Future Generation Computer Systems 152 (2024) 17–29

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Enabling Federated Learning at the Edge through the IOTA Tangle

Carlo Mazzocca [*], Nicolò Romandini, Rebecca Montanari, Paolo Bellavista

Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

ARTICLE INFO

Keywords:
Federated Learning
Edge computing
Blockchain
DAG
Tangle
IOTA

ABSTRACT

The proliferation of Internet of Things (IoT) devices, generating massive amounts of heterogeneous distributed data, has pushed toward edge cloud computing as a promising paradigm to bring cloud capabilities closer to data sources. In many cases of practical interest, centralized Machine Learning (ML) approaches can hardly be employed due to high communication costs, low reliability, legal restrictions, and scalability issues. Therefore, Federated Learning (FL) is emerging as a promising distributed ML approach that enables models to be trained on remote devices using their local data. However, "traditional" FL solutions still present open technical challenges, such as single points of failure and lack of trustworthiness among participants. To address these open challenges, some researchers have started to propose leveraging blockchain technologies. However, the adoption of blockchain for FL at the edge is limited by several factors nowadays, such as long waiting times for transaction confirmation and high energy consumption.

In this work, we conduct an original and comprehensive analysis of the key design challenges to address towards an efficient implementation of FL at the edge, and analyze how Distributed Ledger Technologies (DLTs) can be employed to overcome them. Then, we present a novel architecture that enables FL at the edge by leveraging the IOTA Tangle, a next-generation DLT whose data structure is a Directed Acyclic Graph (DAG), and the InterPlanetary File System (IPFS) to store and share partial models. Experimental results demonstrate the feasibility and efficiency of our proposed solution in real-world deployment scenarios.

1. Introduction

Traditional cloud-based infrastructures cannot manage the massive, heterogeneous, and distributed data generated by billions of Internet of Things (IoT) devices spread all over the world. Edge cloud computing has been proposed as a promising paradigm to bring cloud capabilities closer to data sources, thus reducing latency and requested bandwidth, while improving the resilience and availability of the network [1]. Due to its ability to process the proliferation of IoT data and leverage heterogeneous computing resources, the integration of Machine Learning (ML) and IoT at the edge is a rapidly growing research field.

Traditionally, data storage and model training are performed on resource-rich cloud servers. However, it is becoming widely recognized that transmitting all data collected from edge devices to a geographically distant data center where model training is entirely performed is not feasible. Centralized ML approaches are hindered by high communication costs, low reliability, and legal restrictions In this context, Federated Learning (FL) [2] is a promising distributed ML approach: models are directly trained (at least partly) on remote devices using their local data; a central server, usually referred to as the parameter server, is used to periodically collect partial models from clients and to aggregate them into a new global one by exploiting a predefined strategy. The global model is then returned to the edge devices for the next round of training.

Despite the numerous benefits of FL, using a central parameter server still raises some non-negligible challenges [3], including the lack of trust among unknown participants and the risk of malicious clients intentionally submitting incorrect models. Additionally, a central server is vulnerable to single points of failure, low scalability, and may lead to the possibility of bias in the training process since it may favor a model over the others. These issues are particularly relevant in edge computing scenarios where a large number of participants, who are typically unknown to each other, are involved.

To address these challenges, researchers have recently started to propose to leverage the immutable and transparent nature of the blockchain to build more robust FL architectures [4]. On the one hand, traditional blockchains that rely on third parties (i.e., miners and validators) to validate blocks have the potential to improve the security of FL approaches. For example, through a consensus protocol, they could assure the correctness of a global model, which will be generated without bias. On the other hand, their long waiting time for transaction confirmation and their high energy consumption are among

* Corresponding author.
E-mail address: carlo.mazzocca@unibo.it (C. Mazzocca).

https://doi.org/10.1016/j.future.2023.10.014
Received 31 March 2023; Received in revised form 24 July 2023; Accepted 20 October 2023
Available online 25 October 2023

the most relevant factors that limit their adoption in edge deployment environments, which still remain largely unexplored.

We believe that Directed Acyclic Graph (DAG)-oriented Distributed Ledger Technologies (DLTs), indicated as DAG for briefness, have the potential to guarantee the needed security features as the blockchain, while providing improved performance in terms of latency and energy consumption, thus well meeting the requirements of IoT and edge computing. Therefore, in this work, we first conduct a comprehensive analysis of the main factors to keep into careful account while designing and developing FL at the edge for IoT application scenarios. In addition, we propose a novel architecture called FL at the Edge through the IOTA TAngle (FETA), which leverages the IOTA Tangle [5], a next-generation DAG engineered for the IoT, and the InterPlanetary File System (IPFS), which is commonly used in the literature [6] to overcome scalability issues related to the size of data shared in DLT environments. The reported experimental results show that our original FETA proposal is feasible and efficient in real-world deployment scenarios of practical interest. More in detail, we believe that this paper contributes to the advancement of the state-of-the-art in the field because:

- we review the main factors to consider when efficiently designing and implementing FL for edge-enabled IoT scenarios;
- we originally analyze how DLTs can be employed to address the related design challenges;
- we propose the novel FETA solution that enables FL at the edge by leveraging IOTA and IPFS.

The remainder of the paper is structured as follows. Section 2 highlights the key design challenges to address for FL support at the edge. Then, Section 3 delves into how DAG can be used to overcome them. In Section 4, we present our original FETA proposal, by outlining its key components and the key guidelines for its efficient implementation at the edge. Section 7 reviews the existing literature on the use of blockchain and DAG for implementing FL, while Section 8 ends the paper with conclusive remarks.

2. Federated Learning at the Edge

FL is a decentralized ML technique that decouples the process of training a model from directly accessing raw data. It enables collaboration among different data source owners with similar goals while addressing privacy concerns. Instead of transferring raw data to a centralized cloud data center, FL allows model training to be performed directly on end devices. During each FL round, participating clients train a local model using their on-premises data and then send such a partial model to a server for aggregation. The ML models are trained directly on remote clients, such as edge nodes and devices, while the server is only responsible for combining the updates from all clients. FL and edge computing are key technologies that allow ML approaches to process the vast amount of data generated from various distributed locations. However, deploying model training on mobile edge networks, closer to where data is produced, introduces many challenges that require to be carefully addressed [7].

In this section, we aim to highlight the novel considerations and challenges in designing and developing FL solutions at the edge. We review and discuss the main design choices that must be carefully addressed to ensure efficient and effective FL operations in such a context.

2.1. Scalability

FL typically involves a large number of mobile devices that are geographically distributed and have limited connectivity [8]. With a high number of clients participating in model training, several technical challenges arise. For instance, participants may lose connection to the server (i.e., device dropout) due to weak signals or low battery levels. Another FL challenge relates to the proliferation of data generated by participating devices, which makes it difficult to employ traditional methods based on centralized servers. A single server may experience low communication latency and may not be able to aggregate all the updates offloaded from millions of devices.

2.2. Portability

Portability is crucial for enabling FL at the edge. The ability to run ML models on a wide range of devices, such as smartphones, laptops, and IoT devices, is essential for the widespread adoption of FL. The varying processing power, memory, and storage capacity of these devices necessitate portable and adaptable FL algorithms. Ensuring that FL algorithms can be executed on heterogeneous devices paves the way for the training of models on large datasets collected from diverse sources. This is especially important for edge computing, where devices may have limited resources and battery life.

Due to the above considerations, it essential to keep operating as lightweight as possible. In this direction, containerization is envisioned as a valuable approach to simplify the deployment and management of FL models on edge devices with heterogeneous hardware and software configurations [9]. Containerization involves packaging the model and its dependencies into a single unit that can be consistently executed on different devices, eliminating compatibility concerns. Moreover, containers provide a secure execution environment for ML models, which is particularly relevant since these models often handle sensitive data, exposing end-users to potential data breaches. Finally, containerization facilitates the monitoring and updating of the models deployed on edge devices, simplifying maintenance and improvement over time. Although technology is not mature yet, the industry has been encouraging progress in this direction. WeBank's KubeFate,[1] TensorFlow Federated,[2] PySyft,[3] and PaddleFL[4] are examples of framework that enables executing FL tasks across multiple containers.

2.3. Security and privacy

Security and privacy are critical concerns while implementing FL at the edge. On the one hand, malicious participants of FL may attempt to disrupt collaborative training. On the other hand, honest nodes may not fully trust the server, which could have biases to prefer a model over the others. In addition, curios nodes may seek to extract personal data from ongoing updates.

2.3.1. Security

As participants are highly distributed and unknown to each other, some may act maliciously, such as by using Byzantine attacks (e.g., Gaussian attack, Omniscient attack, and Flip bit attack) [10] to intentionally upload incorrect weights and leading the whole training process to an incorrect phase. Furthermore, malicious nodes may inject attacks into the FL process by leveraging poisoning attacks: data poisoning and model poisoning. The former affects the training by modifying the input data, while the latter crafts local models to inject backdoors into the global model.

To address these security issues, it is necessary to differentiate honest from malicious participants. Defense mechanisms involve scoring each uploaded weight and aggregating the ones with the highest scores, using the geometric median and its modification, and removing outliers based on the Euclidean distance.

[1] https://github.com/FederatedAI/FATE
[2] https://github.com/tensorflow/federated
[3] https://github.com/OpenMined/PySyft
[4] https://github.com/PaddlePaddle/PaddleFL

2.3.2. Privacy

FL was proposed to address the issues related to data privacy, ownership, and legalization. Local updates are computed by participants using their private and local data. However, the sharing of models, parameters, and global models among clients exposes FL at the edge to several risks and vulnerabilities [11]. Curios servers or nodes may attempt to infer information about the dataset from uploaded weights or infer knowledge from the aggregated weights. Privacy attacks in FL can be classified into membership inference (determining if a data record belongs to a node's training dataset) or data inference (collecting training data or a class using the information provided by nodes).

Given the huge amount of sensitive data involved and the aforementioned threats, preserving privacy is a primary concern in FL. In this direction, common techniques include differential privacy [12] (adding noise while maintaining a certain level of utility) and encryption-based solutions (e.g., secure multi-party computation).

2.4. Authentication and authorization

Clients need to interact with each other transparently and securely. However, the distributed nature of edge computing can make it challenging to adopt centralized identities to identify participants and regulate their participation in FL training. With centralized approaches, clients' data is owned and controlled by a central entity, which can share with other services without clients' awareness or consent. This can be especially concerning when sensitive information is stored in a single server, as it increases the risk of data leakage.

To address these concerns, FL at the edge requires decentralized authentication and authorization mechanisms. Decentralized identities are only under the control of the data owner, who can decide with whom to share their information. This provides greater security and control over sensitive data, as clients can ensure that their information is only shared with trusted parties.

2.5. Synchronous and asynchronous communication

The generation of the final global model involves several communication rounds. Thus, the communication method, either *synchronous* or *asynchronous*, is a crucial design choice, especially when updating models across heterogeneous networks. Most existing solutions (e.g., FedAVG [13] and FedSGD [14]) use synchronous communication, which is relatively simple but can lead to stragglers among devices. The assumptions made by such algorithms are not realistic due to the heterogeneous capabilities of edge devices. Therefore, asynchronous training has been envisioned as a valuable approach to enhance efficiency in FL at the edge.

In asynchronous training, the parameter server does not need to wait for all participants to complete their local training before computing the global model. As a result, each node can take its time to train on its private local data, even with limited computational resources and network delay [9]. However, to avoid loss of updates that may arrive late, partial models can be combined through temporarily weighted aggregation [15].

2.6. Resource allocation

Devices heterogeneity remarkably affects the speed of the training resulting and has led to novel solutions referring to levels [9]. In this section, we analyze the main directions that contribute to boosting the training speed of an FL process.

2.6.1. Participant selection

In traditional FL, participants in the training are randomly selected to perform computation. However, this random selection may lead to imbalances in resource availability among the participating nodes. Some nodes may have sufficient resources, while others may have limited capabilities. As a result, the slowest participant the total training time for that iteration. To address such a concern, smarter strategies should be employed for node selection.

FedCS [16] is an example of a protocol that selects clients based on their resource conditions. By leveraging the resource information, FedCS determines which clients are capable of participating in a specific training round and meeting necessary requirements (i.e., uploading model updates within a designated deadline). Similarly, Hybrid-FL [17] requests resource information and a small portion of clients upload their data, addressing non-IID concerns and improving performance. Reputation is another important factor to consider when selecting clients in FL. Since participants are often unknown to each other, previous performance can be used to determine which participants to include [18,19]. Furthermore, by selecting participants based on their performance in previous rounds, it is possible to remove malicious or unhelpful contributions and enhance the overall reliability of the FL system.

2.6.2. Resource optimization

Resource allocation approaches in FL involve transforming the challenge into an optimization problem. The objective is to find the most efficient way to implement FL at the edge while considering constraints such as the computation resources and network limitations of edge nodes. There are two broad categories of resource optimization approaches: black-box and white-box approaches [20]. Black box techniques focus on arranging network entities and system resources based on the observation of the external properties of the involved ML learning models. The following are the main examples of black box techniques:

- Training Tricks: empirical hyperparameter tuning to minimize the number of communication rounds [14] and data augmentation for improved inference accuracy.
- Data Compensation: sharing a small subset of local data from each client globally to alleviate non-IID challenges and potentially boost the model accuracy [21].
- Hierarchical Aggregation: implemented through a client-edge-cloud multi-stage procedure [22], where early model aggregations at the network edge have lower communication cost and can efficiently alleviate the uncertainty of model updates due to the randomness of local data.

In contrast, white-box approaches try to understand the internal structure as well as module functionality of ML learning models. Some common white-box techniques comprise:

- Model Compression: reducing the model size using techniques such as quantization and low-rank approximation [23], making it suitable for resource-constrained environments.
- Knowledge Distillation: using a portion of the activation of a well-trained ML model as an additional regularizer to teach a new model [24].
- Feature Fusion: merging global and local feature extractors and passing the result to the classifier for loss evaluation [25]. This helps balance non-IID data and reduces communication rounds.
- Asynchronous Training: the parameter server does not need to wait for all participants to complete their local training before computing the global model. Each node can train on its private local data at its own pace, even with limited computational resources and network delay.

2.7. Incentive mechanisms

FL faces the challenge of ensuring active and reliable participation from clients. To address this challenge, it is necessary to introduce incentive mechanisms. Incentives can be categorized as positive or negative. Positive incentives offer rewards to motivate participants, while negative incentives seek to deter harmful behaviors by punishing individuals [26].

The rewards often depend on the quality of data used for model training. So it is crucial to assess the right value of the data. Shapley's value is a commonly employed concept in the literature to determine the value of data [27]. Additionally, since the size of training samples is connected to the accuracy of the learning model, the amount of training data is frequently used as an indicator of the contribution. Another way to measure contribution is to calculate each client its reputation. Clients with high reputations are more likely to send reliable and high-quality models. Reputation-based systems can prevent attacks on the FL process by disqualifying clients with low reputations [28]. Incentive mechanisms can also be driven by clients' resource allocation. For example, the budget can be distributed among clients to motivate them to contribute their CPU power [29].

2.8. Energy

Energy consumption is a crucial factor to consider in FL at the edge, as many edge devices have limited energy resources and cannot afford to waste energy on computationally intensive tasks [30]. The energy consumption in FL is influenced by both the frequency of communication between devices and the processing of data on the devices [7]. Communication of model updates between participants and the parameter server can be computationally expensive and energy-consuming. To mitigate this, techniques such as model compression, quantization, and pruning can be used to reduce the size of the models and the amount of data that needs to be transmitted. Another key factor to consider is the processing of data on the devices. In FL at the edge, devices often need to perform computationally intensive tasks such as model training, which can consume significant energy. Techniques to reduce energy consumption include reducing the number of training iterations, using lighter models, or using energy-efficient hardware.

2.9. Hardware requirements

ML algorithms can address a wide range of tasks (e.g., image classification and audio recognition) that require substantial computational and storage resources. However, edge devices are often limited by their available resources, memory footprints, and power consumption. When it comes to implementing FL at the edge, the required hardware depends on several factors, such as the size and complexity of the models being trained, the amount of processed data, and the computational resources available on each edge device. Thus, it is recommended to carefully assess the specific needs of the use case scenario before choosing the hardware. Abreha et al. [11] investigated the edge computing hardware requirements for implementing FL. The results are reported in Table 1.

Generally, edge devices for FL should have enough computational power that can support the training of partial models, store and communicate model updates. In addition, the hardware should provide reliable network connectivity to enable efficient and low-latency communication among the participants of the FL training. Finally, edge devices should be designed to be energy-efficient, as they may be running for extended periods and have limited battery life.

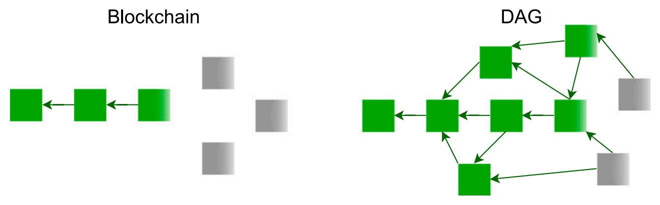

Fig. 1. Blockchain and DAG data structure.

3. Integrating DLTs and FL

DLTs are often employed to enhance the robustness of FL platforms. While some existing works [4,31] analyze how blockchain can fulfill the design requirements of FL in edge computing, they often overlook alternative DLTs such as DAG and fail to provide a comprehensive analysis of all the essential design factors for FL at the edge. Recently, Ko et al. [32] identified design aspects for asynchronous FL with DAG in edge computing. However, similar to other works, they do not thoroughly explore additional factors like energy consumption and hardware requirements. Therefore, in this section, we aim to present a detailed analysis of how blockchain and a specific DAG implementation, the IOTA Tangle, can address the challenges outlined in the previous section.

3.1. Blockchain and DAG

Blockchain is a widely recognized type of distributed ledger characterized by a series of blocks, connected through cryptographic hashes. Each block contains a reference to the previous block's hash, creating a secure chain. Tampering with any block would result in a change in the hash, and thus, can be easily detected. The decentralized nature of blockchain is maintained through a peer-to-peer network, providing enhanced fault tolerance and resilience against cyber-attacks. The network uses consensus mechanisms, such as proof-of-work (PoW) or proof-of-stake (PoS), to ensure secure synchronization and discourage malicious behavior. The choice of consensus protocol can significantly impact energy and power consumption, which are crucial in edge computing environments, where devices are often resource-constrained and battery-powered. In addition, such consensus protocols (in particular PoW) require a long confirmation time that obstacles asynchronous communications, which should be preferred due to the huge number of heterogeneous devices.

A DAG is a novel type of DLT that features, as illustrated in Fig. 1, a distinct data structure compared to the traditional blockchain. On the other hand, in a DAG transactions are connected in a non-linear, tree-like structure. DAG shows unique properties that make it a promising alternative to overcome the well-known limitations of blockchain. The key advantage of this structure is that it enables parallel processing of multiple transactions, resulting in increased throughput and faster confirmation times. This makes DAG-based ledgers particularly well-suited for applications that require real-time transactions and low latency. Another significant difference between DAGs and blockchains lies in their consensus protocols. Contrary to blockchains that rely on mechanisms based on blocks confirming transactions performed by third parties, DAG-based ledgers confirm transactions through a protocol in which transactions confirm each other. Beyond enhancing throughput, the lack of middlemen in the validation process also enables feeless transactions that are fundamental in edge environments where the number of transactions remarkably scales up with the number of involved devices.

Table 1
Comparison of hardware accelerators for implementing FL at the edge.

Name	Owner	Pros	Cons
CPU/GPU	NVIDIA and Radeon	High memory, bandwidth, and throughput	Consumes a large amount of power
FPGA	Intel	High power efficiency with optimized performance per watt of consumption, cost-effective solution for large-scale operations, ideal for battery-powered devices, and well-suited for large applications on cloud servers.	It demands substantial storage capacity, an ample amount of external memory and bandwidth, and requires computational resources with billions of operations per second
ASIC	Intel	Reduces memory transfer to a minimum, boasts the highest energy efficiency compared to FPGAs and GPUs, and boasts the fastest computational speed in comparison to FPGAs and GPUs	Prolonged development cycle and inflexibility to accommodate diverse deep learning network designs

3.2. Scalability and portability

Blockchain and Tangle enable addressing single points of failure and improve the scalability of ML centralized approaches. However, blockchain employs consensus protocols that require a significant amount of time to validate transactions (i.e., model updates). Contrary, the Tangle adopts a lightweight probabilistic consensus protocol that enables parallel validation of transactions without requiring total ordering. Therefore, it is more adequate for scenarios where there are several transactions to process, which is the case of FL at the edge.

3.3. Security and privacy

The integration of DLTs and FL has the potential to address many of the security and privacy challenges, enabling the building of more secure platforms. One of the key advantages of DLTs is that they enable decentralized control of the collaboration process in FL. Each party has more control over its data, which reduces the risk of a single entity having complete control of the whole process. With DLTs, participants can work together to train an ML model without sacrificing their data privacy or security. Moreover, the transparent and immutable nature of DLTs, combined with smart contracts, can significantly boost the level of trustworthiness among participants. The immutability of records ensures that any changes to data and models can be tracked and audited, making it difficult for any party to maliciously alter the data or model. Smart contracts automate the collaboration process by enforcing specific rules and conditions that guarantee the correctness of the generated model.

However, it is important to note that using DLTs to share partial models still exposes the privacy of participants, which demands further privacy-preserving techniques. Therefore, advanced encryption and other privacy-preserving technologies should be used to ensure that the privacy of participants is maintained while also enabling effective collaboration.

3.4. Authentication and authorization

FL requires a way to regulate participation that is both secure and transparent. Decentralized identifiers (DIDs) and Verifiable Credentials (VCs) have emerged as promising solutions for this challenge [33]. DIDs are a new type of identifier that enable verifiable, decentralized digital identity [34]. Unlike traditional identifiers, which rely on a central authority to validate identity, DIDs use public-key cryptography to establish trust between parties. With DIDs, individuals can control their own identity and share only the information they choose, providing a more secure and privacy-preserving approach to authentication and authorization. In FL, DIDs can be used to authenticate participants and ensure that only authorized entities are able to access and participate in the FL process. This helps to ensure that the data being used in the FL process is coming from trusted sources and is not being manipulated or corrupted. On the other hand, VCs are claims made by an issuer that state something about a subject [35]. These claims can be cryptographically signed and shared with third parties, allowing for easy verification of the subject's credentials. VCs are also used to protect the privacy of participants by allowing them to share only the minimum amount of personal information needed for the FL process. Together, DIDs and VCs enable a claim-based identity system, which provides a more flexible and adaptable approach to authentication and authorization.

By using DLTs, DIDs, and VCs can be stored and managed in a decentralized and trustless manner, providing a more secure and transparent way to manage identity and credentials [36]. For example, blockchain can be used to store DIDs and associated VCs in a decentralized and tamper-proof manner. This can help to ensure that only authorized parties can access and verify the credentials of FL participants. Additionally, by leveraging smart contracts, blockchain can enable automatic verification of VCs, eliminating the need for manual verification by a central authority.

3.5. Synchronous and asynchronous communication

Efficient communication is crucial in FL training involving multiple participants. Blockchain technologies rely on a middleman to verify transactions, resulting in long verification and confirmation times, which can take several minutes, that hinder the adoption of asynchronous approaches that are more adequate IOTA, on the other hand, offers feeless transactions that can be immediately attached to the Tangle and verified within seconds. In addition, zero-value transactions can also be used, allowing FL platforms to share information without the added management concerns related to cryptocurrencies. This makes IOTA an ideal technology for implementing efficient communication in FL platforms. Furthermore, the use of IOTA can help reduce communication overhead and costs, enabling FL platforms to operate more efficiently and cost-effectively.

3.6. Resource allocation

Participant selection strategies can easily be adopted when using DLTs. For example, DLTs can be used to account for the reputation of involved nodes or track previous performance. However, contrary to the blockchain, IOTA enables indexing transactions, reducing the time needed to retrieve them and improving the whole efficiency of the FL process. Such a property is beneficial to all the operations that need to share knowledge on the tangle, involving model updates. Concerning resource optimization, most of the existing approaches can be used independently of the technology adopted for aggregating partial models.

3.7. Incentive mechanisms

FL platforms based on blockchain or Tangle can leverage cryptocurrencies and tokens to implement incentive mechanisms that reward clients economically. These incentives can be tied to the amount or quality of resources provided by the clients, such as the data or training services they contribute to the FL network. The use of cryptocurrencies and tokens can enable a fair and transparent system for incentivizing the clients, as their contributions can be easily tracked and quantified. Furthermore, these incentives can attract more participants to the FL network and improve the overall performance of the model. However, it is crucial to design the incentive mechanism carefully to ensure that it aligns with the objectives of the FL network and avoids potential issues such as unfairness or manipulation.

3.8. Energy and hardware requirements

The widespread of blockchain is inhibited by its energy consumption, which is one of the significant concerns also due to the current discussions on climate change and sustainability [37]. Deploying blockchain networks on edge nodes may be unfeasible due to the limited resources. For example, PoW consensus requires a large extent of electrical energy, and the miners compete to validate blocks, resulting in a huge waste of electricity [38]. In contrast, the IOTA tangle is lightweight and energy-efficient, with experiments showing successful deployment on Raspberry Pi 3 and 4, often employed in edge computing environments, with very low energy consumption, ranging from 2 J to 6 J approximately [39].

4. FETA: Federated Learning at the Edge through the IOTA Tangle

In this section, we present FETA[5] a decentralized framework that leverages the IOTA Tangle to enable FL at the edge while addressing the challenges discussed in Section 2. The proposed solution offers both scalability and portability. The Tangle, which is the underlying structure of the network, enables fast and feeless transactions on a large scale. Portability is ensured by allowing each component to execute within a container. This enables the deployment and management of FL models on edge devices independently from the underlying hardware and software configurations, by providing flexibility and ease of use. Furthermore, the Tangle guarantees the same features as the blockchain, such as decentralized control and immutability. However, partial models are not directly shared on it due to their large size, which makes it hard to embed them within transactions. To tackle this problem, they are shared through the decentralized storage of IPFS, while we propose to store only their Content IDentifier (CID) on the Tangle.

In addition, authentication and authorization are implemented through the IOTA Identity framework,[6] which offers DID and VC functionalities. Asynchronous communication, which is fundamental in edge scenarios, is enabled by the consensus protocol: this allows parallel transaction processing in FETA and speeds up the sharing of partial models. Partial models are shared through zero-value transactions that do not require any transfer of value from one entity to another, thus not requiring to provide cryptocurrencies or tokens to FL participants. Although resource allocation algorithms and incentive mechanisms are out of the scope of our FETA proposal, at least at this stage of evolution, they can be easily integrated. Finally, as discussed above, the proposed architecture can be effectively deployed on real-edge devices.

[5] https://github.com/MMw-Unibo/FETA
[6] https://github.com/iotaledger/identity.rs

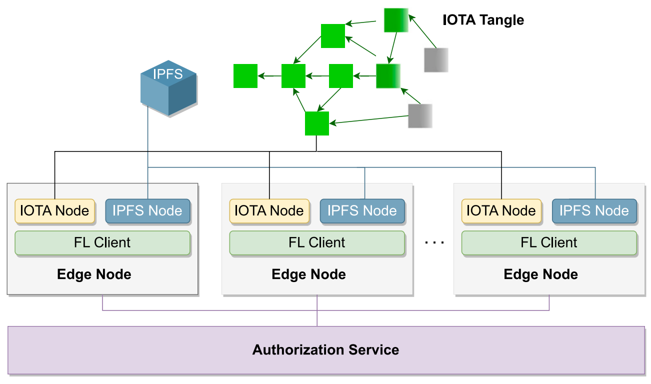

Fig. 2. FETA architecture.

4.1. The FETA architecture

As shown in Fig. 2, to enable FL at the edge our proposed solution consists of multiple edge nodes that receive data from IoT devices and train local models. The edge nodes host the following components:

- FL Client: it trains a local model using data received from IoT devices.
- IPFS Node: it is responsible for storing and distributing the updated models across the network.
- IOTA Node: it belongs to the IOTA Network and cooperates with other nodes to maintain the unified view of the Tangle. It shares references of partial models among participants, by enabling their retrieval via IPFS nodes.

In addition to these components, our architecture includes an Authorization Service (AS) that generates VCs for clients willing to participate in the FL process. Only partial models with valid VC will be included in the aggregation process. The AS is managed by the FETA participants that can establish whether participants are allowed to join.

4.1.1. Authorization service

The AS is a critical component responsible for regulating participation in an FL process. Its primary function is to issue VCs that validate the eligibility of FL clients to contribute to FL training. To ensure the authenticity and prevent VC tampering of the VCs, they are signed by the AS using the private key associated with its DID. As DIDs are public, FL clients can automatically retrieve the corresponding public key of the AS and verify whether a partial model has been published by an authorized participant or not. Therefore, this approach allows AS to be involved only in the initial issuance of VCs. Subsequent verification operations can be performed by the participating clients themselves, promoting decentralization and improving scalability.

4.1.2. FL client

The FL clients receive data from IoT devices to locally train their models, by using any desired ML algorithm. To contribute to FL training, each client must first obtain a valid VC from the AS that allows participating in FL training. The client signs the VC using the private key associated with its DID, obtaining a VP that proves its eligibility. Before including a partial model in the aggregation, each client also verifies that the model has been published by an authorized participant.

4.1.3. IOTA node

An IOTA network consists of multiple IOTA nodes, which store a copy of the Tangle: these nodes are vital to ensuring the integrity and reliability of the IOTA network by participating in the consensus process and validating transactions before adding them to the Tangle. Besides, IOTA nodes can perform other functions, such as acting as

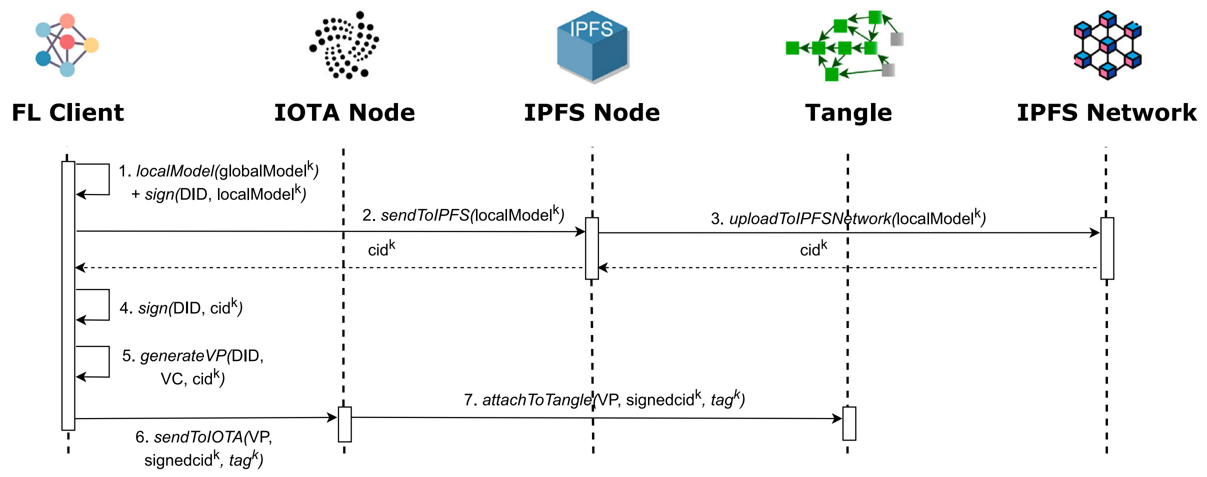
FL Client
IOTA Node
IPFS Node
Tangle
IPFS Network
IPFS
1. localModel(globalModel^k) + sign(DID, localModel^k)
2. sendToIPFS(localModel^k)
3. uploadToIPFSNetwork(localModel^k)
cid^k
cid^k
4. sign(DID, cid^k)
5. generateVP(DID, VC, cid^k)
7. attachToTangle(VP, signedcid^k, tag^k)
6. sendToIOTA(VP, signedcid^k, tag^k)

Fig. 3. Publishing partial models.

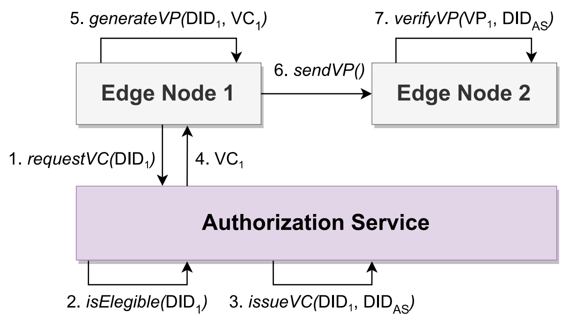

Fig. 4. Authorization workflow.

gateways for communication between users and the IOTA network, participating in transaction routing, and providing access to the distributed ledger for applications and other network participants.

In our architecture, each edge node includes an IOTA node that attaches the partial models provided by the FL client and retrieves other participants' contributions, which are used to generate the global model. This approach ensures that the FL training process is transparent and secure, as all participants' contributions are recorded on the Tangle, which is immutable and tamper-proof. Additionally, the IOTA node capabilities contribute to the overall FL scalability and reliability, by allowing for a large number of participants to contribute to model training simultaneously.

4.1.4. *IPFS node*

IPFS participants, or simply IPFS nodes, are programs that can be executed on a local computer to store files and establish connections with the IPFS network. In IPFS, files are stored and shared using a peer-to-peer (P2P) network. This means that rather than depending on centralized servers to store and retrieve files, IPFS nodes communicate with each other directly to transfer files. In FETA, each edge node includes an IPFS node that is responsible for storing the partial models. FETA exploits efficient integration with IPFS because transactions on IOTA have space limitations, i.e., not allowing the direct publication of ML models in it. Models published on IPFS are signed to ensure the origin of the data can be verified.

4.2. *FETA protocol*

This subsection describes the authorization and FL training stages, which are the main phases that compose the FETA protocol.

4.2.1. *Authorization*

The authorization phase ensures that only authorized FL clients can contribute to the FL training process. In the following, we detail the authorization workflow depicted in Fig. 4:

1. Each FL client requests a valid VC to the AS, providing its DID.
2. The AS verifies the eligibility of the FL client associated with the provided DID.
3. Upon successful verification, the AS issues a valid VC associated with the client's DID and signed through its private key.
4. The FL client receives the issued VC from the AS.
5. The FL client generates a VP by signing the collected VC with its own private key, thus proving its eligibility.
6. The FL client attaches the VP to the transaction that contains the necessary information to retrieve its partial model.
7. Before downloading partial models submitted by other participants, each FL client verifies the validity of the VP using the public keys associated with the AS and other FL clients, which are linked to their respective DIDs.

4.2.2. *FL training*

The FL training phase consists of two sub-phases: *publishing partial models* and *aggregating partial models*. Each sub-phase contributes to the collaborative training process in FL.

Publishing Partial Models. After the issuance of VCs, FL clients can train their models using local data and share the trained partial models with other participants. The protocol for publishing partial models, as depicted in Fig. 3, is described below:

1. Each FL client trains a partial model using its on-premises data. Once the training is complete, the hash of the local model is signed with the private key associated with its DID.
2. The signed hash, along with the model, is sent to the IPFS node, which forwards the data to the IPFS network, which returns a CID to the FL client.
3. The FL client signs the received CID, ensuring the integrity of the model.
4. Using the VC and its associated DID, the FL client generates a VP that serves as proof of eligibility for FL training.
5. Finally, the VP and the signed CID are bundled in a message and sent to an IOTA node. The IOTA node attaches this data to the Tangle and indexes it using a tag corresponding to the FL round for easy retrieval.

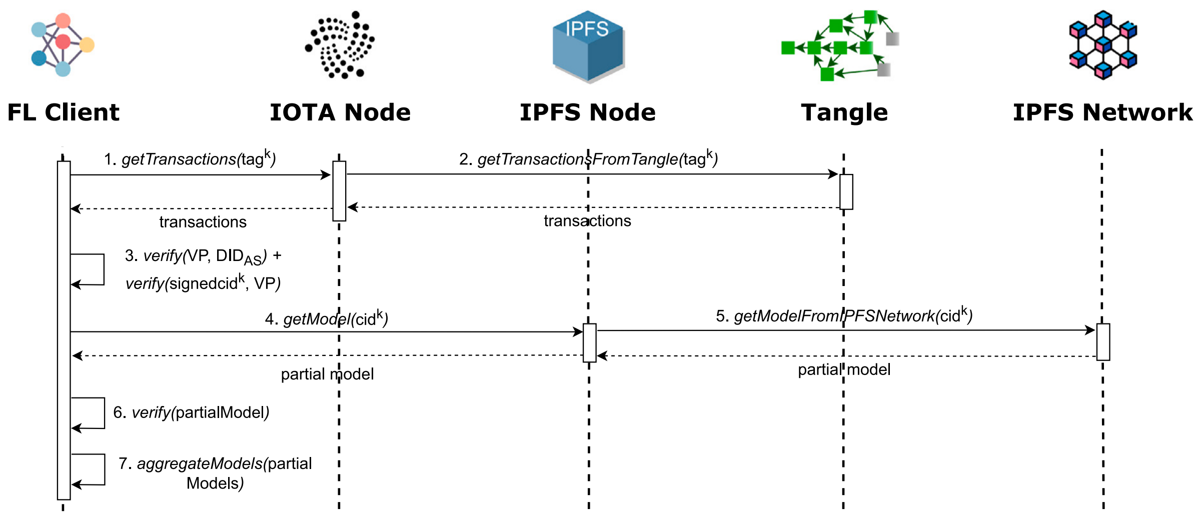
FL Client
IOTA Node
IPFS Node
Tangle
IPFS Network
1. getTransactions(tag^k)
2. getTransactionsFromTangle(tag^k)
transactions
transactions
3. verify(VP, DID_AS) + verify(signedcid^k, VP)
4. getModel(cid^k)
5. getModelFromIPFSNetwork(cid^k)
partial model
partial model
6. verify(partialModel)
7. aggregateModels(partial Models)

Fig. 5. Aggregating partial models.

Aggregating Partial Models. An FL client that has completed its training, listens for transactions associated with the tag for that specific FL round to retrieve the partial models of other participants. Specifically, the steps needed to perform the aggregation of partial models, whose workflow is sketched in Fig. 5, are reported below:

1. Using the tag of the FL round, the FL client sends a request to the IOTA node to retrieve all the transactions associated with that tag.
2. The IOTA node collects the transactions indexed with the tag and returns them to the FL client.
3. For each transaction, the FL client first verifies the validity of the included VP using the public key of the AS. Upon successful verification, it checks that the CID has been signed by the FL client linked with that VP.
4. Given the CID, the FL client requests the corresponding partial model to the IPFS node.
5. The IPFS interacts with the IPFS network and responds to the FL client with the requested partial model and its signed hash associated with that CID.
6. The FL client verifies the hash has been signed by the FL client associated with that VP and checks that the hash of the retrieved partial model matches the signed one.
7. Finally, partial models are aggregated to generate the global model, which will be used for the following FL round.

Alg. 1 presents the algorithm implemented by the FL client, which comprises both the publication and aggregation of partial models.

5. Performance results from in-the-field experimental evaluation

The purpose of this section is to demonstrate the practical feasibility of deploying our FETA solution in real-world scenarios. Our proposed architecture is designed to be highly adaptable and flexible, making it possible to integrate a wide range of FL algorithms and strategies. Our experiments are focused on showcasing the practical applicability of FETA, with particular emphasis on the respect of challenging requirements in terms of latency and power consumption, which are central to the most original aspects of our proposal for FL at the edge.

We deployed FETA on a real cluster by scaling the number of edge nodes involved in the training phase. Each edge node is equipped with an Intel(R) Core(TM) i5-3470 CPU running at 3.20 GHz and 12 GB of RAM, while each component executed at the edge is run within a Docker container.

Algorithm 1 FL Client - k FL round

```
Input: $did_{AS}, did, VC, globalModel^{k-1}, tag^k, stopCond$
Output: $globalModel^k$
$localModel^k \leftarrow trainModel(globalModel^{k-1})$
$modelHash^k \leftarrow hash(localModel^k)$
$signedHash^k \leftarrow sign(did, modelHash^k)$
$cid^k \leftarrow sendToIPFS(signedHash^k, localModel^k)$
$signedCid^k \leftarrow sign(did, cid^k)$
$VP \leftarrow generateVP(did, VC)$
$sendToIOTA(VP, signedCid^k, tag^k)$
$partialModels^k \leftarrow []$
while not stopCond do
    $VP_i, signedCid_i^k \leftarrow getFromIOTA(tag^k)$
    if $verify(VP_i, did_{AS})$ and $verify(signedCid_i^k, VP_i)$ then
        $signedHash_i^k, model_i^k \leftarrow getFromIPFS(cid_i^k)$
        if $verify(signedHash_i^k, VP_i)$ and $check(signedHash_i^k, model_i^k)$ then
            $partialModels^k.push(model_i^k)$
        end if
    end if
end while
$globalModel^k \leftarrow aggregateModels(partialModels^k)$
```

5.1. Experiments

Guaranteeing efficient communication and assessing energy consumption are two critical concerns to successfully enable FL in edge computing scenarios. Therefore, our analysis primarily aims to evaluate the latency and power consumption of FETA during an FL process. Since each component is executed within a Docker container, we obtained resource usage using docker-activity,[7] a tool that provides accurate power consumption, memory and CPU usage. To provide a comprehensive estimation, we conducted a series of experiments by scaling the number of FL clients from 2 to 8, while also varying the complexity of ML models and datasets. To ensure accuracy, each experiment was repeated 10 times, and the results were aggregated.

For the first experiment, we utilized the widely used MNIST dataset, which includes 60,000 training examples and 10,000 test examples. Each example consists of a 28 × 28 grayscale image of a handwritten

[7] https://github.com/jdrouet/docker-activity

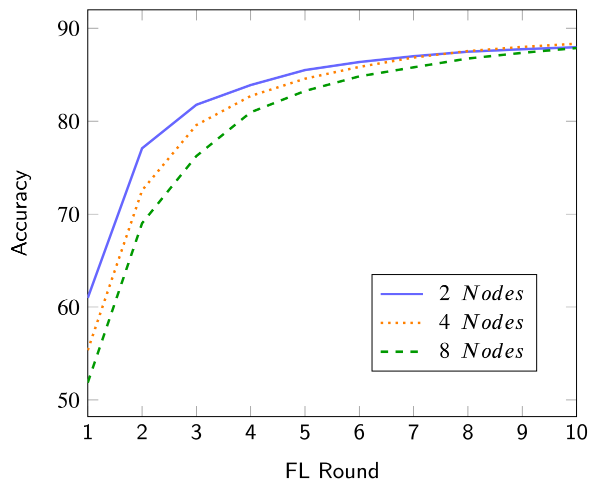

Fig. 6. Accuracy comparison with 2, 4, and 8 edge nodes for the first experiment.

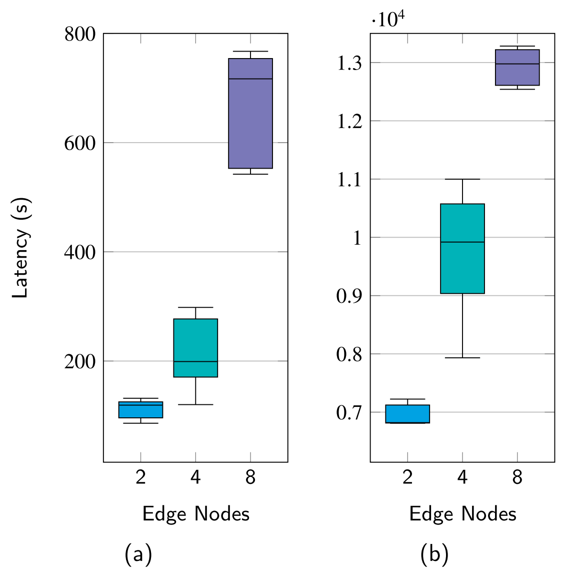

Fig. 7. Latency comparison with 2, 4, and 8 edge nodes in first (a) and second (b) experiments.

digit (0 to 9) along with its corresponding label. Our ML model was a neural network with three layers: a Flatten input layer that transformed the 28 × 28 image into a vector of 224 elements, a Dense layer with 128 neurons using the ReLU activation function, and an output layer with 10 neurons, one for each digit class. We employed the Adam optimization algorithm with a learning rate of 0.001 and used the sparse categorical cross-entropy loss function. For the second experiment, we aimed to increase the complexity by employing the CIFAR-10 dataset and a Deep Convolutional Neural Network (DCNN). The CIFAR-10 dataset consists of 60,000 32 × 32 color images in 10 classes, with 6,000 images per class. It contains 50,000 training images and 10,000 test images. On the other hand, the model is a more complex neural network with 2,626,634 parameters, about 20 times more than the network used for the first experiment. The network comprises multiple convolutional and fully connected layers, interspersed with activation functions, batch normalization, max pooling, and dropout layers to enhance performance and prevent overfitting. This setup allowed us to assess the performance of the framework on a more intricate dataset and a larger neural network. In both cases, the neural network was trained locally for 5 epochs, and a total of 10 FL rounds were performed. To ensure fairness and comparability, the datasets were fairly split among all the FL clients.

5.2. Performance results

In this section, we present the results of our experiments, which evaluated the accuracy, latency, power, and resource consumption of our proposal.

5.2.1. Accuracy

Our experiments prioritize the evaluation of latency and energy consumption. However, we recognize that achieving high accuracy is still a crucial aspect of any FL platform. An FL platform that sacrifices accuracy for the sake of low latency and energy consumption may not be widely adopted. Thus, while ML metrics are not the primary focus of our proposal, we understand their importance and have taken steps to ensure that our platform can deliver satisfactory results in terms of accuracy. Fig. 6 reports the training results obtained using the MNIST dataset, showcasing the high accuracy achieved by FETA. For the sake of brevity, we only depicted the results from the first experiment. However, it is worth noting that the training performed with the CIFAR-10 dataset also demonstrates notable accuracy, reaching approximately 80%. Notably, these results are independent of the FETA architecture and are solely determined by the adopted aggregation strategy. Specifically, in these experiments, we employed FedAVG.

Table 2
Latency for authorization procedures and FL training.

Operation	No. of Nodes	Mean Exp 1	Mean Exp 2
DID Creation	2	90.10 s	94.87 s
	4	96.94 s	72.51 s
	8	94.35 s	104.48 s
VC Creation	2	5.41 ms	5.49 ms
	4	5.38 ms	5.32 ms
	8	5.82 ms	6.88 ms
VP Creation	2	1.76 ms	1.87 ms
	4	1.72 ms	1.87 ms
	8	1.92 ms	2.00 ms
FL Rounds	2	112.80 s	6951.02 s
	4	212.19 s	9258.29 s
	8	675.52 s	12918.49 s

5.2.2. Latency

In Fig. 7, we present the results of our experiments on the mean latency of the FL training as we scale up the number of edge nodes involved. The reported latency includes both the FL training and the authorization procedure. The higher latency of the second experiment is due to the complexity of the CIFAR-10 dataset and the employed ML model that demands a longer training phase. Indeed, the overhead related to FETA operations is minimal compared to the time needed to perform the training. For a detailed breakdown of the latency, please refer to Table 2. The table highlights a sublinear relationship between the number of edge nodes and the latency of the training. Our observations indicate that a smaller number of participants results in lower latency. This can be attributed to the fact that FL clients have to wait for a smaller number of contributions before aggregating them. The creation of DIDs, VCs, and VPs is an occasional process that is not affected by the number of nodes, as well as the ML model and dataset. On the other hand, the latency of their verification depends on the number of participants and it is included in the whole FL process.

5.2.3. Power consumption

Fig. 8 depicts, for both experiments, the results of the mean power consumption of each component deployed on an edge node, while Table 4 presents their respective mean and maximum power consumption. The power consumption of the IPFS node cannot be directly observed in Fig. 8 since its contribution is lower than 0.02 W and it can be

Table 3
Comparison of memory and CPU usage for each component.

Component	No. of Nodes	Exp 1				Exp 2			
		Memory (MB)		CPU (%)		Memory (MB)		CPU (%)	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
IOTA Node	2	76.88	112.39	15.06	75.88	93.24	133.95	14.95	67.75
	4	74.57	115.68	15.46	76.38	87.95	134.96	15.64	75.53
	8	76.51	117.96	16.18	97.97	91.65	141.65	15.13	73.54
IPFS Node	2	18.89	94.23	0.13	3.81	219.78	1487.89	0.11	7.70
	4	25.89	156.55	0.16	4.80	392.86	2780.80	0.15	8.43
	8	47.79	282.28	0.19	6.65	528.13	3265.12	0.24	15.34
FL Client	2	134.37	728.02	16.46	100.0	1239.20	2264.21	43.33	96.43
	4	151.62	622.24	12.83	99.0	1117.36	2280.95	21.57	95.29
	8	184.45	458.30	10.24	90.0	1107.95	3010.91	12.28	96.90

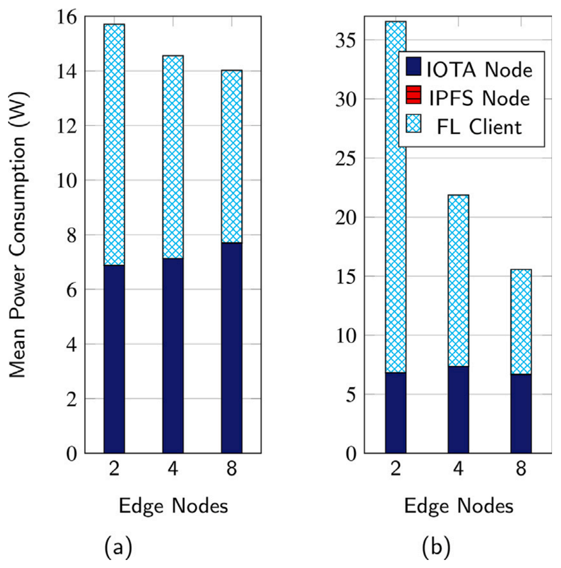

Fig. 8. Power consumption comparison with 2, 4, and 8 edge nodes in first (a) and second (b) experiments.

Table 4
Comparison of the power consumption (W) for each component.

Component	No of Nodes	Exp 1		Exp 2	
		Mean	Max	Mean	Max
IOTANode	2	6.87	44.43	6.81	33.65
	4	7.12	43.61	7.34	44.33
	8	7.69	58.25	6.95	38.17
IPFS Node	2	0.007	0.32	0.009	1.87
	4	0.01	0.49	0.01	3.64
	8	0.013	1.00	0.03	6.44
FL Client	2	8.83	64.32	29.74	74.29
	4	7.42	63.33	14.51	69.72
	8	6.31	54.16	8.88	86.70

Table 5
Power consumption, memory, and CPU usage of the authorization server.

No. of Nodes	Power Consumption (W)		Memory (MB)		CPU (%)	
	Mean	Max	Mean	Max	Mean	Max
2	0.0004	0.05	5.42	6.73	0.008	1.03
4	0.0003	0.05	6.32	8.35	0.007	1.03
8	0.0003	0.09	9.35	11.34	0.005	1.80

neglected. Concerning the IOTA node, it does not show remarkable differences when scaling the number of edge nodes or increasing the complexity of the ML model.

Interestingly, we observe that a smaller number of participants leads to higher medium power consumption. This can be attributed to the fact that with a lower number of FL clients, the latency related to waiting for all participants is reduced. However, this also means that the waiting time is very low, causing FL clients to continuously perform operations, which increases the mean power consumption per unit of time. It is worth outlining that the whole energy consumption of the FL training with 4 and 8 nodes will be higher since it demands more time. The collected metrics for the second experiment show a slight increase in power consumption due to the longer training time.

5.2.4. Resource consumption

To provide a comprehensive analysis of our proposal, we included Tables 3 and 5 which details the memory, CPU, and power usage for each component. We found that increasing the number of participants results in higher memory consumption for the FL client, AS, and IPFS Node, as they have to store more data. The IOTA node is not affected since it does not directly manage the partial models. It is worth noting that the AS exclusively handles information pertaining to the authorization phase, making its resource consumption independent of the complexity of ML models.

Regarding CPU usage, we observed that only the FL clients show remarkable differences. Specifically, we found that CPU usage decreases when scaling the number of nodes. This trend is due to the fact that when clients have low latency, they consume more resources during that period since they are continuously involved in performing operations such as training local models. As expected, the resource consumption of the FL clients is notably impacted by both the complexity of the dataset and the ML model. Employing a more complex ML model and larger dataset results in longer training times consequently increasing the overall usage of the CPU and memory.

6. Discussion

Our experimental results demonstrate that the introduced overhead by our architecture is minimal, while the numerous advantages it brings far outweigh any potential drawbacks. FETA offers an efficient and promising implementation of FL at the edge. One of the key benefits is the use of the Tangle, which ensures the same level of security and scalability as traditional blockchains used in FL integration (e.g. Ethereum), but with a more lightweight consensus protocol. This design choice translates to lower latency and power consumption, as validated by various studies in the field [40]. These features make the Tangle the ideal DLT for enhancing the robustness of FL in edge computing scenarios. Additionally, our architecture leverages DIDs and VCs to facilitate the decentralized management of FL participants and reduce reliance on central authorities. Furthermore, the advantages of using the Tangle extend to transaction confirmation times. Transactions are confirmed in seconds, compared to traditional blockchains that typically take minutes to confirm a block. This faster transaction confirmation significantly enhances the overall efficiency of the FL process, enabling quicker updates and more responsive training.

As is widely known, the time required for ML training depends on the dataset size and model complexity. In FETA, the latency primarily arises from the training phase itself, which is independent of our framework. For example, in the second experiment, the latency of FETA operations and waiting for clients' contributions constitute only about 5%–10% of the total training time. However, the relationship between the number of edge nodes and the training latency shows a sublinear pattern. On the other hand, the authorization and publishing of partial model phases remain unaffected by the complexity of ML models. The resource and power consumption of the AS and IOTA nodes exhibit negligible changes between the two experiments. Additionally, procedures involving the AS, such as DID and VC creation, occur only once, having an extremely limited impact on performance that can be safely ignored.

7. Related work

Recent years have seen an increasing number of researchers exploring the potential of integrating FL with DLTs. In this section, we review some notable works that leverage blockchain and DAGs to enable FL at the edge.

However, it is worth noting that most existing solutions tend to focus solely on the algorithms presented within their architecture, with a missing analysis of the critical metrics of latency and energy consumption, which are essential factors in assessing the feasibility of a platform that enables FL at the edge.

7.1. Blockchain-based FL

Blockchain is considered an appealing solution to address FL centralized challenges [31]. Most of the existing proposals present blockchain-based platforms to perform FL that replace the centralized server with a network of peers that, through a consensus protocol, aggregate partial models and generate a global one. In this way, participants are assured that the shared model is properly generated without bias to prefer a model over the others, increasing the trustworthiness among unknown participants that join an FL process [41].

Liu et al. [42] proposed BD-FL an FL platform that leverages blockchain and edge computing techniques. The authors introduced an incentive mechanism to encourage local devices to actively participate in model training, increasing the number of samples, and improving model accuracy. Furthermore, to improve the utilization of edge resources and reduce the transmission delay, BD-FL employs a preference-based stable matching algorithm that binds local devices to appropriate edge servers. Liu et al. [43] presented a novel approach named FedAC, which incorporates a staleness coefficient and exploits blockchain for automated aggregation of partial models. The proposed framework is designed to be robust against various security threats, such as poisoning attacks and single-point failures. In addition, the asynchronous aggregation makes the framework efficient for edge scenarios. Lu et al. [44] combined FL with blockchain to enable IoT users to collaborate on training neural networks and share training parameters with edge servers. The authors aimed to solve a data-relaying optimization problem. However, in these works, the authors have mainly focused on the proposed algorithms. Experiments have been conducted in simulated environments, but the lack of attention to implementation hinders our understanding of how these approaches can be effectively applied in practice. Additionally, existing metrics for evaluation, such as accuracy and latency, do not fully capture the energy consumption, which is a crucial concern in edge environments due to the heterogeneity of devices involved.

7.2. DAG-based FL

The constrained capabilities of IoT and edge devices hinder the use of blockchain-based architectures to perform FL. Therefore, novel approaches leveraging DAGs have been recently emerging. Schmid et al. [45] are among the first to propose a Tangle for decentralized learning, where transaction represents a full set of parameters of the shared ML model. To achieve consensus, each transaction validates the parent transactions by embedding their contributions in the training results. The obtained model is published only if it achieves higher performance than the reference model on the local dataset. However, the authors do not explain how an ML model can be embedded within a transaction given its potentially huge size, while the transaction size in the Tangle is about 64 KB. Similarly, Cao et al. [46] proposed DAG-FL, a framework that empowers asynchronous FL using DAGs. The authors also introduced two algorithms that regulate the consensus mechanism and use a smart contract for scheduling ML tasks through external agents. The paper lacks satisfactory implementation details about the DAG and smart contracts, which are crucial components of their proposal.

Recently, hybrid approaches have also been developed. Jiang et al. [47] exploit the Tangle and a consortium blockchain for enabling cooperative FL. In particular, they exploit the Tangle to secure sharing of local and global model updates. Lee et al. [48] presented a hierarchical blockchain system for robust FL that comprises a public blockchain (e.g., Ethereum) for global aggregation and local shards based on DAGs. Each shard is autonomously established by participants according to their data distribution. Thus, DAGs are used to compute partial models of nodes having a small entropy, whose contributions are then aggregated on the public blockchain. Such an approach brings a significant overhead of computational resources, which do not suit constrained environments.

The reviewed solutions are evaluated in terms of accuracy, neglecting relevant metrics such as latency and energy that are fundamental to assessing their feasibility in edge computing scenarios. Furthermore, they lack implementation details that could be beneficial to fully understand their proposal.

8. Conclusive remarks

Implementing FL in edge computing scenarios raises several challenges that need to be carefully addressed. Many researchers have proposed to enrich FL with DLT to improve scalability and build more robust platforms. However, traditional blockchain technologies still present some concerns such as long waiting times to confirm transactions and remarkable energy consumption.

In this paper, we originally provide a detailed discussion of the main design challenges for enabling efficient FL at the edge. With practical solution guidelines and examples, we discuss how most of them can be solved through the proper adoption of DAG. Furthermore, we propose FETA, a novel architecture that leverages the IOTA Tangle and addresses the concerns related to the implementation of FL at the edge. The reported in-the-field experimental results show that our FETA approach is effective in implementing FL at the edge and can be successfully used to support IoT applications of practical interest in real-world deployment environments.

CRediT authorship contribution statement

Carlo Mazzocca: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Nicolò Romandini:** Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Rebecca Montanari:** Conceptualization, Writing – review & editing, Visualization, Supervision. **Paolo Bellavista:** Conceptualization, Writing – review & editing, Visualization, Supervision.

Declaration of competing interest

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

Data availability

We shared our code.

Acknowledgments

This work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan program funded by the European Union - NextGenerationEU. All authors have read and agreed to the published version of the manuscript.

References

[1] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, D.S. Nikolopoulos, Challenges and opportunities in edge computing, in: 2016 IEEE International Conference on Smart Cloud, SmartCloud, 2016, pp. 20–26, http://dx.doi.org/10.1109/SmartCloud.2016.18.

[2] P. Bellavista, L. Foschini, A. Mora, Decentralised learning in federated deployment environments: A system-level survey, ACM Comput. Surv. 54 (1) (2021).

[3] L.U. Khan, W. Saad, Z. Han, E. Hossain, C.S. Hong, Federated learning for internet of things: Recent advances, taxonomy, and open challenges, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1759–1799.

[4] D.C. Nguyen, M. Ding, Q.-V. Pham, P.N. Pathirana, L.B. Le, A. Seneviratne, J. Li, D. Niyato, H.V. Poor, Federated learning meets blockchain in edge computing: Opportunities and challenges, IEEE Internet Things J. 8 (16) (2021) 12806–12825.

[5] S. Popov, The tangle, White Paper 1 (3) (2018) 30.

[6] H.R. Hasan, K. Salah, I. Yaqoob, R. Jayaraman, S. Pesic, M. Omar, Trustworthy IoT data streaming using blockchain and IPFS, IEEE Access 10 (2022) 17707–17721.

[7] W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 2031–2063.

[8] M. Zhang, E. Wei, R. Berry, Faithful edge federated learning: Scalability and privacy, IEEE J. Sel. Areas Commun. 39 (12) (2021) 3790–3804.

[9] Q. Xia, W. Ye, Z. Tao, J. Wu, Q. Li, A survey of federated learning for edge computing: Research problems and solutions, High-Confidence Comput. 1 (1) (2021) 100008.

[10] M. Fang, X. Cao, J. Jia, N. Gong, Local model poisoning attacks to Byzantine-robust federated learning, in: 29th USENIX Security Symposium, USENIX Security 20, USENIX Association, 2020, pp. 1605–1622, URL https://www.usenix.org/conference/usenixsecurity20/presentation/fang.

[11] H.G. Abreha, M. Hayajneh, M.A. Serhani, Federated learning in edge computing: A systematic survey, Sensors 22 (2) (2022).

[12] K. Wei, J. Li, M. Ding, C. Ma, H.H. Yang, F. Farokhi, S. Jin, T.Q.S. Quek, H. Vincent Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3454–3469.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A.y. Arcas, Communication-efficient learning of deep networks from decentralized data, in: A. Singh, J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 54, PMLR, 2017, pp. 1273–1282, URL https://proceedings.mlr.press/v54/mcmahan17a.html.

[14] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: AISTATS, 2017.

[15] Y. Chen, X. Sun, Y. Jin, Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation, IEEE Trans. Neural Netw. Learn. Syst. 31 (10) (2020) 4229–4238.

[16] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019 - 2019 IEEE International Conference on Communications, ICC, 2019, pp. 1–7, http://dx.doi.org/10.1109/ICC.2019.8761315.

[17] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, R. Yonetani, Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data, in: ICC 2020 - 2020 IEEE International Conference on Communications, ICC, 2020, pp. 1–7, http://dx.doi.org/10.1109/ICC40277.2020.9149323.

[18] M.H.u. Rehman, A.M. Dirir, K. Salah, E. Damiani, D. Svetinovic, TrustFed: A framework for fair and trustworthy cross-device federated learning in IIoT, IEEE Trans. Ind. Inform. 17 (12) (2021) 8485–8494.

[19] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, M. Guizani, Reliable federated learning for mobile networks, IEEE Wirel. Commun. 27 (2) (2020) 72–80.

[20] R. Yu, P. Li, Toward resource-efficient federated learning in mobile edge computing, IEEE Network 35 (1) (2021) 148–155.

[21] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-IID data, 2018.

[22] L. Liu, J. Zhang, S. Song, K.B. Letaief, Client-edge-cloud hierarchical federated learning, in: ICC 2020 - 2020 IEEE International Conference on Communications, ICC, 2020, pp. 1–6, http://dx.doi.org/10.1109/ICC40277.2020.9148862.

[23] S.M. Shah, V.K.N. Lau, Model compression for communication efficient federated learning, IEEE Trans. Neural Netw. Learn. Syst. (2021) 1–15.

[24] D. Li, J. Wang, FedMD: Heterogenous federated learning via model distillation, 2019, CoRR abs/1910.03581.

[25] X. Yao, T. Huang, C. Wu, R. Zhang, L. Sun, Towards faster and better federated learning: A feature fusion approach, in: 2019 IEEE International Conference on Image Processing, ICIP, 2019, pp. 175–179, http://dx.doi.org/10.1109/ICIP.2019.8803001.

[26] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, S. Guo, A survey of incentive mechanism design for federated learning, IEEE Trans. Emerg. Top. Comput. 10 (2) (2022) 1035–1044.

[27] T. Song, Y. Tong, S. Wei, Profit allocation for federated learning, in: 2019 IEEE international conference on big data, Big Data, 2019, pp. 2577–2586, http://dx.doi.org/10.1109/BigData47090.2019.9006327.

[28] V. Mothukuri, R.M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Gener. Comput. Syst. 115 (2021) 619–640.

[29] Y. Sarikaya, O. Ercetin, Motivating workers in federated learning: A Stackelberg game perspective, IEEE Network. Lett. 2 (1) (2020) 23–27.

[30] J. Konečný, H.B. McMahan, F.X. Yu, P. Richtarik, A.T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, in: NIPS Workshop on Private Multi-Party Machine Learning, 2016.

[31] W. Issa, N. Moustafa, B. Turnbull, N. Sohrabi, Z. Tari, Blockchain-based federated learning for securing internet of things: A comprehensive survey, ACM Comput. Surv. 55 (9) (2023).

[32] S. Ko, K. Lee, H. Cho, Y. Hwang, H. Jang, Asynchronous federated learning with directed acyclic graph-based blockchain in edge computing: Overview, design, and challenges, Expert Syst. Appl. 223 (2023) 119896.

[33] J. Geng, N. Kanwal, M.G. Jaatun, C. Rong, DID-EFed: Facilitating federated learning as a service with decentralized identities, in: Evaluation and Assessment in Software Engineering, in: EASE 2021, Association for Computing Machinery, New York, NY, USA, 2021, pp. 329–335, http://dx.doi.org/10.1145/3463274.3463352.

[34] W3 Recommendation, Decentralized identifiers (DIDs) v1.0, 2022.

[35] W3 Recommendation, Verifiable Credentials Data Model v1.1, 2022.

[36] B.G. Kim, Y.-S. Cho, S.-H. Kim, H. Kim, S.S. Woo, A security analysis of blockchain-based did services, IEEE Access 9 (2021) 22894–22913.

[37] N. Romandini, C. Mazzocca, R. Montanari, Federated learning meets blockchain: A power consumption case study, in: 2023 31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP, 2023, pp. 206–211, http://dx.doi.org/10.1109/PDP59025.2023.00040.

[38] W. Wang, D.T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, D.I. Kim, A survey on consensus mechanisms and mining strategy management in blockchain networks, IEEE Access 7 (2019) 22328–22370.

[39] IOTA Wiki, Energy Efficiency, 2022, https://wiki.iota.org/learn/about-iota/energy-efficiency/.

[40] Q. Wang, J. Yu, S. Chen, Y. Xiang, SoK: DAG-based blockchain systems, ACM Comput. Surv. 55 (12) (2023).

[41] C. Mazzocca, N. Romandini, M. Mendula, R. Montanari, P. Bellavista, TruFLaaS: Trustworthy federated learning as a service, IEEE Internet Things J. (2023) 1.

[42] S. Liu, X. Wang, L. Hui, W. Wu, Blockchain-based decentralized federated learning method in edge computing environment, Appl. Sci. 13 (3) (2023).

[43] Y. Liu, Y. Qu, C. Xu, Z. Hao, B. Gu, Blockchain-enabled asynchronous federated learning in edge computing, Sensors 21 (10) (2021).

[44] Y. Lu, X. Huang, K. Zhang, S. Maharjan, Y. Zhang, Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks, IEEE Trans. Ind. Inform. 17 (7) (2021) 5098–5107.

[45] R. Schmid, B. Pfitzner, J. Beilharz, B. Arnrich, A. Polze, Tangle ledger for decentralized learning, in: 2020 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW, 2020, pp. 852–859, http://dx.doi.org/10.1109/IPDPSW50202.2020.00144.

[46] M. Cao, L. Zhang, B. Cao, Toward on-device federated learning: A direct acyclic graph-based blockchain approach, IEEE Trans. Neural Netw. Learn. Syst. (2021) 1–15.

[47] L. Jiang, H. Zheng, H. Tian, S. Xie, Y. Zhang, Cooperative federated learning and model update verification in blockchain-empowered digital twin edge networks, IEEE Internet Things J. 9 (13) (2022) 11154–11167.
[48] J. Lee, W. Kim, DAG-Based blockchain sharding for secure federated learning with non-IID data, Sensors 22 (21) (2022).

Carlo Mazzocca received his M.Sc. and B.Sc. degrees in Computer Engineering in 2018 and 2020, respectively, both from the University of Naples Federico II, Italy. He is currently a Ph.D. student in Computer Science and Engineering at the University of Bologna, Bologna, Italy. His research interests mainly include security mechanisms based on distributed ledger technologies, authentication and authorization solutions for the cloud-to-thing continuum.

Nicolò Romandini graduated from the University of Bologna, Italy, where he received M.Sc. degree in computer science engineering. He is currently a Ph.D. student at the Department of Computer Science and Engineering at the University of Bologna. His research focuses mainly on blockchain, cybersecurity and machine learning, and how to integrate them into IoT domains.

Rebecca Montanari full professor at the University of Bologna since 2020 carries out her research in the area of information security and of the design/development of middleware solutions for the provision of services in mobile and IoT systems. Her research is currently focused on blockchain technologies to support various supply chains, including agrifood, manufacturing and fashion and on security systems for Industry 4.0.

Paolo Bellavista received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 2001. He is currently a Full Professor with the University of Bologna. His research interests include middleware for mobile computing, QoS management in the cloud continuum, infrastructures for big data processing in industrial environments, and performance optimization in wide-scale and latency-sensitive deployment environments. He serves on the Editorial Boards of IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON SERVICES COMPUTING, ACM CSUR, ACM TIOT, and PMC (Elsevier). He is the Scientific Coordinator of the H2020 IoTwins Project (https://www.iotwins.eu).