and node i. The sealed-bid auction is quite suitable for network scenarios and can be easily implemented in our proposed method.

Computing Nash

The Nash equilibrium strategy t_i^{ne} for edge node i consists of two components, as in our case, the qualities of resources and the expected payment. it is found that the choice of quality is only relevant to the private cost parameter θ. In other words,we can say that quality should be independently chosen.

As we already mentioned we are using sealed bid (first-price) auction with $K(K \geq 1)$ winners, the quality of resource of each client, available to perform a task at the moment, is chosen at $q_{s(\theta)}$ for all $\in [\underline{\theta}, \bar{\theta}]$. The unique Nash equilibrium strategy $t^{ne}(\theta) = (q_s(\theta), p_s(\theta))$ for each node in the first-price auction with k winner is given as:

$$q_s(\theta) = \arg\max s(q) - c(q, \theta) \tag{4.8}$$

$$p_s(\theta) = c(q_s, \theta) + \int_u^0 \left(\frac{g(x)}{g(u)}\right) dx \tag{4.9}$$

$$u(\theta) = s(q(\theta)) - c(q(\theta), \theta) \tag{4.10}$$

4.4.3 Winner Determination:

When the aggregator collects sufficient bids with a predefined threshold number, the aggregator finishes the bid collection process. Then, it starts to determine the winners. In tour approach, we extend the classic multi-dimensional auction to multiple winners. In the winner determination, the aggregator has to maximize the profit function $V(\cdot)$ as:

$$V = \sum_{i \in W} (U(q_{i1}, q_{i2}, \ldots, q_{im}) \quad - \quad p_i), \tag{4.11}$$

where $\mathbb{W}$ is the winner set, and $U(\cdot)$ is the utility function of $q = (q_1, q_2, \ldots, q_m)$. Similar to the literature [25], we also assume that

$$U^{'}(\cdot) \geq 0, \quad U^{''}(\cdot) < 0, \quad \lim_{q=0} U^{'}(q) = \infty, \tag{4.12}$$

and $\lim_{q=\infty} U^{''}(q) = 0$.

Moreover, the constraint of individual rationality, i.e., $V \geq 0$, should be satisfied for the rational aggregator as well. In proposed method, the aggregator chooses K edge nodes with the best scores to construct the winner set $\mathbb{W}$. The parameter K can be fixed initially and can also be estimated with historical data. Besides the winner determination, the aggregator has to perform the payment allocation. We use the first-price auction for simplicity.

4.4.4 Task Assignment:

After collection of bids and sorting them according to their scores, this step involves determining which subset of nodes should participate in a particular training round. This is based on clients getting high score based on provided resource qualities.we have N edge nodes, each indexed as $i = 1, 2, \ldots, N$. Each node has an associated score S_i representing its performance or suitability for participating in the current training round. These scores are calculated based on various factors such as past performance, available resources, or other relevant metrics. To assign tasks to nodes with higher scores, we sort the nodes in descending order of their scores:

$$S_{i1} \geq S_{i1} \geq, \ldots, S_{iN}$$

Next, we select the top K nodes with the highest scores for participation in the training round:

$$Selected \quad Node : \{i_1, i_2, \ldots, i_k\}$$

Where $i_1, i_2, \ldots, i_K$ are the indices of the selected nodes with the highest scores.This strategy ensures that nodes with superior performance or capabilities are given priority in the training process, potentially leading to faster convergence and better model outcomes.

The last three steps are similar to the classic federated learning RandFL, where winners locally train the model with declared resources, according to Eq 4.2 . After finishing local updates, they submit the result of model parameters to the aggregator and then obtain the corresponding payment. If any edge node does not comply with the contract, it will be put into the blacklist by the aggregator. The pseudocode of FMore is given in Algorithm 5. Compared with RandFL, our scheme FMore just adds one round of information exchange between edge nodes and the aggregator, and the total communication cost is a linear function so our proposed scheme is lightweight, which is much more appropriate for MEC.

4.4.5 Local Training:

Let's denote the local loss function for device i as $L_i(w)$, where w represents the model parameters. The goal of local training is to minimize this local loss. Mathematically, this is done by adjusting the model parameters w using optimization methods like stochastic gradient descent (SGD). In each iteration of local training, the model parameters are updated as follows:

$$w_i^{(t+1)} = w_i^{(t)} - \eta * \nabla L_i(w_i^{(t)}) \tag{4.13}$$

Whereas: t is the iteration step of local training. η is the learning rate, a hyperparameter that determines the step size in the optimization process. $\nabla L_i(w_i^{(t)})$ is the gradient of the local loss with respect to the model parameters at iteration t.

Devices repeat this process for a predefined number of local training iterations. FedAvg aggregates these locally trained models from multiple devices to create a more globally updated model. The process involves averaging the model parameters of the devices, which helps in collaborative learning while preserving data privacy. The detail of the algorithm is already describe in chapter 2 of section 1.

4.4.6 Golbal Aggregation:

Gradient Calculation: After local training, each device computes the gradients of its local model with respect to a global objective function. These gradients represent the direction in which the local model parameters should be adjusted to improve the global objective.

Aggregation: The central server aggregates these gradients by averaging them, which involves summing up the gradients from all devices and dividing by the total number of devices. This aggregated gradient represents the overall direction of improvement for the global model.

Model Update: The central server updates the global model parameters using the aggregated gradient. This updated global model is then sent back to the devices for the next round of local training.

we have N devices, and for device i, the local loss function is denoted by $L_i(w)$, where w represents the model parameters. The global objective function is given by the average of the local losses across all devices:

$$F(w) = \frac{1}{N} * \sum_i L_i(w) \tag{4.14}$$

In each local training step, device i computes the gradient of its local loss function with respect to its model parameters. After aggregating the gradients, the updated global model parameters w' are computed using the averaged gradient:

$$w' = w - \eta * \frac{1}{N} * \sum_i g_i \tag{4.15}$$

Here, η is the learning rate, and the summation is over all devices.The process is iterated over multiple rounds, where in each round, devices perform local training, compute gradients, and the central server aggregates the gradients to update the global model. This way, FedAvg enables collaborative training while keeping data localized and preserving privacy.

Algorithm 5 Proposed Algorithm for Incentive Federated Learning

```
Input: Node Set $\mathbb{N} = 1, 2, \ldots, N$, local data size $D_1, D_2, \ldots, D_N$
Output: global model parameter $w(t)$.
$t = 0$;
Set parameters of all edge nodes $w_i(0) = w(0)$;
for $t = 1, 2, \ldots, T$ do
    the aggregator sends scoring rule $S(\mathbf{q}, p)$
    for node $i \in N$ in parallel do
        Node $i$ computes its $\mathbf{q}$ as Eq. 4.8;
        Node $i$ obtains its $p$ using Euler's mehtod;
        Node $i$ submits bid $(\mathbf{q}_i, p_i)$ to the aggregator;
    end for
    The aggregator computes $S(\mathbf{q}_i, p_i)$ and sorts all the scores;
    $\mathbb{W} \leftarrow K$ nodes with the top $K$ scores;
    for in parallel $i \in \mathbb{W}$ in parallel do
        The aggregator sends payment $p_i$ and global $w(t)$ to node $i$;
        Node $i$ trains $w_i(t+1)$ wit $D_i$ according to Eq 4.2.
        Node $i$ sends $w_i(t+1)$ to the aggregator
    end for
    The aggregator computes $w(t+1)$ according to equation 4.3;
end for
```

4.5 Flower Framework [1]

To simulate our method we have used flower framework, that makes easy to work in reliable environment in decentralized manner. In central machine learning, data is moved to the computation, whereas in federated learning, computation is moved to the data. The Flower Framework offers an organized python programming framework, empowering an aggregator to initiate the process by initializing a global model and distributing it to connected devices. These devices independently train the model using their personal datasets, subsequently sending back their model updates to the aggregator. The aggregator then consolidates these updates to form a global model. This approach allows to repeat the process by choosing the number of rounds and epochs, thereby enhancing convergence.

Let's explore how the various stages of the Flower framework enable us to encourage connected devices to gain more incentives by sharing their private

[1]`https://flower.dev/`

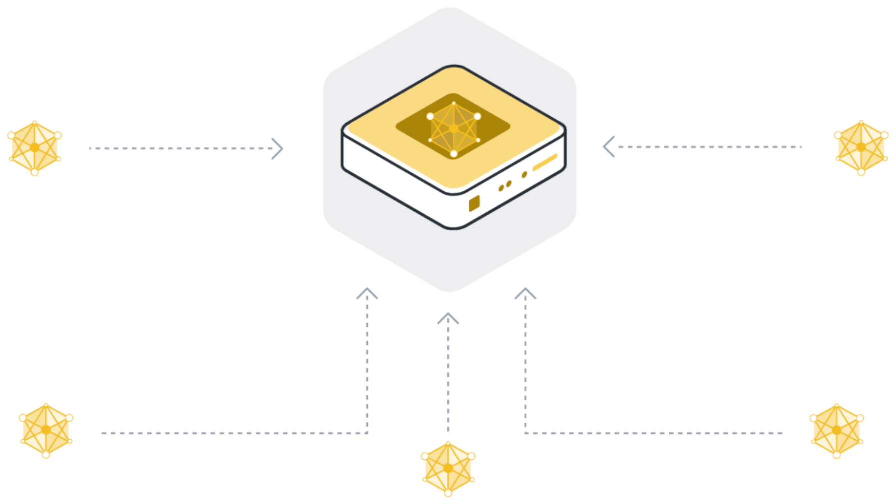


Figure 4.5: Flower Framework Concept

resources, allowing each selected device to train the model on personal data and also we will explore how Flower enables us to evaluate the performance of these trained models.

4.5.1 Initializing Simulation

The flower process starts by setting the model's parameters on the aggregator as per the strategy. Our strategy involves randomly initializing the model's parameters. The following code initiate the simulation process:

```python
import flwr as fl

fl.simulation.start_simulation(
    client_fn=client_fn,
    num_clients=NUM_CLIENTS,
    config=fl.server.ServerConfig(num_rounds=NUM_ROUND),
    strategy=strategy,
    client_resources=client_resources,
)
```

Code 4.1: FLower start simulation

This initialization phase serves as a fundamental step and involves three main parameters: the number of clients, how many clients to be included in the bidding step, the number of rounds and the chosen bidding strategy.

4.5.2 Request for bidding

After knowing the strategy and number of clients, the aggregator starts asking to bid from the client, for the task at hand. Those client who are having enough required resources are only allowed to take part in the bidding process. They calculate their resources according to equation defined in the section and send the bidding resource quality values back to aggregator.

4.5.3 Winner determination

After collection of bidding the aggregator arrange them in descending order, and choose the top scorer. Here we assuming that all the clients they are rational and they only play fairly.

```
 def configure_fit(
     self, server_round: int, parameters: Parameters,
client_manager: ClientManager) -> List[Tuple[ClientProxy, FitIns
]]:
     all_connected_clients_dict = client_manager.all()
     all_client_properties = list()
     for dict_key in all_connected_clients_dict.keys():
       config = {}
       ins = GetPropertiesIns(config=config)
       res = all_connected_clients_dict[dict_key].get_properties(
ins, 100)
       res = (res.properties)
       all_client_properties.append(res)
     # preprocess to calculte bid Score  and p_value
     client_id = [[client_id["client_id"]] for client_id in
all_client_properties]
     bandwidth_data = [[d["bandwidth"],d["data"]] for d in
all_client_properties]
     bandwidth_data = np.asarray(bandwidth_data)
     print("bandwidth_data",bandwidth_data)

     # NE to calculate Payment Allocation
     scoring_board = list()
     p_client = list()
     payment = list()
     for q in bandwidth_data:
       p_value, total_resource_cost = calculate_p(q)
       p_client.append(p_value)
```

```
      client_bid_socre = scoring_function(q,p_value)
      node_payment = client_bid_socre  + integ(total_resource_cost,client_bid_socre)
      scoring_board.append(client_bid_socre)
      payment.append(node_payment*100)
      print(client_bid_socre, node_payment)

    score_dic = [{"client_id":id , "bid_score" : score , "p_value" : p , "payment" : node_payment } for id, score , p ,node_payment in zip(client_id,scoring_board, p_client,payment)]
    print("Client Scoring Board",score_dic)
    top_k = sorted(score_dic, key=lambda x:x["bid_score"], reverse =True)
    top_k = top_k[:K]
    print("The Client with Higher Score", top_k)
```

Code 4.2: Chosing the top clients

4.5.4 Send model to a number of connected devices

Once we have picked the two best-performing nodes (top scorer), we send the global model's settings to the client nodes that are connected. This helps make sure that all the participating nodes start their local training with the same model settings.

We adopt this approach because involving every client node, especially in scenarios with a large number of them, doesn't necessarily enhance the overall process; in fact, it might lead to decreased efficiency. Therefore, we meticulously select the nodes to participate in this process, aiming to achieve optimal results while taking into account the available computing power and network resources. It's akin to striking the right balance between maximizing everyone's input and ensuring the smooth operation of the system.

4.5.5 Train model locally on the data of each device

Once all the selected client nodes have received the latest version of the global model parameters, they commence their respective local training processes. Each client node utilizes its own local dataset to train its individual local model. Importantly, client nodes do not aim for full convergence during this phase. Instead, they engage in relatively short training intervals, which could be as brief as a single epoch on their local data or even just a few steps using mini-batches. This approach reflects the federated learning paradigm, where

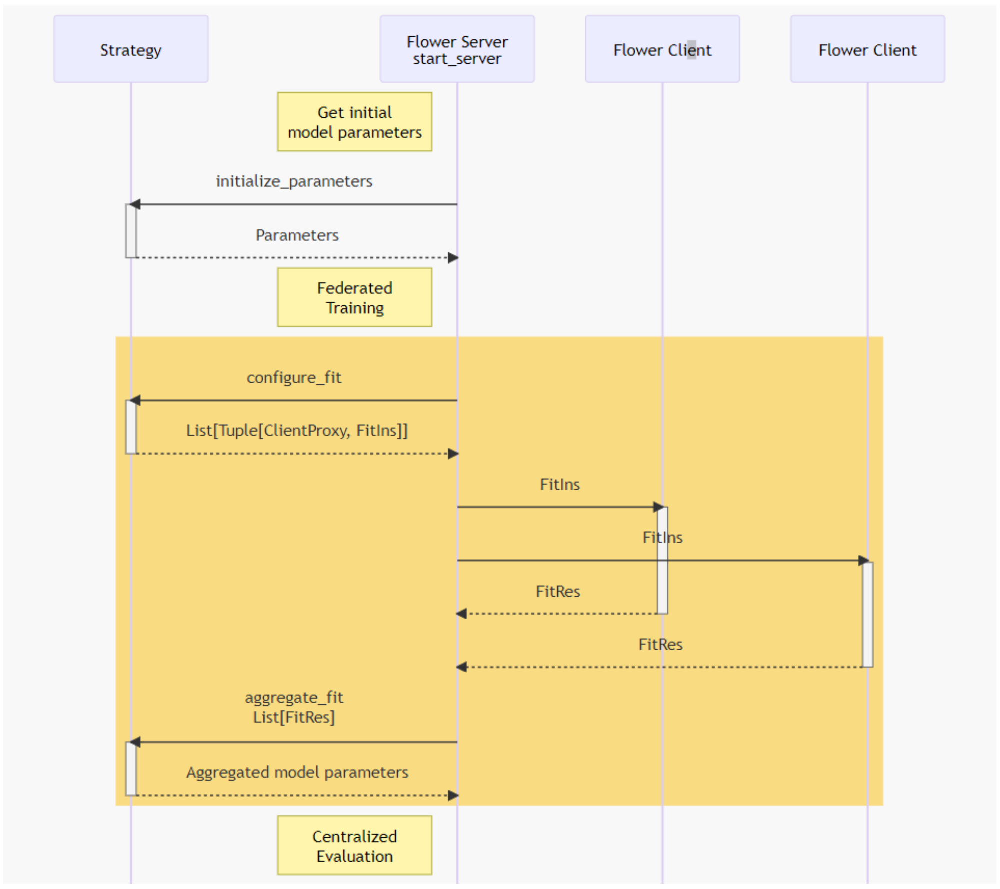


Figure 4.6: Flower WorkFlow

client nodes contribute their incremental knowledge to the global model without necessarily seeking complete convergence. It strikes a balance between retaining data privacy and minimizing computational demands while collectively improving the global model's performance across the federated network.

4.5.6 Return model updates back to the server

After local training, each client node possesses a slightly modified version of the model parameters that were initially distributed to them. These variations arise because each client node's local dataset comprises distinct examples, resulting in model parameter differences.

Subsequently, these client nodes transmit their model updates back to the server. These updates can take the form of either the full model parameters, reflecting the client's refined model, or solely the gradients that accumulated during their local training process. The choice between sending full model parameters or gradients can depend on factors like network bandwidth, privacy considerations, and the specific federated learning approach being employed. Regardless of the format, these updates play a crucial role in collectively improving the global model during the federated learning process.

4.5.7 Aggregate model updates into a new global model

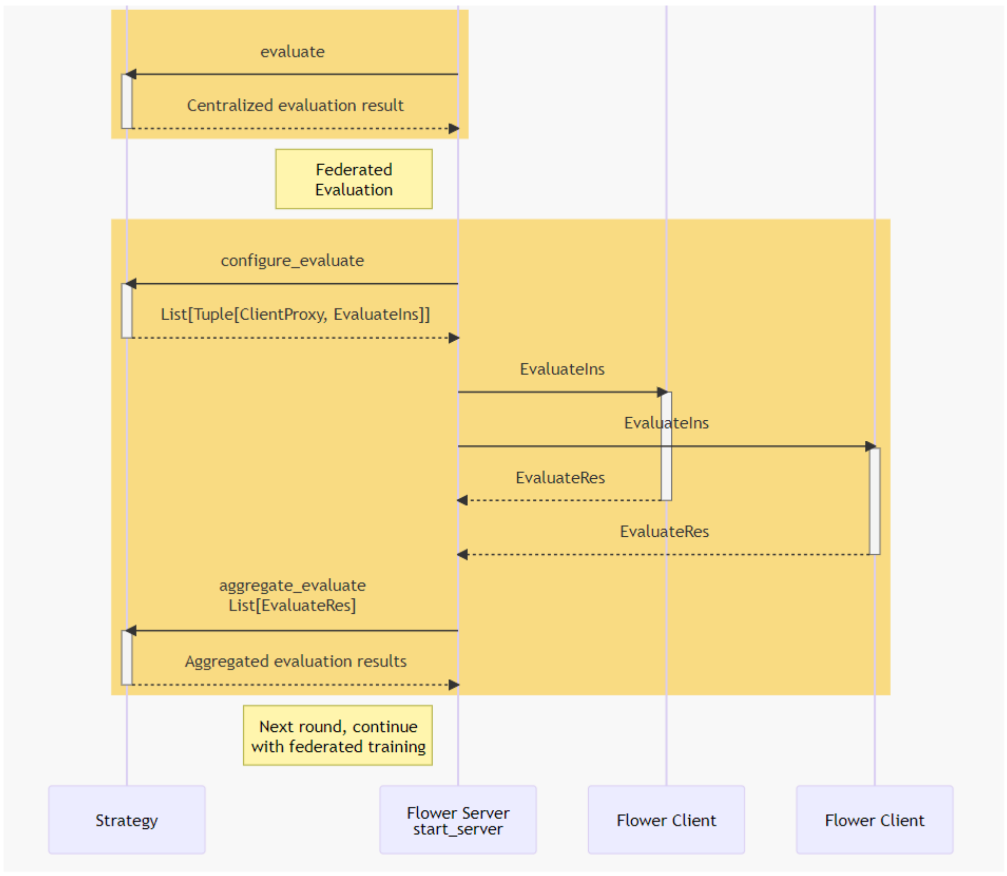

evaluate
Centralized evaluation result
Federated Evaluation
configure_evaluate
List[Tuple[ClientProxy, EvaluateIns]]
EvaluateIns
EvaluateIns
EvaluateRes
EvaluateRes
aggregate_evaluate List[EvaluateRes]
Aggregated evaluation results
Next round, continue with federated training
Strategy
Flower Server start_server
Flower Client
Flower Client

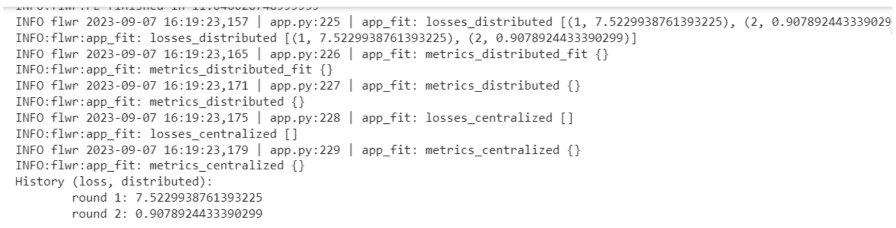
Figure 4.7: Flower Work Flow

The server receives model updates from the two client nodes. In order to get one single model, we have to combine all the model updates we received from the client nodes. This process is called aggregation, and we use FedAvg (described in section) which takes the model updates and averages them. To be more precise, it takes the weighted average of the model updates, weighted by the number of examples each client used for training.

The weighting is important to make sure that each data example has the same influence on the resulting global model. If one client has 10 examples, and another client has 100 examples, then - without weighting - each of the 10 examples would influence the global model ten times as much as each of the 100 examples.

4.5.8 Repeat steps 1 to 4 until the model converges

Steps 1 to 4 are what we call a single round of federated learning. The global model parameters get sent to the participating client nodes (step 1), the client nodes train on their local data (step 2), they send their updated models to the server (step 3), and the server then aggregates the model updates to get a new version of the global model (step 4).



```
INFO flwr 2023-09-07 16:19:23,157 | app.py:225 | app_fit: losses_distributed [(1, 7.5229938761393225), (2, 0.907892443339029
INFO:flwr:app_fit: losses_distributed [(1, 7.5229938761393225), (2, 0.9078924433390299)]
INFO flwr 2023-09-07 16:19:23,165 | app.py:226 | app_fit: metrics_distributed_fit {}
INFO:flwr:app_fit: metrics_distributed_fit {}
INFO flwr 2023-09-07 16:19:23,171 | app.py:227 | app_fit: metrics_distributed {}
INFO:flwr:app_fit: metrics_distributed {}
INFO flwr 2023-09-07 16:19:23,175 | app.py:228 | app_fit: losses_centralized []
INFO:flwr:app_fit: losses_centralized []
INFO flwr 2023-09-07 16:19:23,179 | app.py:229 | app_fit: metrics_centralized {}
INFO:flwr:app_fit: metrics_centralized {}
History (loss, distributed):
        round 1: 7.5229938761393225
        round 2: 0.9078924433390299
```

Figure 4.8: Results

During a single round, each client node that participates in that iteration only trains for a little while. This means that after the aggregation step (step 4), we have a model that has been trained on all the data of all participating client nodes, but only for a little while. We then have to repeat this training process over and over again to eventually arrive at a fully trained model that performs well across the data of all client nodes.

5 Dataset and Results

In this chapter, we'll explore the datasets we used and the outcomes we achieved in our research. Since there was no existing dataset that suited our needs, we crafted a custom dummy dataset. We also put our methods to the test using the widely recognized MNIST and CIFAR10 dataset, which is a standard in the field of machine learning. This section offers a brief summary of these datasets and results, shows that appraoch that we used perform better than the conventional FL strategy.

5.1 Dataset

The dataset we used in our experiment is simulated representation of drone delivery scenarios, encapsulating critical factors that influence the delivery time of a drone. These factors encompass drone speed, parcel weight, battery life, initial and final battery percentages, delivery distance, and both the anticipated and actual delivery duration. Within this dataset, each sample embodies a distinct hypothetical delivery scenario. The generation of this dataset involves the creation of 10000 samples, random values are generated, adhering to predefined ranges for the factors mentioned earlier. To calculate the expected delivery time, the drone's travel time, accounting for its speed and the distance to be covered, is considered. Additionally, the dataset factors in battery usage during delivery. Actual delivery times are introduced with a degree of randomness to mimic real-world variations.This dataset offers a versatile resource for various analytical purposes, including modeling and prediction of drone delivery times, exploring the influence of different variables on delivery efficiency, and the evaluation of machine learning algorithms designed for regression tasks.

Given the non-availability of real-world data in this domain, it's important to note that this dataset has been generated synthetically to bridge this gap and tried to mimic the complexities of real-world scenarios.

5.2 Environment Setting

In practical implementation, we simulated the environment with multiple datasets for multiple clients (a "cross-silo" setting in FL) by partitioning the original dataset. In the real-world scenario, there's no need for data splitting, as each device naturally possesses its own data. Figure 5.1 shows that the general setting that we have applied for starting our simulations. We use *fraction fit = 1* which indicates that it should sample all the connected client to the aggregator and aggregator should wait to start the bidding process until atleat 5 clients are connected to it.

Parameters	**Values**	**description**
Fraction Fit	1	Sample 100% of available clients for training
Fraction Evaluate	1	Sample 100% of available clients for evaluation
Min Fit Clients	5	Never sample less than 5 clients for training
Min Evaluate Clients	5	Never sample less than 5 clients for evaluation
Min Available Clients	5	Wait until all 10 clients are available

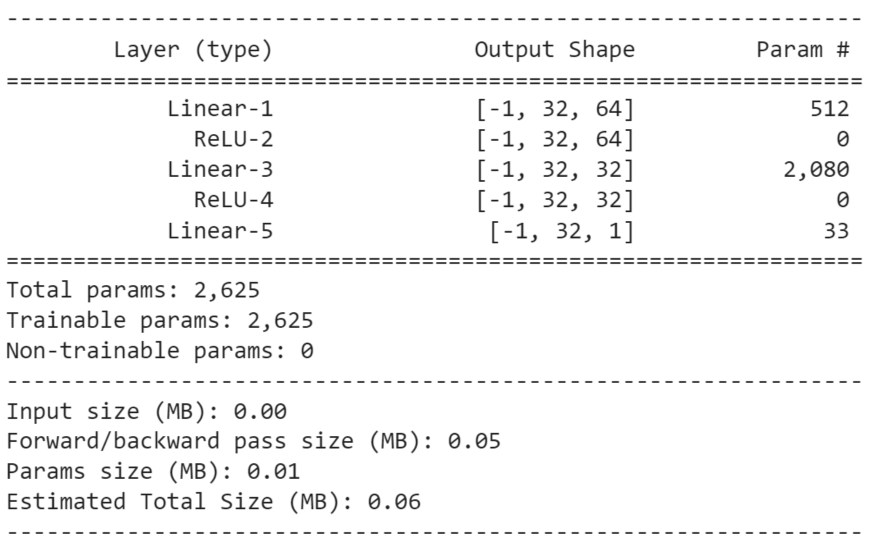
Table 5.1: Environment setting for Flower.

Flower calls client_fn whenever it needs an instance of one particular client to call fit or evaluate, these clients are identified by a client ID. We need two helper functions to update the local model with parameters received from the server and to get the updated model parameters from the local model: *set_parameters* and *get_parameters*.

5.3 Network Model

In our experimentation, we employed a multi-layer perceptron (MLP) for the dummy dataset, where the task primarily involved straightforward linear regression. However, in the interest of evaluating the versatility and robustness of our approach, we extended our testing to more complex datasets, namely MNIST and CIFAR10. For these datasets, we leveraged neural networks to handle more complex classification tasks they present.

Particularly deep learning models, excel in testing on the MNIST dataset for handwritten digit recognition. Their capacity to automatically learn intricate patterns, handle non-linear relationships, and adapt to different complexities makes them a powerful choice. Their scalability, regularization, optimization algorithms, and outstanding performance have established them as the go-to choice for digit recognition on MNIST. Below figure 5.1 shows the model we have used in our experiment.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1               [-1, 32, 64]             512
              ReLU-2               [-1, 32, 64]               0
            Linear-3               [-1, 32, 32]           2,080
              ReLU-4               [-1, 32, 32]               0
            Linear-5                [-1, 32, 1]              33
================================================================
Total params: 2,625
Trainable params: 2,625
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.05
Params size (MB): 0.01
Estimated Total Size (MB): 0.06
----------------------------------------------------------------
```

Figure 5.1: Neural Network Model

The model's hyperparameters, as employed in this study, are detailed in Table 5.3. Specifically, a learning rate of 0.01 was selected, and the Adam optimizer was utilized to facilitate efficient model training. Furthermore, the choice of the cross-entropy loss function was made, as it has demonstrated robust performance in the context of both binary and multi-class classification tasks.

Parameters	**Values**
Batch Size	32
Learning Rate	0.001
Activation Function	Sigmoid
Optimizer	Adam
Loss	CrossEntropyLoss
Epochs	5

Table 5.2: Hyper Parameters for the Model

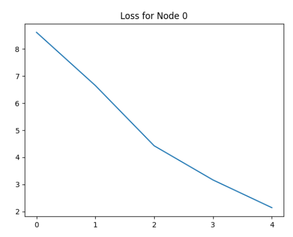
5.4 Dummy dataset result

We have used the dummay dataset to train and test the model. In the interest of comprehensive comparison, we conducted a parallel assessment using the conventional central machine learning approach and FedAvg that is foundation of aggregating in FL and built-in class in flower framework and compared the resutls with our approach.

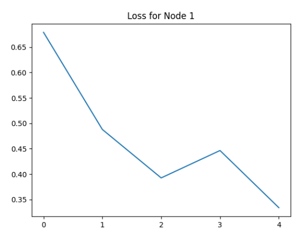
Rounds	**FedAvg Loss**	**Proposed Loss**
1	12.2763	9.7969
2	10.4348	1.7179
3	7.1088	0.4334
4	4.3250	0.3220

Table 5.3: Dummy, Loss

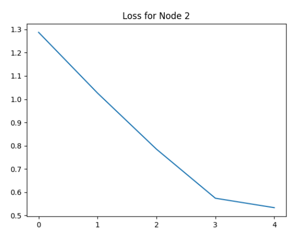
Table 5.3 presents a comprehensive overview of the training rounds applied to our model. A notable observation is the swift loss convergence achieved by our approach, culminating at 0.3220 during the fourth round. This rapid reduction in loss underscores the agility of our method, highlighting its ability to swiftly adapt to the diverse client data while ensuring model stability.In contrast, FedAvg exhibits a comparatively slower loss convergence trajectory from the initial round, indicating a delayed model convergence and a potential increase in communication overhead. Consequently, our approach emerges as a compelling solution, particularly in scenarios characterized by resource constraints or the need to address data distribution disparities among clients, as it offers the potential to expedite federated learning outcomes.



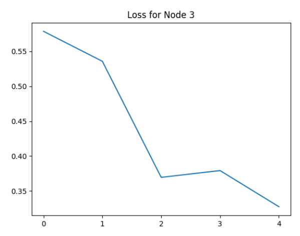
(a) Device 1



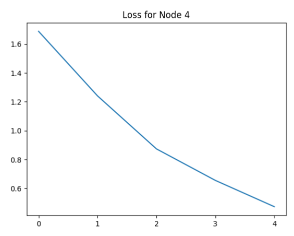
(b) Device 2



(c) Device 3



(d) Device 4



(e) Device 5

Figure 5.2: Connected Devices Loss

Figure 5.2 provides a detailed visualization of loss convergence for the select devices that successfully secured bids and participated in the training process. Specifically, two devices were engaged in each training round. Notably, by the conclusion of the fourth round, a significant outcome is observed: all five devices within the network have undergone training and contributed to the overall learning process. This visual representation offers valuable insights into the progressive involvement of devices and the convergence of loss across multiple rounds of training.

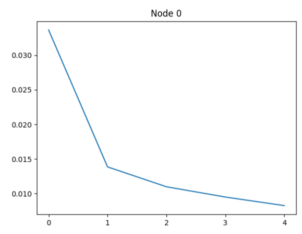
5.5 MNIST Dataset Result

The MNIST dataset is a cornerstone in the field of machine learning and computer vision. It stands as one of the most widely used benchmark datasets for handwritten digit recognition. Comprising a collection of 28x28 pixel grayscale images, MNIST contains a comprehensive set of handwritten digits from 0 to 9. Each image in the dataset is labeled with its corresponding digit.

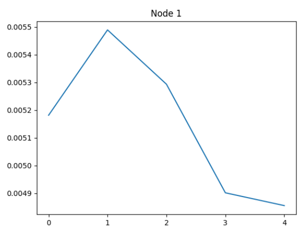
Rounds	**FedAvg Loss**	**Accuracy**	**Our Method Loss**	**Accuray**
1	0.014940	89.87%	0.003849	96.38%
2	0.006084	94.47%	0.002317	97.66%
3	0.004524	95.66%	0.002025	98.05%
4	0.003815	96.22%	0.001897	98.06%

Table 5.4: MNIST Loss & Accruacy

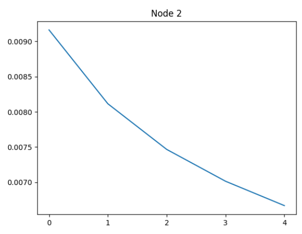
In Table 5.4, we present the results obtained over four training rounds, utilizing loss and accuracy metrics for evaluation. A comparative analysis is conducted between the FedAvg, which serves as the foundational baseline for federated learning, and our proposed methodology.



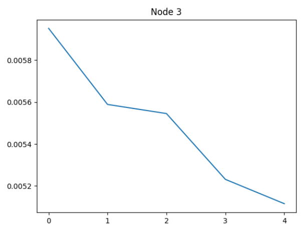
(a) Device 1



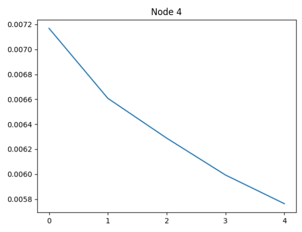
(b) Device 2



(c) Device 3



(d) Device 4



(e) Device 5

Figure 5.3: MNIST Connected Devices Loss

The discernible disparity in loss convergence between the two methods is evident. FedAvg exhibits a protracted convergence process, necessitating a greater number of rounds to reach convergence. Conversely, our approach showcases a more efficient trajectory, commencing with lower initial loss values and demonstrating rapid convergence.Turning our attention to accuracy, we observe noteworthy distinctions. After four rounds, FedAvg attains an accuracy of 96.22%. In stark contrast, our approach exhibits remarkable performance right from the initial round, initiating at an accuracy of 96.38%. By the culmination of the fourth round, our method achieves a substantial accuracy rate of 98.06

Figure 5.4 illustrates the loss convergence for each of the connected devices. It is noteworthy that all five devices actively participated in the training process. Since we employed two devices in each training round, the cumulative result, upon completion of all rounds, demonstrates that every single device contributed to the training process effectively. These results highlight our approach's efficacy in accelerating loss convergence and achieving superior accuracy in fewer training rounds. We have also tried untill 10 rounds and results shows that our approach achieved the better accuracy than the FedAvg and approahes to almost 99%.

5.6 CIFAR10 Dataset Result

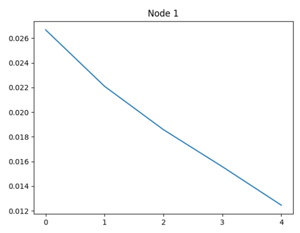
s a collection of images that are commonly used to train machine learning and computer vision algorithms. It features 60,000 small, color images divided into ten distinct categories, encompassing a diverse range of objects and scenes. With its challenging variations and robust size, CIFAR-10 serves as an essential tool for developing and evaluating image classification algorithms and deep learning models.

Rounds	**FedAvg Loss**	**Proposed Loss**	**Loss**	**Accuray**
1	0.054003	38.33%	0.043944	52.38%
2	0.043886	49.33%	0.032357	64.29%
3	0.040138	54.45%	0.034940	66.35%
4	0.037486	57.48%	0.036480	67.33%

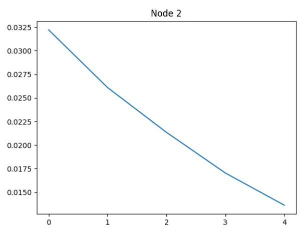
Table 5.5: CIFAR10 Loss & Accuracy

Table 5.5, similar to the MNIST dataset, demonstrate a similar trend in performance. Specifically, our approach demonstrates early convergence to loss compared to the traditional FedAvg methodology. However, when applying the same model to both datasets, we observe a lower accuracy. It is crucial to emphasize that our approach presents a notable advantage in terms of initial accuracy. In the first training round, our method achieves a significantly higher accuracy than the final round of FedAvg. Moreover, by the completion of the fourth round, our approach achieves an impressive accuracy rate of 67.33%. By increasing the number of round, we demonstrated that our approach has early convergence and achieve the better accuracy than the basic FL strategy.

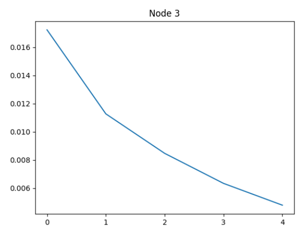
These findings underscore the effectiveness of our approach in achieving rapid convergence to lower loss values and an elevated initial accuracy, ultimately contributing to improved model performance.



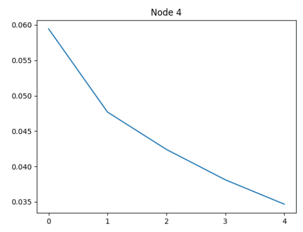
(a) Device 1



(b) Device 2



(c) Device 3



(d) Device 4

Figure 5.4: CIFAR10, Connected Devices Loss

The figure 5.4 provides a visual representation of the training process involving four nodes. Notably, Node 5 did not participate in the training process because it lacked the necessary resources to engage and emerge as the winning candidate in the bidding process.

6 Conclusions and Future Works

The primary objective of this thesis revolved around the development of an automated system tailored to facilitate logistics operations through the utilization of Unmanned Aerial Vehicles (UAVs). Notably, the logistical landscape is characterized by the deployment of diverse UAV devices, each possessing distinct attributes and generating data of varying natures. This inherent heterogeneity within the system posed a substantial challenge.Given that the devices do not voluntarily participate in the training process, a strategy was devised to incentivize their engagement through the provision of rewards.

To ensure fair distribution of rewards among participating clients in proportion to their resource contributions, we adopted the Nash Equilibrium (NE) strategy. This approach enables us to allocate the most optimal rewards to each client, with the highest-performing participants being selected for subsequent training rounds. Indeed, it's important to emphasize that the system's inherent diversity makes traditional machine learning methods ineffective. As a result, we transitioned to the domain of Federated Learning, which provides the capacity to train localized models on individual devices utilizing their specific datasets.

By employing the Federated Learning framework and running multiple training rounds, our research has uncovered an intriguing phenomenon: early convergence of loss. In contrast, the traditional FedAvg approach requires a larger number of rounds to achieve this convergence. This rapid loss convergence highlights our model's impressive capability to adjust to the inherent variations in data distribution among participating clients while maintaining model stability and robustness.These results showcase a significant advantage in terms of swift synchronization of model parameters. This further emphasizes the model's potential to improve the optimization of learning processes within a distributed and diverse data environment.

6.1 Future Work

There are certain limitations to this work which would ideally be done for future work. One significant limitation is that our model was trained using artificial data, which doesn't fully mimic real-world situations. To address this, future work could involve creating a more realistic dataset for training and testing, closely mirroring the complexities of practical logistics scenarios.

Another exciting direction for future research involves exploring reinforcement learning (RL). Unlike traditional methods we used, RL lets an agent learn by interacting with its environment. It receives feedback, like rewards or penalties, based on its actions, gradually learning a strategy that maximizes rewards over time. RL is valuable in situations where we don't know the best strategy upfront but need to learn it through trial and error, often referred to as 'reward-based learning.'

Incorporating RL into future research could lead to the development of intelligent agents capable of autonomously optimizing logistics operations. These agents would learn how to make decisions and adapt their strategies based on real-world feedback, making logistics processes more adaptable and efficient in dynamic, unpredictable settings. In summary, while our current work lays a strong foundation, it also highlights exciting possibilities for future research that could significantly advance logistics optimization.

References

[1] M. F. Aktas, P. Peng, and E. Soljanin. "Effective Straggler Mitigation: Which Clones Should Attack and When?" In: *ACM SIGMETRICS Performance Evaluation Review* 45.2 (2017), pp. 12–14.

[2] Harikrishnan Nellippallil Balakrishnan et al. "ChaosNet: A chaos based artificial neural network architecture for classification". In: *Review of Chaos: An Interdisciplinary Journal of Nonlinear Science* (2019). DOI: 10.1063/1.5120643.

[3] Pourya Behmandpoor, Panagiotis Patrinos, and Marc Moonen. "Federated Learning Based Resource Allocation for Wireless Communication Networks". In: *2022 30th European Signal Processing Conference (EUSIPCO)*. 2022, pp. 1656–1660. DOI: 10.23919/EUSIPCO55093.2022.9909708.

[4] Keith Bonawitz and et al. "Towards Federated Learning at Scale: System Design". In: *CoRR* abs/1902.01046 (2019).

[5] Samuel Bowles and Sandra Polanía-Reyes. "Economic Incentives and Social Preferences: Substitutes or Complements?" In: *Journal of Economic Literature* 50.2 (2012), pp. 368–425. URL: http://www.jstor.org/stable/23270024.

[6] Fernando Branco. "The Design of Multidimensional Auctions". In: *RAND Journal of Economics* 28.1 (Spring 1997), pp. 63–81.

[7] Facundo Bre and Fachinotti Gimenez. "Prediction of wind pressure coefficients on building surfaces using artificial neural networks". In: *Elsevier* (2018).

[8] P. Briest, P. Krysta, and B. Vocking. "Approximation techniques for utilitarian mechanism design". In: *Proceedings of the Annual ACM Symposium on Theory of Computing* 40 (2011), pp. 1587–1622.

[9] Y. Che. "Design Competition Through Multidimensional Auction". In: *RAND Journal of Economics* 24.4 (1993), pp. 668–680.

[10] X. Chen et al. "Exploiting massive D2D collaboration for energy-efficient mobile edge computing". In: *IEEE Wireless Communications* 24.4 (2017), pp. 64–71.

[11] J. Dean and L. A. Barroso. "The Tail at Scale". In: *Communications of the ACM* 56.2 (2013), pp. 74–80.

[12] N. Ding, Z. Fang, and J. Huang. "Incentive Mechanism Design for Federated Learning with Multi-Dimensional Private Information". In: *International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*. 2020.

[13] Ningning Ding et al. "Incentive Mechanism Design for Distributed Coded Machine Learning". In: *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications* (2020), pp. 1–10. URL: https://api.semanticscholar.org/CorpusID:229221600.

[14] K. Doppler et al. "Device-to-device communication as an underlay to LTE-Advanced networks". In: *IEEE Communications Magazine* 47.12 (2009), pp. 42–49.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[16] Eider Moore H. Brendan McMahan, Seth Hampson Daniel Ramage, and Blaise Aguera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data." In: *JMLR* (2023).

[17] John Harsanyi. "Games with Incomplete Information Played by ?Bayesian? Players, I?III". In: *Management Science - MANAGE SCI* 50 (Jan. 2004), pp. 1804–1817. DOI: 10.1007/978-94-017-2527-9_8.

[18] Nathalie Baracaldo Heiko Ludwig. *Federated Learning, A Comprehensive Overview of Methods and Applications*. [Online].Available:https://link.springer.com/book/10.1007/978-3-030-96896-0. Springer Nature Switzerland AG, 2022.

[19] Heiko Hotz. *A Short Introduction to Game Theory*. RAND Journal of Economics, pp. 2–3. URL: https://www.theorie.physik.uni-muenchen.de/lsfrey/teaching/archiv/sose_06/softmatter/talks/Heiko_Hotz-Spieltheorie-Handout.pdf.

[20] drohnen in der intralogistik. *Drohnen in der Intralogistik: Geflügelter Lieferservice*. 03/04/2023. URL: https://www.produktion.de/trends-innovationen/drohnen-in-die-intralogistik-gefluegelter-lieferservice-108.html.

[21] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML*. 2015.

[22] Junqi Jin et al. "Real-time Bidding with Multi-agent Reinforcement Learning in Display Advertising". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management - CIKM '18*. 2018.

[23] Noreen Khan et al. "MACRS: An Enhanced Directory-Based Resource Sharing Framework for Mobile Ad Hoc Networks". In: *Electronics* 11 (Feb. 2022), p. 725. DOI: 10.3390/electronics11050725.

[24] H. Kim et al. "On-device Federated Learning via Blockchain and its Latency Analysis". In: *arXiv preprint arXiv:1808.03949* (2018).

[25] T. Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *arXiv preprint arXiv:1908.07873* (2019).

[26] Zhengyang Li et al. "Federated Split BERT for Heterogeneous Text Classification". In: *arXiv preprint arXiv:2205.13299* (May 2022). Version 1. arXiv: 2205.13299.

[27] Jinjin Liu et al. "Prediction of rupture risk in anterior communicating artery aneurysms with a feed-forward artificial neural network". In: *European Radiology* 28.8 (2018), pp. 3268–3275. DOI: 10.1007/s00330-017-5201-y.

[28] Q. Ma et al. "Incentivizing Wi-Fi Network Crowdsourcing: A Contract Theoretic Approach". In: *IEEE/ACM Transactions on Networking* 26.3 (2018), pp. 1035–1048.

[29] H. McMahan et al. "Federated Learning of Deep Networks using Model Averaging". In: *arXiv preprint arXiv:1602.05629v2* (2017).

[30] Flavio M. Menezes and Paulo K. Monteiro. "Private Values". In: *Auction Theory*. Oxford University Press, 2004, pp. 13–38. DOI: 10.1093/019927598X.003.0003.

[31] Kathryn Merrick and Kamran Shafi. "A game theoretic framework for incentive-based models of intrinsic motivation in artificial systems". In: *Frontiers in Psychology* 4 (2013). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2013.00791. URL: https://www.frontiersin.org/articles/10.3389/fpsyg.2013.00791.

[32] Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1997. ISBN: 978-0-674-34116-6.

[33] John Nash. "Equilibrium Points in n-Person Games". In: *Proceedings of the National Academy of Sciences* 36.1 (1950), pp. 48–49.

[34] Geoffrey Hinton Nitish Srivastava, Ilya Sutskever Alex Krizhevsky, and Ruslan Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting." In: 15, 2014.

[35] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. [Online]. Available:https://arielrubinstein.tau.ac.il/books/GT.pdf. MIT Press, 2012-9-24.

[36] Xumin Pu et al. "Incentive Mechanism and Resource Allocation for Collaborative Task Offloading in Energy-Efficient Mobile Edge Computing". In: *IEEE Transactions on Vehicular Technology* (2023), pp. 1–6. DOI: 10.1109/TVT.2023.3274513.

[37] Christoph Raab and Frank-Michael Schleif. "Transfer Learning for the Probabilistic Classification Vector Machine". In: *7th Symposium on Conformal and Probabilistic Prediction and Applications, COPA 2018*. Vol. 91. Proceedings of Machine Learning Research. PMLR, 2018, pp. 187–200.

[38] Mohsen Soori. Roza Dastres. "Artificial Neural Network Systems." In: *International Journal of Imaging and Robotics (IJIR)* (2021), pp. 13–25.

[39] Martin J. Osborne Ariel Rubinstein. *MODELS IN MICROECONOMIC THEORY*. [Online]. Available:https://books.openbookpublishers.com/10.11647/obp.0211.pdf. open book publisher, 2020.

[40] S. Shi et al. "Communication-efficient Distributed Deep Learning with Merged Gradient Sparsification on GPUs". In: 2020.

[41] D. Yang et al. "Incentive Mechanism for Crowdsensing: Crowdsourcing with Smartphones". In: *IEEE/ACM Trans. on Networking (TON)* 24.3 (2016), pp. 1732–1774.

[42] Xun Yang et al. "Federated Learning Incentive Mechanism Design via Shapley Value and Pareto Optimality". In: *Axioms* 12 (June 2023), p. 636. DOI: `10.3390/axioms12070636`.

[43] Rongfei Zeng et al. "FMore: An Incentive Scheme of Multi-dimensional Auction for Federated Learning in MEC". In: *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. 2020, pp. 278–288. URL: `arXiv:2002.09699`.

[44] Dietlind Zühlke et al. "Learning Vector Quantization for Heterogeneous Structured Data". In: *ESANN 2010*. 2010.

Acknowledgments

I would like to thank my supervisor Prof. Frank Michael Schleif and my co-supervisor Prof. Pietro Zanuttigh for guiding me on this path and leaving me to experiment with my knowledge; their assistance and their suggestions have been paramount many times during this thesis work. I would also like to thank the University of Padova, which allowed me to learn a lot of beautiful stuff during these years, and CAIRO center, which hosted me for six months making me feel at home and giving me inspiration for my future life.

I am profoundly grateful to my parents and siblings for their unwavering encouragement, understanding, and patience throughout this academic pursuit. I extend my appreciation to my colleagues and peers for their stimulating discussions and valuable feedback during the course of this research.

Lastly, I would like to acknowledge the support of Mr. Raza Ul haq for his contributions throught this entire journey.This research would not have been possible without the collective support and encouragement of these individuals and institutions. Thank you all for being a part of this journey.