Federated Learning Meets Blockchain: A Kafka-ML Integration for reliable model training using data streams

1st Antonio Jesús Chaves
ITIS Software
University of Málaga
Málaga, Spain
chaves@uma.es

2nd Cristian Martín
ITIS Software
University of Málaga
Málaga, Spain
cristian@uma.es

3nd Kwang Soon Kim
School of Electrical and Electronic Engineering
Yonsei University
Seoul, South Korea
ks.kim@yonsei.ac.kr

4nd Adnan Shahid
IDLab, Department of Information Technology
Ghent University – imec
Ghent, Belgium
adnan.shahid@ugent.be

5rd Manuel Díaz
ITIS Software
University of Málaga
Málaga, Spain
mdiaz@uma.es

***Abstract*—Machine learning data privacy has been improved with Federated Learning approaches. However, some obstacles to guaranteeing traceability, openness, and participant contribution incentives prevent its widespread use. In this study, Ethereum blockchain technology is integrated into the data stream Kafka-ML framework, presenting a novel asynchronous and blockchain-based Federated Learning approach. By utilising Ethereum for transparent and auditable participant tracking, this integration overcomes some shortcomings such as auditability and model sharing reliability. Furthermore, Ethereum smart contracts allow for automatic reward distribution systems, which promote equitable incentive systems and increased involvement in the Federated Learning process. To demonstrate its potential, an extensive evaluation has been carried out on a wireless network technology detection use case. By improving transparency, traceability, and incentive structures of Federated Learning, it is expected to strengthen the robustness of flexible machine learning collaboration with data streams.**

***Index Terms*—Kafka-ML, Data Streams, Deep Learning, Blockchain, Federated Learning**

I. Introduction

Machine learning (ML) paradigms have changed with the introduction of Federated Learning (FL), especially in situations where strict data security and privacy requirements are required. Kafka-ML [1], an open-source framework for integrating ML with data streams, has made an important contribution by allowing for seamless and flexible ML collaboration with data streams [2]. Without centralising sensitive data, Kafka-ML enables decentralised model training across many data sources, protecting privacy and improving scalability.

By communicating just model updates rather than raw data, FL is a general approach that facilitates distributed model training while preserving data privacy. Kafka-ML, a framework that supports the implementation of FL, enables different data custodians to work together training models. Several obstacles still exist in spite of these developments. Keeping an unchangeable log of every training session, promoting fairness and incentives for participants, and guaranteeing contribution transparency are a few of these. The acceptance and efficacy of FL frameworks may be impeded by the absence of efficient solutions for tracking contributions and allocating rewards [3].

Blockchain technology (in this case Ethereum) [4], addresses the challenges of promoting fairness and ensuring transparent contributions among participants. All transactions and events are recorded in its decentralised and immutable ledger, improving accountability by offering a permanent and verifiable history of contributions, meaning that each participant's input can be traced. Integrating Ethereum's capabilities into the Kafka-ML FL process can significantly improve transparency, traceability, and accountability, building a fairer environment where all contributions are easily verifiable.

The integration of Ethereum blockchain technology into Kafka-ML can provide a reliable system for the recording of all training-related operations on an immutable ledger while maintaining the data stream support for ML models. This ensures that each participant's efforts are publicly monitored and verifiable, which builds confidence among collaborators. Furthermore, the Ethereum blockchain's usage of smart contracts makes it possible to automate reward systems. Participants can be compensated for their contributions, with awards distributed in a transparent and tamper-proof manner.

In recent years, the integration of Electric Vehicles (EVs) and Vehicle-to-Everything (V2X) ecosystems has attracted a great deal of attention due to the potential benefits in terms of connectivity, efficiency and safety. Seamless communication between EVs and the surrounding infrastructure is fundamental to the optimal functioning of these systems. A critical aspect for this interaction is the accurate characterisation of the

different wireless communication technologies. Considering the significant volumes of data generated by these systems, leveraging big data analytics is essential to extract valuable insights and enable intelligent decision-making. This study leverages a dataset of IQ samples from multiple wireless technologies to develop a ML model that can effectively characterise these technologies. By employing a FL approach in which EVs collect, train and share models with a central server located at charging stations, this research aims to enhance the collaborative nature of EVs and V2X networks.

In this work, a novel redefinition to improve the transparency, traceability, and incentive structure of Kafka-ML's FL feature thanks to blockchain is described. In addition to address the current issues, this integration creates new opportunities for fair and safe ML collaboration with data streams, aiming to create a more reliable and robust FL environment.

The rest of the paper is structured as follows. Section II presents the related work and their differences against the implementation of blockchain-based federated training into Kafka-ML. Section III presents the redefined architecture of Kafka-ML. Section IV shows the components and the pipeline of a Blochain-based federated training job in Kafka-ML. Section V provides an evaluation over different test cases. Finally, Section VI provides a concise overview of the study and explores potential avenues for future growth.

II. Background and Related Work

Kafka-ML [1] is an open-source framework for managing ML pipelines via data streams, supporting different popular ML libraries (TensorFlow and PyTorch), GPU acceleration and training and inference natively with data streams [5]. It allows different training techniques such as distributed learning [6], FL [2] or online training [7], being able to combine some of these techniques according the user's needs [8]. It provides a user-friendly web interface for both experts and non-experts to handle the ML/AI pipeline. Users can define models similarly to how they would do so on their local environments. Once models are set up, they can be trained, evaluated, and inferred easily through the Web UI.

The identification of Radio Access Technologies (RATs) via different spectrum sensing technologies has been the subject of extensive research [9]. Traditional methods, such as energy detection and matching filter detection, have provided preliminary solutions, but they have problems in detecting numerous, simultaneous wireless signals. Recent developments use deep neural networks to increase the accuracy of recognition over several RATs, including Wi-Fi and LTE. For these kinds of dispersed training activities, FL methodology and Kafka-ML works well because of their capabilities.

Asynchronous FL is a variation of FL in which the devices update their models independently without needing to synchronise with each other [10]. It reduces delays caused by slower devices, improving overall efficiency in heterogeneous ecosystems. This technique makes faster model updates possible while maintaining accuracy in different environments. This flexibility is critical in applying FL to diverse real-world scenarios.

The integration of blockchain technology into FL systems has been intensively researched to improve trustworthiness, accountability, and fairness of training tasks. Google introduced FL in 2016 to address concerns about communication costs and privacy by enabling decentralised model training using local data [11]. However, FL has a lot of issues with the quantification of equity metrics (client fairness is difficult to assess) and the robustness of accountability measures (hard to track each individual contribution). The absence of means to check local models and ensure data integrity in traditional FL systems might result in malevolent behaviour, as it allows malicious participants to introduce tampered or poisoned data into the system. Numerous scholars have suggested blockchain-based methods to deal with these problems. Others use blockchain for safe exchange and verification of local model updates, guaranteeing client accountability and guarding against erroneous model updates. FLChain [12], for instance, introduces an auditable and decentralised system that rewards honest participants and identifies malicious nodes.

A trustworthy FL architecture based on blockchain especially made for COVID-19 detection by X-rays is introduced at [13]. This architecture implements a smart contract-driven and data-model provenance registry to maintain and record local data and model versions, leveraging blockchain technology and smart contracts to ensure data integrity and auditability. In addition, a weighted fair training data sampler algorithm is suggested to handle heterogeneity in data and enhance the FL model's fairness. This method shows how blockchain technology can be used to improve FL systems' fairness and accountability, thereby enhancing the credibility of AI applications in this field.

Similarly, VFChain [14] offers a blockchain-based verifiable and auditable FL system. In order to record verifiable proofs and collectively aggregate models, VFChain employs a committee selection procedure via blockchain, improving robustness and guarding against manipulation. Additionally, it facilitates safe committee rotation, which increases auditability, and adds an authenticated data structure to boost verification proof efficiency. Another similar framework is BlockFed [15], which is designed to enhance blockchain-based FL systems by addressing critical challenges such as communication overhead, privacy concerns, and vulnerability to poisoning attacks. They manage to address this problems by employing sparse local training, compressing model updates using singular value decomposition, and injecting differential privacy noise. Zhang et al. [16] propose a FL framework that integrates blockchain technology and tackle the challenges associated with non-independent and identically distributed (non-IID) data.

Although all of these frameworks provide solutions for adopting blockchain-based FL, there are several features which are missing in comparison with Kafka-ML. First of all, none of them is open source, or at least no reference or example of any source code has been found, which limits the applicability

and reproducibility of those solutions. In addition, all of them seem to work with static datasets and not with dynamic and real-time data streams. One of the advantages of using Kafka-ML is the ability to work directly with data streams, enabling the adoption of use cases related to Big Data or IoT. Finally, the approaches studied seem to work synchronously between models and clients, which is not an efficient approach when working with IoT devices due to possible disconnections or resource imbalances between the different involved clients.

There are other frameworks that work asynchronously and in a similar way to that proposed in this paper. Some of those frameworks are BAFL [17], AFLChain [18] or BCAFL [19]. By using different algorithms or variations in blockchain utilisation, model aggregation can be performed asynchronously. However, the same limitations persist: while previous solutions are not open-source, a more critical issue is their inability to work directly with data streams, making them less adaptable for real-time applications.

III. Blockchain-enabled Kafka-ML FL Architecture Overview

Kafka-ML orchestrates ML/AI jobs by using data streams (streamed information at Kafka). In a previous work a new architecture was defined to support the FL training procedure [2]. In this work, slight changes has been made into the architecture to allow blockchain integration with Kafka-ML and FL.

As per the single responsibility principle, the Kafka-ML architecture comprises a set of services, which together form a microservice-based system. To provide isolation and ease of portability, every component resides inside Docker containers. Through Kubernetes orchestration, this platform is managed and deployed across a cluster of nodes and dispersed production infrastructures, enabling continuous monitoring of Kafka-ML services and deployed training tasks. In this work, no new modules has been developed for Kafka-ML, but some of them has been modified in order to be able to interact with blockchain. Also, in order to be able to work with a blockchain in an easy way, a private Ethereum node has been deployed in the cluster for this proof of concept.

Fig. 1 depicts the current architecture of Kafka-ML, presenting the different functionalities separated by modules, highlighting the modifications presented in this work and emphasising the separation between Kafka-ML Platform (the original framework with slight modifications) and Kafka-ML Federated (the module that enables federated model training using data streams).

Now, the Kafka-ML Platform will leverage some details of the deployment and orchestration of the federated process into blockchain. The way how Kafka-ML Federated clients work has been slightly modified to be able to read key details from blockchain. In the next subsections the main components are detailed.

A. Kafka-ML Platform Architecture

The Kafka-ML framework (Fig. 1a) is composed of frontend and backend modules that manage user interactions such as model creation, configuration deployment, and metric visualisation. These modules, along with their backend counterparts, have been slightly adjusted to gather additional information when deploying federated models with blockchain tracking. This allows for the recording of every action on the blockchain during the federated training process, which will be further explained in Section IV-B. Despite these modifications, model creation remains unchanged, allowing users to easily define models through the user-friendly interface, even those without technical expertise.

One of the modified elements is the "Federated Training Module". It coordinates training across devices in a federated way, interacting with both Apache Kafka and the chosen Ethereum blockchain to store model weights and metadata. Upon initiating training through "Kafka-ML Federated", the module awaits trained weights for aggregation and redistribution, ensuring that every federated client receives the latest update. The model weights from "Kafka-ML Federated" clients are combined to converge towards a consistent model.

The remaining Kafka-ML Platform architecture modules remain unaltered. The "Platform Data Control Logger" collects metadata from user data sent for training to generate the datasources that the models will target. The "Model training" and "Model inference" modules are deployed when a user wants to train or serve a model, respectively. Training options include classic training in TensorFlow and PyTorch [5], distributed model training [6], incremental model training [7], or a combination of these with GPUs if specified [8]. For inference, users can deploy models by specifying brokers, Apache Kafka input/output topics, replication settings, and deployment details for Kubernetes clusters.

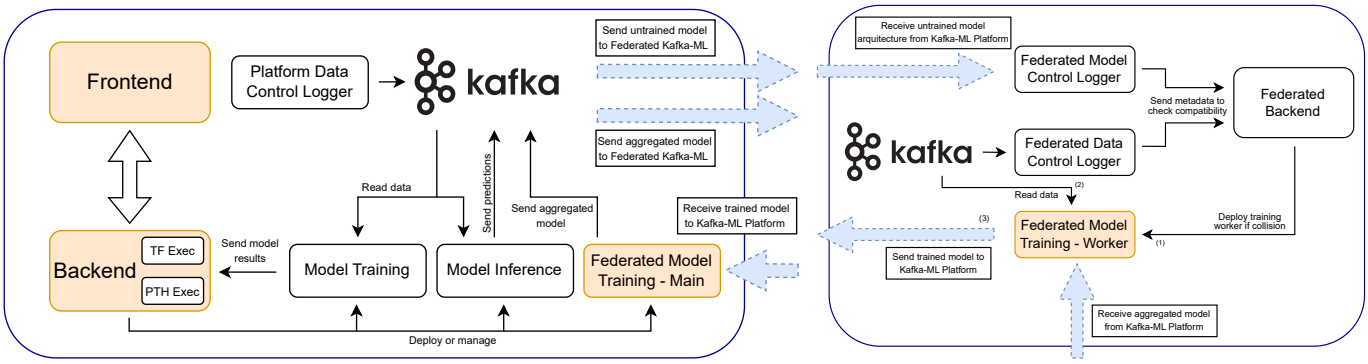
Kafka-ML, along with its latest enhancements, implementation, configuration files, and several illustrative examples, can be found in the GitHub repository[1].

B. Kafka-ML Federated Architecture

Four components comprise up the "Kafka-ML Federated" architecture (Fig. 1b), allowing asynchronous federated model training. The "Federated Backend", "Federated Data Control Logger", "Federated Model Control Logger", and "Federated Model Training - Worker" are these modules. To protect data privacy, every "Kafka-ML Federated" client instance runs a separate Apache Kafka deployment.

The "Federated Data Control Logger" and "Federated Model Control Logger" components constantly monitor different Apache Kafka topics. The main goal of this monitoring is to collect metadata related to the models and data, and if their metadata matches, training jobs may be deployed. All gathered data is sent to the "Federated Backend", which handles requests from the previously stated loggers. Also, it evaluates if the models and the incoming data streams are compatible. The module starts a new Kubernetes job to start a "Federated Model Training - Worker" if compatibility is confirmed. These modules remain unaltered.

[1]Kafka-ML GitHub repository: https://github.com/ertis-research/kafka-ml



(a) Kafka-ML Platform

(b) Kafka-ML Federated

Fig. 1: Kafka-ML Architecture divided into Kafka-ML Platform and Kafka-ML Federated modules. Orange component: modified

Models are obtained and trained using data streams from the federated module in the "Federated Model Training - Worker" component. This module reconstructs the model and trains it using data streams by utilising the metadata that was previously acquired from the "Federated Backend" about the models that are available on the Kafka-ML Platform (Fig. 1b, step 1), in addition to the data streams that are available from Kafka-ML Federated, guaranteeing the privacy of information (Fig. 1b, step 2). Finally, it sends the trained model back to the Kafka-ML Platform so that it may be aggregated and the asynchronous federated training cycle can continue (Fig. 1b, step 3). In this new version, rather than working with control topics for the management of transmission and reception of model weights, this workload is derived to the blockchain, so that it traces by whom and when this operation has been carried out.

C. Blockchain integration to the Architecture

The blockchain integration within the "Kafka-ML Federated" architecture adds a layer of security and transparency to the model training and transmission process. Instead of relying on control topics for managing the transmission and reception of model weights, the blockchain now handles this responsibility. Each time a "Federated Model Training - Worker" completes a training cycle, the resulting model weights, along with a timestamp and the worker's identity, are recorded as a transaction on the blockchain. This ensures that every model update is immutably logged, establishing a clear and verifiable history of the model's evolution.

Additionally, the blockchain's transparency can be leveraged to provide stakeholders with insights into the model training process. By querying the blockchain, users can trace the lineage of a model, identifying which workers contributed to its training and when updates were made. This information can be valuable for auditing purposes, ensuring compliance with regulatory requirements, and fostering trust in the FL process.

Furthermore, the blockchain can facilitate a reward system where contributors are rewarded based on their contributions to the model's development. This can be achieved by assigning tokens or other forms of value to successful model updates, encouraging active participation and rewarding users for their valuable contributions.

IV. Kafka-ML & Blockchain Integration

This section explores the integration of blockchain within the FL approach of the Kafka-ML framework, managed and orchestrated by smart contracts deployed on the Ethereum blockchain to ensure transparency and traceability in the FL process.

A. Smart Contract for FL process traceability

A smart contract is an agreement that has its terms encoded directly into code. When certain requirements are satisfied, they operate on the Ethereum blockchain and automatically enforce and execute the rules. Ethereum's smart contracts are useful because they may provide confidence and transparency in transactions by facilitating, confirming, and enforcing the negotiation or fulfilment of an agreement without the need for middlemen.

Smart contracts are essential in the FL process at Kafka-ML, governing clients contributions. They facilitate the FL process by organising the acquisition and distribution of model updates and ensure the integrity and accuracy of the shared models.

Furthermore, a smart contract has been developed that stores all the information of the queue of models to be aggregated as well as the last global model updated continuously by federated clients. The entire contract is openly available and free to use on Kafka-ML's GitHub Repository[2]. Algorithm 1 shows how the different training elements work, emphasising the functions that interact with the blockchain. Clients now use functions like *getLastGlobalModel* or *sendModel* to get the latest model or send their contribution while the platform use *saveGlobalModel* to set the latest model available or *getQueueSize* to fetch one of the contributions and aggregate them, being all of this interactions written into the blockchain and thereby creating the cyclical procedure of FL procedures.

[2]Kafka-ML blockchain-Based FL Smart Contract: https://github.com/ertis-research/kafka-ml/blob/master/model_training/tensorflow/contracts/FederatedLearning.sol

Algorithm 1: Algorithm for blockchain-based FL procedure at Kafka-ML

Generic Variables: $Contract$, $Topic_{Agg}$, $Config_{Kafka}$
Note: Blockchain-related functions are underlined.
Procedure at Kafka-ML Platform
Input Data : $Model_0$, $maxAggRounds$, $Config_{Train}$, $currRound = 0$
Output Result : $Model_{Global}$, $Metrics$
$Model_{Global} = Model_0$, $Metrics = \{\}$
while $currRound \leq maxAggRounds$ **do**
  saveGlobalModel($Model_{Global}$, $Config_{Train}$, $currRound$, $Contract$)
  while *getQueueSize*($Contract$) ≤ 1 **do**
    // wait
  end
  $Model_{currRound}$ = dequeueModel($Contract$)
  $Model_{Global}$ = weightsAverage($Model_{Global}$, $Model_{currRound}$)
  calculateReward($Model_{currRound}$, $Contract$)
end
stopTraining($Contract$)
distributeTokens($Contract$)
sendFinalMetrics($Model_{maxAggRounds}$)
Procedure at Federated Kafka-ML Client
Input Data : $Contract$, $Config_{Kafka}$
while *isTrainingActive*($Contract$) **do**
  $Model_{currRound}$, $Config_{Train}$ = getLastGlobalModel($Contract$, $Config_{Kafka}$)
  $trainedModel$ = train($Model_{currRound}$, $Config_{Train}$)
  sendModel($trainedModel$, $Contract$, $Config_{Kafka}$)
end

B. Blockchain integration into the Kafka-ML FL process

In this section, the pipeline for a blockchain-based asynchronous FL task in Kafka-ML is explained, highlighting the key components that has been modified to interact with the blockchain and trace the process. Fig. 2 shows a sequence diagram describing the whole procedure.

The user will first define the architecture of the models to be deployed, just by inserting the source code of the model in the Web UI. If desired, they can also group the models into configurations and deploy them all at once. The user will need to fill out a few more fields after creating the setup and deciding to deploy the models for federated training. These parameters mostly contain the number of rounds of aggregation, the technique of aggregation to be employed, and some requirements that the data should adhere to in order to train the model. Since all of the blockchain configuration up to this point was made during the Kafka-ML Platform deployment, this process is identical to the non-blockchain Kafka-ML federated training approach.

The model is then set up as the genesis model once it is deployed for federated training. Its architecture and metadata are published in a newly generated instance of the Smart Contract at the blockchain. As a result, a new model is created, and all deployed Kafka-ML Federated clients —which are always listening to the "Model Control Topic"— may be notified. If the model is ready for blockchain-based training, they can then configure everything necessary to communicate with the appropriate Smart Contract instance.

Upon discovering a new model, a Kafka-ML Federated client examines its instance of Apache Kafka to check if any data is available for training. The process of matching involves using schema matching, which is a JSON representation of the properties of the dataset, to verify features of the dataset such as shape and properties (e.g., number of classes, labels, etc.). The Kafka-ML Federated client will deploy a "Federated Model Training - Worker" task if matching schemes and dataset shapes match.

In this job, the model is loaded using the details from the Smart Contract, trained using data from the local Apache Kafka, and then returned to Kafka-ML Platform by using the Smart Contract (metadata and weights location at Kafka) and Apache Kafka. The Smart Contract is used to store the model metadata and where the weights are located at the Platform Apache Kafka (as store weights at blockchain is a problem because their size).

Continuing with the training, the "Federated Model Training - Worker" at Kafka-ML Platform will retrieve the models and weights from the Smart Contract queue and aggregate them, obtaining the resulting model as the new global model. Each of these actions is considered an aggregation round. These aggregation rounds will be repeated as many times as chosen by the user. Kafka-ML Federated training clients always read the latest global model written by the Kafka-ML Platform training orchestrator, so in case they take longer to train or get temporarily disconnected when they request the model again, they will read the most updated version.

Thanks to the blockchain, every action taken during the FL process (sending model updates, setting up new global models, etc.) is traceable. Anyone can monitor the progress and functioning of training between the Kafka-ML platform and Kafka-ML Federated clients by using blockchain exploring tools such as Blockscout. Therefore, a complete auditability of the training process is guaranteed and every single step of the training process can be validated, strengthening the trustworthiness of the system.

Fig. 3 shows the Blockscout tool where the transactions carried out during the federated training process can be observed.

V. Evaluation: Blockchain-Enhanced FL for Traffic Characterisation of Wireless Technologies

The rapid evolution of wireless technologies needs advanced methods for managing and analysing the massive amounts of data created by connected devices. FL can be used as an approach to address this, allowing the training of models with

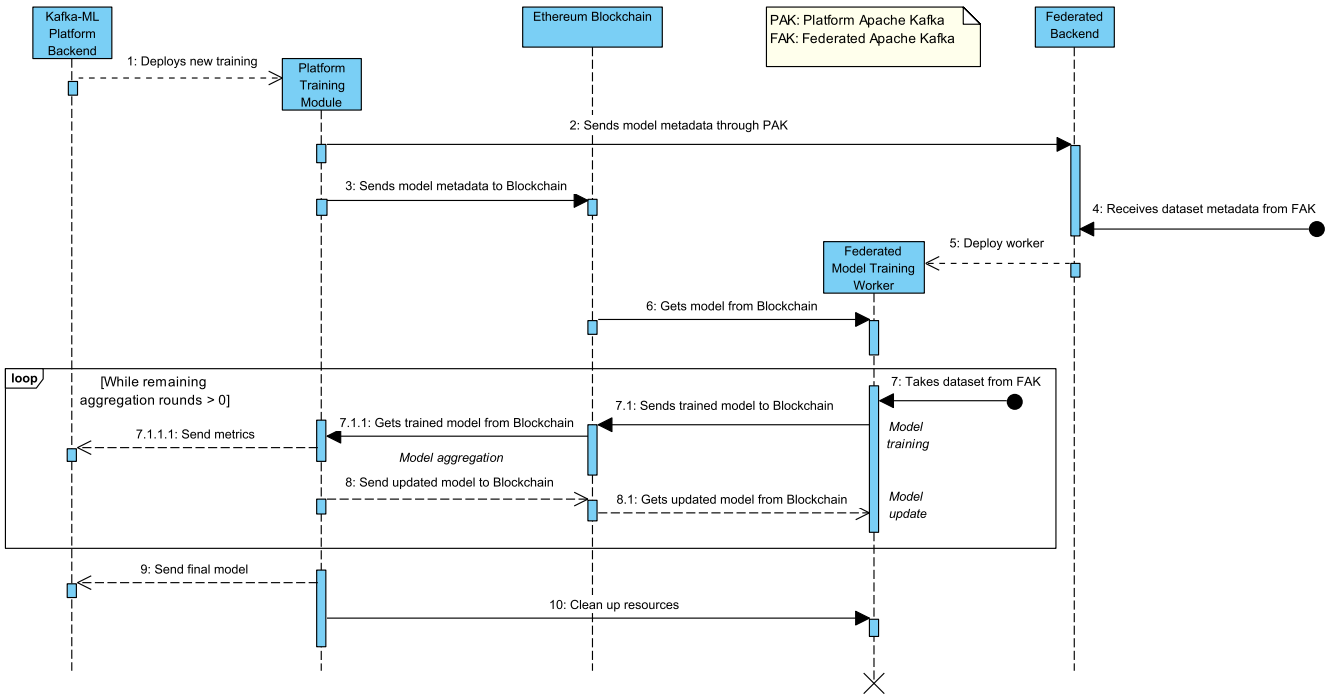


Fig. 2: Federated model training pipeline in Kafka-ML

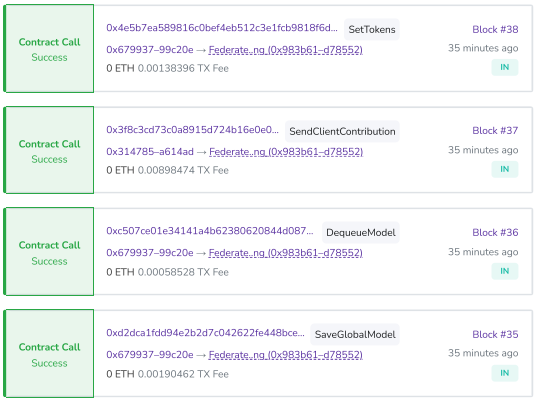


Fig. 3: Blockscout displaying on-chain transactions for Kafka-ML Federated Learning.

data from multiple devices while maintaining data privacy. Also, if blockchain is integrated at the FL process, it can enhance transparency and traceability, ensuring reliable data exchange across devices.

In this use case, the Kafka-ML framework and the blockchain-based FL approach are considered for traffic characterisation of wireless technologies. A comprehensive evaluation of the Kafka-ML blockchain FL feature is presented, using a detailed dataset of IQ samples from various wireless technologies to train a classifier for their recognition.

A. Traffic Characterisation in Wireless Technologies. Dataset Overview

The objective of this use case is to characterise different wireless technologies through a deep learning model within the context of (EVs). This characterisation is crucial for enhancing the performance and integration of EVs within (V2X) ecosystems. In this use case EVs would be the clients collecting data about the different wireless technologies and training a model to identify them, while also collaborating with other EVs by sharing their model and updating a global model at the charging stations (which play the role of central server). Fig. 4 shows a simulated illustration of the use case.

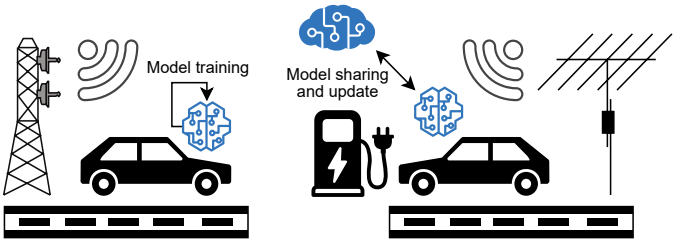


Fig. 4: EVs characterising wireless technologies (left) and updating a global model via charging stations (right).

The dataset utilised in this research has been captured by UGent [20] and includes IQ samples of different wireless communication technologies (LTE, 5G NR, WiFi, ITS-G5, and C-V2X PC5), along with noise samples captured at a sampling rate of 20 Msps.

The LTE dataset was collected using the srsRAN platform and USRP X310 boards in FDD mode at 10 MHz and 5.9

GHz, with traffic loads ranging from 5 to 50 Mbps and MCS indices of 1 to 28. The 5G NR dataset used OpenAirInterface in a 1:1 TDD configuration for NSA mode at 10 MHz and 5.9 GHz, with comparable traffic volumes and MCS indices. WiFi data gathered using the openwifi SDR solution included traffic loads from 10 to 200 packets per second, packet sizes ranging from 500 to 1500 bytes, and MCS indices ranging from 0 to 7. The ITS-G5 and C-V2X PC5 datasets, obtained on the Belgian Smart Highway testbed with the CAMINO framework, featured a variety of packet sizes and intervals, with MCS indices from 0 to 7 for ITS-G5 and 0 to 20 for C-V2X PC5. More detailed about the dataset and the use case can be found at [21].

To evaluate the new Kafka-ML feature using this dataset, the same preprocessing and a similar deep learning model (reduced in size from the original) used in [21] will be employed for comparison of the results. This approach allows us to effectively analyse the model's performance and characteristics under similar conditions, leveraging the established methodologies and findings from the earlier study.

White Gaussian Noise is applied to the IQ samples to replicate different SNR levels, from -10 dB to 30 dB, as part of the preprocessing process. Subsequently, the Fast Fourier Transform (FFT) of the IQ samples is calculated for neural network training and the real-time recognition of wireless technologies.

A convolutional neural network (CNN) with an 880x2 input shape is the model which is being used. Three convolutional blocks are used at first, then flattening and dense layers for classification. The real and imaginary components of the IQ sample FFT, organised as a 2xM matrix, make up the CNN's input.

The input data are processed using the three convolutional blocks to extract features. The first layer has 32 filters of size 1x3, the second layer contains 16 filters of size 2x3, and the third layer contains 8 filters of size 2x3. ReLU activation, batch normalisation, and dropout are the regularisation techniques used in each layer. Max pooling is done after the convolutional layers.

The output of the convolutional layers is flattened and fed through two fully connected layers. The first fully connected layer has 64 neurons and employs ReLU activation, batch normalisation, and dropout. The last layer is a softmax classifier, which allows the input signals to be classified by producing the probabilities for every class.

B. Experimental setup

A highly capable infrastructure is available for conducting a thorough evaluation [22]. The following is a description of the system where the assessment took place:

- **Hardware Configuration**. The deployment of Kafka-ML Cloud has been performed on a computer with an Intel(R) Core(TM) i9-10900K CPU and 64 GB of RAM. The deployment of the multiple Kafka-ML Federated clients has been performed on a Kubernetes cluster with 7 state-of-the-art servers where each server has two Intel(R) Xeon(R) Gold 6230R CPU with two NVIDIA(R) Tesla(R) V100 GPUs as well as 384 GB of RAM.
- **Software configuration** The PC with the Kafka-ML runs Ubuntu 22.04 and K3s v1.28.3. Each node of the cluster runs K3s v1.28.3 on top of Ubuntu 22.04.3 LTS. Every deployment of Kafka-ML Federated has been done in a separated namespace emulating different entities.

C. Training evaluation

The subsequent experiments were prepared to assess the performance of the training. The model was trained on the chosen dataset in several configurations, concretely with 2, 4 and 8 federated clients. Initially, all clients had access to the entire dataset in order to track how the training metrics (validation accuracy, elapsed time and token distribution) vary with the number of clients. Then the dataset was divided in a randomly dispersed way to carry out another test. This should produce a more robust and accurate model than if the model had been trained individually for each customer, even if the classes are slightly out of balance. Every test were carried out with a batch size of 512, 3 epochs, and 32 aggregation rounds and using previous FL implementation available at Kafka-ML and the new blockchain-based implementation.

To assess the effectiveness of these FL approaches, the evaluation considers not only the model's final accuracy but also the cumulative training time and the rewards received by each client. This comprehensive evaluation will reveal how workloads are distributed among clients, ensuring that no client is overworked during the training process. Additionally, analysing the reward distribution will provide insights into each client's contribution to the overall model improvement, helping to identify any clients who may be underperforming or facing challenges during training.

1) Experiment 1: Training evaluation using the complete dataset: Firstly, the test was performed on the full dataset to observe how, regardless of 2 or more customers, they all achieve similar results. The results of the test with the whole dataset can be seen in Fig. 5.

It can be seen that regardless of the number of customers or the implementation used, the results are similar. Something to note is that once the training weights stabilise after several rounds (weights are initialised randomly), higher accuracies are achieved earlier when fewer clients are involved. This is because the same number of aggregation rounds are being maintained for a larger number of clients, causing each client to train fewer rounds, resulting in slower learning. Another detail that is similar is the training time between techniques given the number of clients and this dataset, with the blockchain implementation being slightly slower as transactions have to be confirmed. However, the additional task of interacting with the blockchain does not significantly increase the workload. Fig. 6 shows a graph with a time comparison.

An interesting detail is the way in which the tokens have been distributed. These tokens are distributed in a fixed way such that if the contribution is better than the previous one, a base amount is distributed, decreasing to 0 if it is similar

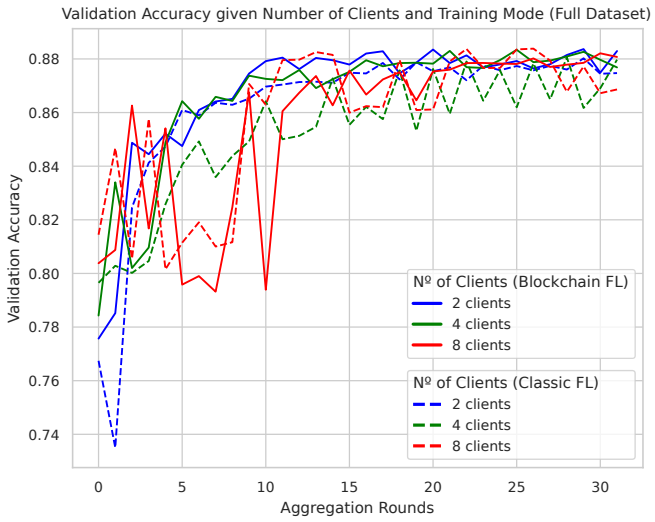


Fig. 5: Validation Accuracy using the complete dataset at all clients

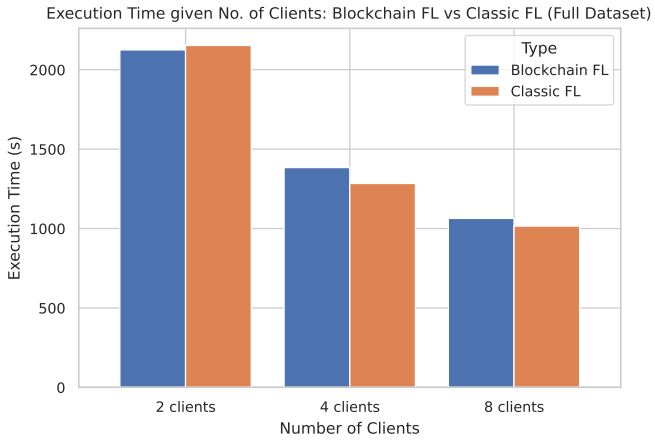


Fig. 6: Elapsed training time using the complete dataset at all clients

given the metrics, slightly worse, or a bad contribution. Fig. 7 shows the distribution of these tokens given the number of clients for this dataset.

As can be seen, there is a slight imbalance in the tokens distributed, which becomes more pronounced as more customers appear. This may be due to the fact that the first clients to finish their training get higher rewards when comparing their metrics at earlier aggregation rounds, which causes subsequent clients to compare their models with a more competitive set of metrics.

2) Experiment 2: Training evaluation with randomly splitting the dataset: In the next experiment, the dataset has been randomly distributed across the different federated clients (with consistent seeding for reproducibility). The results obtained are shown in Fig. 8.

In this case, it becomes slightly clearer that as the number of customers increases, the accuracy decreases slightly. This

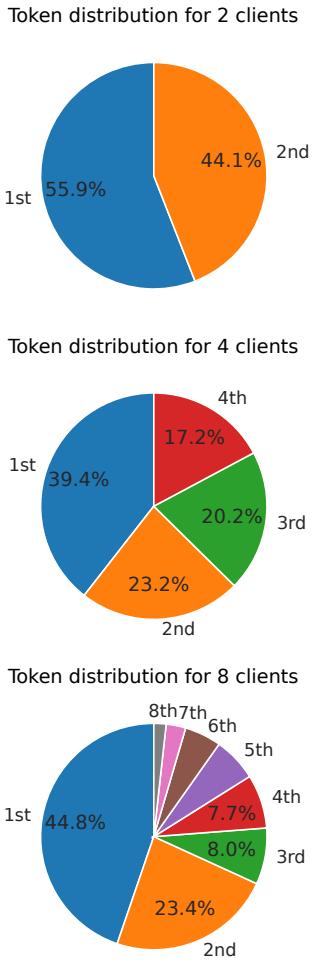


Fig. 7: Distributed tokens using the complete dataset at all clients

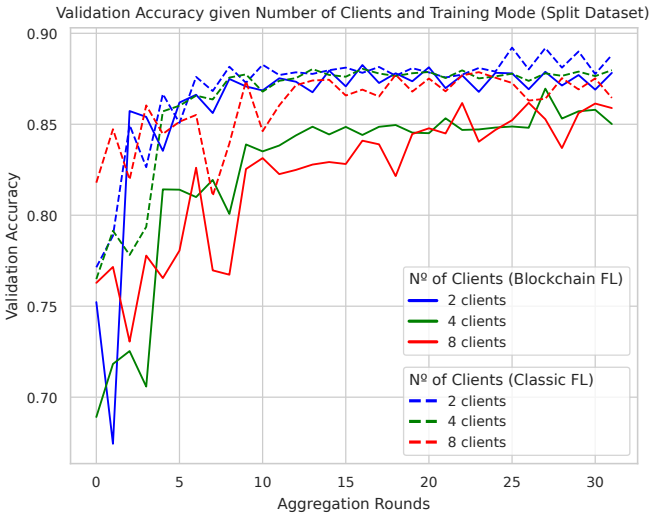


Fig. 8: Validation Accuracy using a randomly distributed dataset split for each client

is due to what was described previously, but as the full dataset is not available now, this is somewhat more accentuated. It is worth noting that the models trained with the new methodology have slightly lower accuracy. Although this should not be a concern since the algorithm is the same, it might be due to the data distribution and/or the model weights randomness at the initialisation stage.

A thing to highlight is the training times in this example, where the new implementation is faster than the old one (Fig. 9). This may be because, as the datasets are now smaller, the training phases are completed earlier, so weights are sent earlier to Kafka, overloading the control topics. This is accentuated in cases with a larger number of clients (it can be fixed by a using a good topic policy, partitioning or replicating the topics as needed). In the new implementation, as there is no dependence on Kafka for control messages, there are no bottlenecks here.

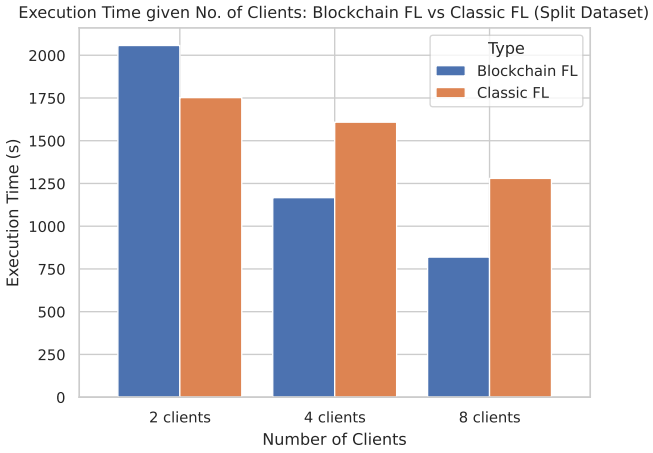


Fig. 9: Elapsed training time using a randomly distributed dataset split for each client

Regarding the token distribution (Fig. 10), something similar is obtained compared to the previous test, with the difference that now larger imbalances are observed with less clients, decreasing this imbalance in comparison to the previous test in the cases of a higher number of clients. This may be because since clients do not have the full dataset, a client who collaborate later may still improve the model noticeably, distributing the rewards slightly better.

VI. Conclusion and Future Work

In this work, traceability and transparency has been integrated through the use of blockchain technologies (and the Ethereum network) in FL trainings at the Kafka-ML platform. Thanks to this new integration, clients and their model contributions can be identified and rewarded, increasing the confidence of those involved in the process. Since data streams are needed as datasets for model training with continual data streams as well as for transferring model weights, this strategy has potential to increase the adoption of Federated Learning in the IoT field.

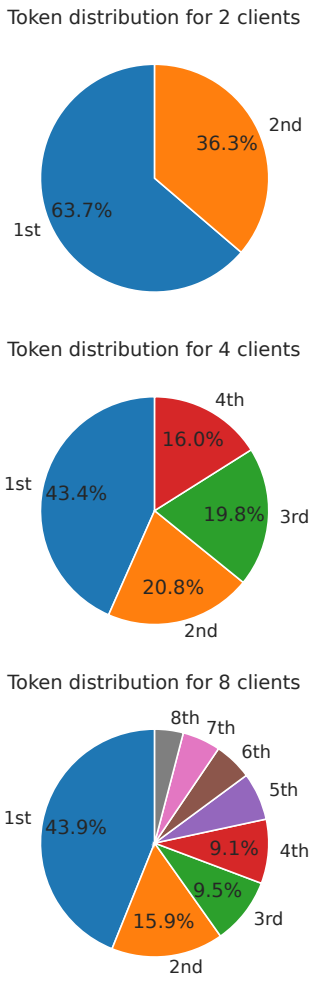


Fig. 10: Distributed tokens using a randomly distributed dataset split for each client

To demonstrate the potential of this integration, an use case for traffic characterisation of wireless technologies in EVs has been presented and a comprehensive evaluation of the current implementation with blockchain compared to the existing implementation has been performed.

As future work, the following improvements to Kafka-ML and Kafka-ML Federated have been identified:

- Integrate new model aggregation techniques. Although FedAvg is a suitable aggregation strategy for the proposed use cases, there are other use cases where the data are not as uniformly distributed, or cases where there are customers are significantly slower and participate fewer times. Currently, these cases are not fully covered, which would lead to skewed models. However, there exist more aggregation techniques such as ASO-Fed, FedDR o AsyncFedED. Integrating some of these techniques into Kafka-ML could enhance its performance and applicability in different contexts.
- Improving reward distribution system. While the concept presented is still a PoC, for further iterations of this in-

tegration, new ways of distributing rewards to customers should be explored.

- Integrate other processing tasks. At times, it is necessary to integrate data processing tasks either before or after the data enters the model. It would be interesting if this processing could be applied directly to the data stream from Kafka-ML.
- Automatic optimisation of hyperparameters. This is another interesting functionality to integrate as it would save time when looking for a fine-grained solution while training ML/AI models.
- Another possible approach: IPFS for model sharing. An alternative for model sharing is the InterPlanetary File System (IPFS). While Kafka-ML uses Apache Kafka for centralised data transfer, IPFS offers decentralised storage, enhancing availability and robustness. Content addressing via cryptographic hashes ensures data integrity, and IPFS provides scalability as more participants join the network.

ACKNOWLEDGMENTS

This work is funded by the Spanish projects: Grant CPP2021-009032 ('ZeroVision: Enabling Zero impact wastewater treatment through Computer Vision and Federated AI') funded by MICIU/AEI/10.13039/5011000110331 and by 'European Union NextGenerationEU/PRTR'. Grant TED2021-130167B-C33 ('SIERRA: Application of Digital Twins to more sustainable irrigated farms') funded by MICIU/AEI/10.13039/5011000110331 and by 'European Union NextGenerationEU/PRTR'. Grant TSI-063000-2021-116 ('5G+TACTILE_2: Digital vertical twins for B5G/6G networks') funded by MICIU/AEI/10.13039/501100011033 and by 'European Union NextGenerationEU/PRTR'. Grant PID2022-141705OB-C21 ('DiTaS: A framework for agnostic compositional and cognitive digital twin services') funded by MICIU/AEI/10.13039/501100011033/ and by 'FEDER'. Grant MIG-20221022 ('GEDERA: Intelligent Flexible Energy Demand Management in Coupled Hybrid Networks'), funded by MICIU/AEI/10.13039/501100011033/ and by 'European Union NextGenerationEU/PRTR'. This project has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement EVOLVE No 101086218.

DATA & CODE AVAILABILITY

The data that support the findings of this study are available at [20]. These data were derived from the paper [21]. Instructions to deploy Kafka-ML and all its code can be found as open-source at Kafka-ML's GitHub Repository (https://github.com/ertis-research/kafka-ml).

REFERENCES

[1] C. Martín, P. Langendoerfer, P. S. Zarrin, M. Díaz, B. Rubio, Kafka-ml: Connecting the data stream with ml/ai frameworks, Future Generation Computer Systems 126 (2022) 15–33.

[2] A. J. Chaves, C. Martín, M. Díaz, Towards flexible data stream collaboration: Federated learning in kafka-ml, Internet of Things 25 (2024) 101036.

[3] X. Tu, K. Zhu, N. C. Luong, D. Niyato, Y. Zhang, J. Li, Incentive mechanisms for federated learning: From economic and game theoretic perspective, IEEE transactions on cognitive communications and networking 8 (3) (2022) 1566–1593.

[4] W. Issa, N. Moustafa, B. Turnbull, N. Sohrabi, Z. Tari, Blockchain-based federated learning for securing internet of things: A comprehensive survey, ACM Computing Surveys 55 (9) (2023) 1–43.

[5] A. J. Chaves, C. Martín, M. Díaz, The orchestration of machine learning frameworks with data streams and gpu acceleration in kafka-ml: A deep-learning performance comparative, Expert Systems 41 (2) (2024) e13287.

[6] A. Carnero, C. Martín, D. R. Torres, D. Garrido, M. Díaz, B. Rubio, Managing and deploying distributed and deep neural models through kafka-ml in the cloud-to-things continuum, IEEE Access 9 (2021) 125478–125495.

[7] A. Carnero, C. Martín, G. Jeon, M. Díaz, Online learning and continuous model upgrading with data streams through the kafka-ml framework, Future Generation Computer Systems (2024).

[8] A. Carnero, O. Waqar, C. Martín, M. Díaz, Distributed federated and incremental learning for electric vehicles model development in kafka-ml, in: 2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM), IEEE, 2024, pp. 1–8.

[9] M. Girmay, M. Seif, V. Maglogiannis, D. Naudts, A. Shahid, H. V. Poor, I. Moerman, Over-the-air aggregation-based federated learning for technology recognition in multi-rat networks, in: 2024 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), IEEE, 2024, pp. 465–472.

[10] C. Xu, Y. Qu, Y. Xiang, L. Gao, Asynchronous federated learning on heterogeneous devices: A survey, Computer Science Review 50 (2023) 100595.

[11] J. Konečnỳ, Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv:1610.05492 (2016).

[12] X. Bao, C. Su, Y. Xiong, W. Huang, Y. Hu, Flchain: A blockchain for auditable federated learning with trust and incentive, in: 2019 5th International Conference on Big Data Computing and Communications (BIGCOM), IEEE, 2019, pp. 151–159.

[13] S. K. Lo, Y. Liu, Q. Lu, C. Wang, X. Xu, H.-Y. Paik, L. Zhu, Toward trustworthy ai: Blockchain-based architecture design for accountability and fairness of federated learning systems, IEEE Internet of Things Journal 10 (4) (2022) 3276–3284.

[14] Z. Peng, J. Xu, X. Chu, S. Gao, Y. Yao, R. Gu, Y. Tang, Vfchain: Enabling verifiable and auditable federated learning via blockchain systems, IEEE Transactions on Network Science and Engineering 9 (1) (2021) 173–186.

[15] R. Ning, C. Wang, X. Li, R. Gazda, H. Wu, Blockfed: A high-performance and trustworthy blockchain-based federated learning framework, in: GLOBECOM 2023-2023 IEEE Global Communications Conference, IEEE, 2023, pp. 892–897.

[16] F. Zhang, Y. Zhang, S. Ji, Z. Han, Secure and decentralized federated learning framework with non-iid data based on blockchain, Heliyon 10 (5) (2024).

[17] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, P. Yu, Bafl: A blockchain-based asynchronous federated learning framework, IEEE Transactions on Computers 71 (5) (2021) 1092–1103.

[18] X. Huang, X. Deng, Q. Chen, J. Zhang, Aflchain: blockchain-enabled asynchronous federated learning in edge computing network, in: 2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring), IEEE, 2023, pp. 1–5.

[19] J. Yun, Y. Lu, X. Liu, Bcafl: A blockchain-based framework for asynchronous federated learning protection, Electronics 12 (20) (2023) 4214.

[20] M. Girmay, A. Shahid, Dataset: Iq samples of lte, 5g nr, wi-fi, its-g5, and c-v2x pc5 (2023). doi:10.21227/72qq-z464.
URL https://dx.doi.org/10.21227/72qq-z464

[21] M. Girmay, V. Maglogiannis, D. Naudts, M. Aslam, A. Shahid, I. Moerman, Technology recognition and traffic characterization for wireless technologies in its band, Vehicular Communications 39 (2023) 100563.

[22] I. Software, Ertis research fog computing cluster, accessed: 2024-09-23 (2024).
URL https://itis.uma.es/en/5437-2/