Expert Systems With Applications 223 (2023) 119896

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Asynchronous federated learning with directed acyclic graph-based blockchain in edge computing: Overview, design, and challenges

Seoyoung Ko [a], Keewoo Lee [b], Hyunhum Cho [c], Yoonjae Hwang [d], Huisu Jang [e,*]

[a] School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, United States
[b] Department of Mathematical Sciences, Seoul National University, 1, Gwanak-ro, Gwanak-gu, Seoul, 08826, Republic of Korea
[c] Department of Industrial Engineering, Seoul National University, 1, Gwanak-ro, Gwanak-gu, Seoul, 08826, Republic of Korea
[d] DSRV, 38, Bongeunsa-ro 20-gil, Gangnam-gu, Seoul, 06126, Republic of Korea
[e] School of Finance, Soongsil University, 369, Sangdo-ro, Dongjak-gu, Seoul, 06978, Republic of Korea

ARTICLE INFO

Keywords:
Asynchronous federated learning
Blockchain
Directed acyclic graph
Edge computing
Internet of Things
Security

ABSTRACT

Asynchronous Federated Learning (AFL) has been introduced to improve the efficiency of FL by reducing the latency of Machine Learning (ML) model aggregation, particularly in the Internet of Things (IoT) environment. Meanwhile, decentralized FL, such as leveraging blockchain and directed acyclic graph (DAG)-based ledgers, also has drawn much attention to the integration of FL owing to the security benefit against single-point-failure and Byzantine fault tolerant consensus. We observe that the inherent network asynchrony of DAG-based ledgers is beneficial for implementing asynchronous FL particularly in edge computing domains, even though there is no existing survey work that provides a fundamental overview of such integration. This paper surveys on the integration of asynchronous FL with DAG, which we call AFL-DAG, as a promising approach to realize the intersection of decentralized FL and asynchronous FL. Motivated by the lack of a concrete taxonomy of asynchronous FL and a global model, especially with decentralized FL, we introduce universally applicable terminologies and the extensive classification method of FL in terms of asynchrony. Based on the proposed taxonomy, we present a generic system model of AFL-DAG for edge computing applications. To provide a horizontal overview and cover fundamental concepts in AFL-DAG, we identify four critical design factors and the state-of-the-art solutions are discussed accordingly. Future directions to achieve a practical AFL-DAG are also highlighted. Finally, we explore the opportunities of AFL-DAG by investigating a popular edge computing application, the on-device crowdsourcing, and provide a high-level evaluation of AFL-DAG compared to blockchained-FL.

1. Introduction

Due to the rapid development of both machine learning (ML) and Internet of Things (IoT), the massive amounts of data generated by edge devices, such as smartphones, laptops, and AI speakers, have been largely exploited to train ML models. To enhance the data privacy and reduce the latency of training, *Multi-Access Edge Computing* (MEC, also called *Mobile Edge Computing*) has been introduced, which performs computation tasks or applications in close proximity to edge users via a distributed edge network, removing the requirement of cloud computing (Hu, Patel, Sabella, Sprecher, & Young, 2015). Similarly, as opposed to conventional ML, federated learning (FL) has recently emerged as a prominent privacy-preserving ML solution for edge computing applications; FL enables the training of a global model by coordinating several edge devices to perform model training without sharing their raw data. In FL, each device locally trains the ML model using its dataset, and a centralized server aggregates the learning parameters retrieved from the multiple devices.

Although FL has been extensively studied in many different works, several challenges still exist, especially in the IoT industry. First, the centralized aggregation server is vulnerable to the single-point-failure issue and several abnormal actions by malicious devices, such as *poisoning* attacks. Given the distributed property of data orchestration in FL, the centralized aggregation server has several challenges with scalability and security. Second, classical synchronous FL has a critical performance bottleneck, the so-called straggler problem, and it is challenging to coordinate the FL process entirely because of the different resources in edge devices, such as central processing unit

* Corresponding author.
E-mail addresses: sko2@andrew.cmu.edu (S. Ko), activecondor@snu.ac.kr (K. Lee), heum@dm.snu.ac.kr (H. Cho), yoonjae.hwang@dsrvlabs.com (Y. Hwang), yej523@ssu.ac.kr (H. Jang).

https://doi.org/10.1016/j.eswa.2023.119896
Received 28 July 2022; Received in revised form 2 December 2022; Accepted 14 March 2023
Available online 21 March 2023

(CPU)/memory resources or network bandwidth, particularly in a dynamic IoT environment.

To mitigate the aforementioned single-point-failure problem and anomaly detection issue triggered by a centralized aggregation server, leveraging blockchain in FL, the so-called *blockchained-FL*, has been actively discussed owing to its decentralization and security benefits (Nguyen, Ding, et al., 2021). Blockchained-FL does not require a central server for the aggregation. Instead, edge devices communicate with each other and build the global model in a peer-to-peer manner. On the other hand, as a solution to resolve the straggler problem, asynchronous FL has recently been introduced, which aggregates a local model asynchronously without waiting for other local models from low-performing devices (Xie, Koyejo, & Gupta, 2019).

As a combination of asynchronous and decentralized FL, a few studies have proposed the integration of a directed acyclic graph (DAG)-based ledger with FL (Cao, Zhang, & Cao, 2021; Lu, Huang, Zhang, Maharjan, & Zhang, 2020; Schmid, Pfitzner, Beilharz, Arnrich, & Polze, 2020; Yuan, Cao, Peng, & Sun, 2021). As these studies have discussed, the DAG-based architecture provides considerable advantages for asynchronous FL, especially in an edge computing environment, when compared to the legacy linear blockchain-based architecture. We observe that the DAG's inherent property of *network asynchrony* (Zhao & Yu, 2019) naturally enables flexible integration of asynchronous FL, because a DAG network itself can tolerate the stale transaction, which is a critical prerequisite of asynchronous FL. In Section 4, we further discuss our integration motivation.

Inspired by the above observations, in this study, we focus on asynchronous FL empowered by the DAG architecture. Although blockchained-FL has been actively investigated in previous survey studies (Ali, Karimipour, & Tariq, 2021; Nguyen, Ding, et al., 2021; Unal et al., 2021), the research that horizontally analyzes DAG-based asynchronous FL (the so-called *AFL-DAG*) is still lacking. In the existing literature, we also notice that the classification between asynchronous FL and synchronous FL is obscure, especially when it comes to decentralized FL. To fill this research gap, we propose an extensive terminology and classification methods to clarify the global model and asynchronous FL concepts. Based on the proposed methodology, we present a comprehensive overview of AFL-DAG and further identify its potential opportunities and concerns in edge computing. The main contributions of this study are as follows:

- We provide a concrete terminology of the global/reference model that can be universally utilized not only for the existing centralized FL, but also for the decentralized FL. In addition, we present an extensive method to horizontally classify FL architectures into synchronous, asynchronous, pseudo-asynchronous, and hybrid-asynchronous FL.
- We propose a generic system architecture for AFL-DAG applications in edge computing. Based on the system model, we identify four key consideration factors while designing AFL-DAG (global/reference model, FL classification, consensus, and security) and summarize the key solutions of the latest research works.
- We highlight the outlook learned from the prior studies and discuss the technical challenges and future directions towards the full realization of AFL-DAG in practice.
- Finally, we explore a new opportunity brought by AFL-DAG in one of the popular application domains in edge computing, the on-device crowdsourcing, and compare with blockchained-FL.

The rest of this paper is organized as follows. Section 2 provides a brief introduction to blockchain, DAG, FL, and blockchained-FL. Then, an in-depth overview of asynchronous FL and the proposed method for asynchronous FL classification is described in Section 3. In Section 4, we elaborate on the integration motivation of AFL-DAG in edge computing and present a generic system model. Section 5 mainly covers the design and application of AFL-DAG in edge computing. We first horizontally analyze the state-of-the-art research works on AFL-DAG and blockchained-FL based on the four major categories, and discuss the outlook and possible future research directions. One of the applications in edge computing and comparative evaluation of AFL-DAG are presented in Section 5. Finally, we conclude this study in Section 6.

2. Preliminaries

2.1. Blockchain and DAG

Blockchain is a distributed ledger to maintain tamper-proof transaction records. Blockchain utilizes cryptographic techniques to function as an immutable database managed by decentralized entities in a peer-to-peer manner. In a blockchain system, every node in the decentralized network must agree on the same state of the database for every round. Therefore, a consensus mechanism to create a block, which means to agree on the state, is one of the primary components of the blockchain system.

Proof-of-work (PoW) is a conventional consensus mechanism, which is suggested from a whitepaper on Bitcoin. The author shows that the blockchain would be probabilistically tamper-proof if the hash rate of the malicious node is lower than that of the honest nodes (Nakamoto, 2008). The blockchain system implemented in Bitcoin considers two distinctive players: those who issue transactions and those who approve transactions.

Peers in the blockchain system are completely separated into the demand and supply for transaction approvals. Participants who wish to issue a transaction must pay the transaction fee, the cost of approving the transaction, to the miner who is the supplier to the demand for approvals. Incentives for transaction approvals are the only reason to participate in the system for the miners, who validate transactions and create a block. The miners select transactions in the order of a higher transaction fee because the penalty to non-approve the transaction is absent. Therefore, the transaction fee increases as the number of transactions to be approved increases.

On the other hand, a DAG-based blockchain has been introduced to provide features that the legacy linear blockchain failed to deliver, which are crucial in an IoT environment such as better scalability and shorter transaction time for instant micropayments. In this paper, we will interchangeably refer to DAG-based blockchain as DAG and DAG-based ledger. Theoretically, DAG-based blockchain is not blockchain because blockchain stores transactions in a shape of linked list while DAG stores transactions in a forkable tree data structure. However, we consider DAG as an extension of blockchain and compare DAG with blockchain, given that both of them function as a decentralized ledger and maintain the ledger by peer-to-peer connected nodes. Cao et al. (2021), Zhao and Yu (2019) have also utilized the terminology of DAG-based blockchain in a similar way. Among all different types of DAG-based ledgers (Baird, 2016; Lewenberg, Sompolinsky, & Zohar, 2015; Popov, 2018), we will consider IOTA (Popov, 2018) as a representative of DAG in this paper.

As a data structure of IOTA's DAG-based consensus, Tangle is a block-free DAG-based distributed ledger in which the group issuing transactions is identical to the group confirming transactions. The tangle, a directed acyclic graph, consists of vertices and directed edges, where the vertices are transactions, and the directed edges are approvals of other transactions. Every incoming transaction in IOTA needs to approve two unapproved transactions, so-called *tips*, and each node selects two transactions to approve by a *tip selection algorithm*. Therefore, all participants in the system have the same demand (purpose) for issuing a transaction. The tangle presents a conditional cost of mandatory transaction confirmations to achieve the purpose of issuing a transaction. In other words, the issuance of a new transaction needs the incentive of a former transaction approval, so that the demand and

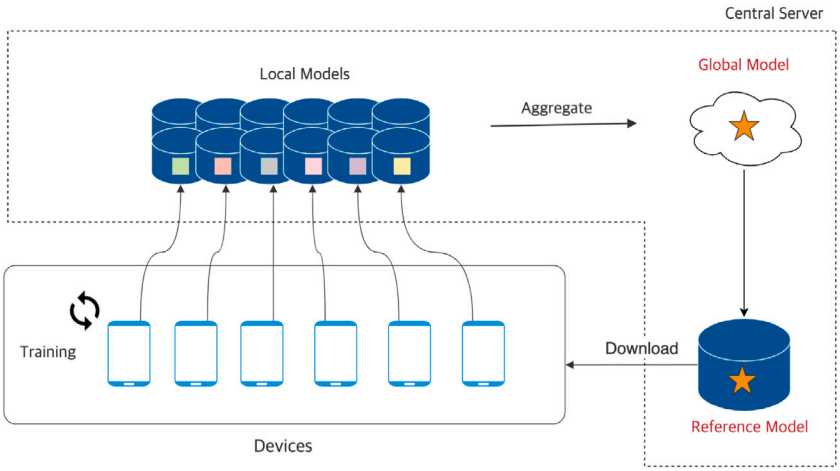

Fig. 1. Classic FL architecture with reference and global model.

supply are matched within the system without an exogenous supply such as the miners. Thus, participants try to maintain the distributed ledger without a transaction fee. It is noted that DAG-based ideas have been considered to circumvent several drawbacks of the blockchain in both the cryptocurrency and IoT fields (Lewenberg et al., 2015; Poon & Dryja, 2015; Sompolinsky & Zohar, 2013).

2.2. Federated learning (FL)

FL is a distributed machine learning technique from decentralized data that enables edge devices to train a model without exchanging their private data. Classic FL was initially proposed in McMahan, Moore, Ramage, Hampson, and y Arcas (2017) where a centralized server aggregates the local models retrieved by different devices and builds the global model shown in Fig. 1. FL mainly aims at achieving collaborative ML and addressing critical issues regarding data privacy and security. Classic FL trains the model according to the following process:

1. The authorized central server initializes a global model M_g^0 for training and determines hyper-parameters such as the learning rate. Then, the initial model M_g^0 and the hyper-parameters are sent to local clients to train.
2. At each training step t, local client i trains the model M_i^{t-1}, where $i \in [1, N]$. Let M_i^t be the learned model. Then local clients send the local model M_i^t to the central server when the training is finished.
3. The central server waits for all local models to be sent and aggregates them to generate a next global model M_G^t by the following process: $M_G^t = \sum_{i=1}^{N} M_i^t$. Then a new global model M_G^t is sent to every local client for the next training step $t+1$.

However, there are several existing problems in classic FL (Kairouz et al., 2019). First, FL is susceptible to so-called *poisoning* attacks as other collaborative ML in which malicious data providers submit wrongly generated data/label (or gradients in FL) to compromise the model to be trained (Bagdasaryan, Veit, Hua, Estrin, & Shmatikov, 2020; Nguyen, Rieger, Miettinen, & Sadeghi, 2020; Xie, Huang, Chen, & Li, 2019). Defenses against these poisoning attacks are vital in the FL setting because participating clients cannot be assumed as trusted parties in typical FL scenarios. In addition, privacy-related attacks are feasible while sharing the model gradients in FL systems. Additionally, statistic/system heterogeneity and high communication cost are critical issues that affect the performance of the FL global model. Statistic heterogeneity is triggered by the non-IID (independent and identically distributed) data due to the property of FL in that it does not share the data itself. On the other hand, system heterogeneity is an issue because of the diverse device environments such as CPU, memory, or network bandwidth.

The asynchronous approach in FL has been actively discussed in many studies as an effective way to solve several performance and security issues in the IoT environment (Lim et al., 2020; Xu, Qu, Xiang, & Gao, 2021). Asynchronous FL is a mechanism in which a centralized server aggregates a global model as soon as it collects a local model (Xu et al., 2021). For this reason, it is efficient to mitigate a straggler issue, in which all participating clients need to wait for the slowest client or dropping-out client within a round. Especially, the straggle issue can be exacerbated in the circumstance where each device has a large deviation in terms of computing capacity, network bandwidth/reliability, or so on. The research in Xie, Koyejo, and Gupta (2019) proposes a new algorithm for asynchronous federated optimization and shows the convergence of the machine learning model. Another study Nguyen, Malik, et al. (2021) discusses the pros and cons of asynchronous and synchronous FL and proposes a novel aggregation scheme, the so-called FedBuff, which covers both drawbacks from asynchronous and synchronous FL. Further discussion regarding the FL classification in terms of asynchrony will be covered in Section 3.2.

On the other hand, decentralized FL by leveraging blockchain technology has emerged as a promising solution to mitigate the poisoning attacks and single-point-failure triggered by the centralized aggregator in classic FL, which will be covered in the following section (Feng et al., 2021; Kim, Park, Bennis, & Kim, 2020).

2.3. Blockchained federated learning (blockchained-FL)

As a representative of decentralized FL, blockchain has been actively discussed in many studies as having a great potential to improve the security of FL due to the decentralization of the model aggregation and byzantine fault tolerant characteristics. This approach is called *blockchained-FL* or *FLChain* (Kim et al., 2020; Liu et al., 2022; Majeed & Hong, 2019; Nguyen, Ding, et al., 2021).

The general operation procedure of blockchained-FL is shown in Fig. 2. Although the actual blockchain network is running by equal nodes, for better understanding, we divide the nodes into two types: a group of miner nodes and a group of training nodes. The overall FL procedure of blockchained-FL is as follows. (1) First, the training nodes download the machine learning model as the model distribution process and start the local model training by their local data. (2) Second, once the training nodes finish the local training, they submit the local parameter to their peer nodes via a peer-to-peer network. (3) Third, the miner nodes validate and aggregate the submitted local models and generate a new block that includes the global model which has better accuracy than the previous global model. (4) Fourth, via a consensus mechanism in each system, the global model is finalized and propagated to all training nodes again.

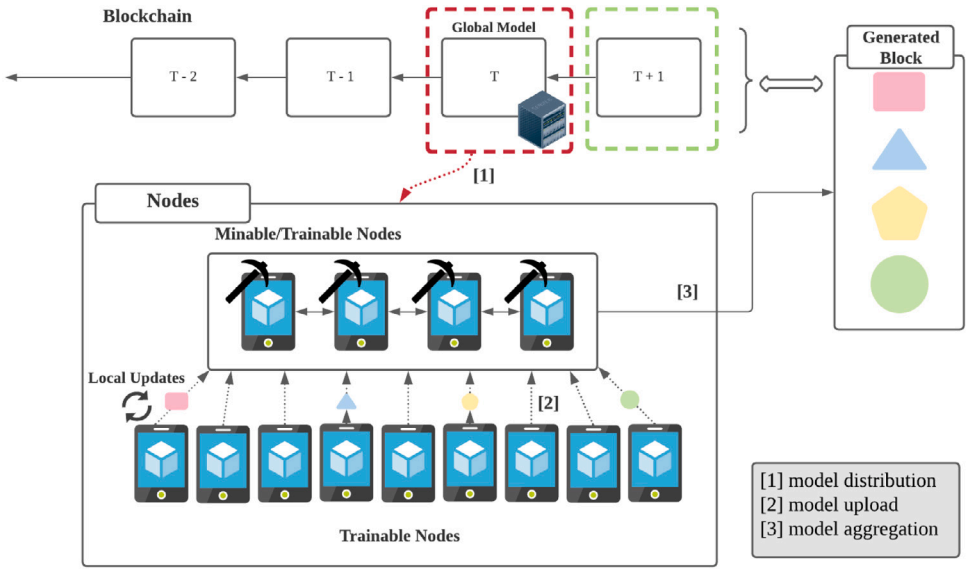

Fig. 2. Blockchained-FL architecture.

Table 1
A general comparison of asynchronous and synchronous FL.

	Straggler problem	Aggregation frequency	Communication latency	Local gradient privacy
Synchronous FL	Vulnerable	Low	High	Achievable via DP, MPC, HE
Asynchronous FL	Relatively unaffected	High	Low	No known effective technique

3. Classification for asynchronous federated learning

Although asynchronous FL is actively studied today (Kairouz et al., 2019; Xu et al., 2021), there are no existing studies that horizontally classify between asynchronous and synchronous FL, when it comes to decentralized FL solutions. To fill such a research gap, we propose a novel definition of global models in FL solutions and present a generic classification method that can be easily utilized for decentralized FL. In this section, we first provide an in-depth comparison the pros and cons of asynchronous and synchronous FL. And then, we propose a generic terminology and algorithm to classify FL works in terms of asynchrony.

A general local training process in asynchronous FL consists of the following steps:

1. In the ith iteration, a local server receives the initial model, which is a model to learn in this turn and could be different for each local server.
2. After the learning process, a local sever transmits its trained model to an administration server or other networks, which can generate the global model that aggregates each locally trained model.
3. The above two steps are repeated again, and the initial model for learning received in the $i+1$-th step may be different from the global model aggregated in the second step of the ith step.

As shown in Fig. 3(a), synchronous FL aggregates a global model synchronously in which each client must wait for the other clients during the aggregation. In contrast, asynchronous FL stands for asynchronous aggregation in which each client's local update is immediately applied to the model as in Fig. 3(b).

3.1. Comparison between asynchronous and synchronous FL

As shown in Table 1, we categorize four factors to be considered when comparing asynchronous and synchronous FL. In this section, we will discuss these metrics from a performance and security perspective. Such comparison gives us a general understanding of the properties of asynchronous FL, which will be discussed in the following sections.

3.1.1. Performance

Regarding the performance, we compare asynchronous and synchronous FL based on the straggler problem, the frequency of the global model aggregation, and the latency of the communication for the aggregation.

Synchronous FL is prone to the straggler problem, which deteriorates in cases of device heterogeneity or insecure network connectivity (Nguyen, Malik, et al., 2021). However, asynchronous FL is relatively unaffected by the straggler problem because each client's update is aggregated asynchronously without waiting for the other clients' updates. In terms of communication efficiency and cost, we observed that there is a lack of an extensive comparison framework which considers different factors such as the convergence speed, device environment, and cost calculation. Therefore, we concluded that it is controversial to horizontally compare the communication efficiency of asynchronous and synchronous FL, although some research studies have discussed these topics (Liu et al., 2020; Lu et al., 2020). Instead, we discuss the aggregation frequency and communication latency here. In the case of asynchronous FL, because the aggregation happens asynchronously, the process of building global models happens frequently unlike synchronous FL. On the other hand, because synchronous FL is more vulnerable to stragglers, the overall communication latency is bigger than that of asynchronous FL. Aside from the efficiency and cost of communication, we compare the aggregation frequency and communication latency by investigating the properties of each type of FL.

3.1.2. Privacy

Privacy in FL is claimed by communicating local gradients rather than raw data. However, several studies suggest that this may not be enough: (Melis, Song, Cristofaro, & Shmatikov, 2018; Nasr, Shokri, & Houmansadr, 2019; Salem, Bhattacharya, Backes, Fritz, & Zhang, 2019; Wang et al., 2019; Zhu & Han, 2020). Because there is no cryptographic tool to hide local gradients containing side information about the local data of clients, it may be non-trivial but far from impossible to invert the data.

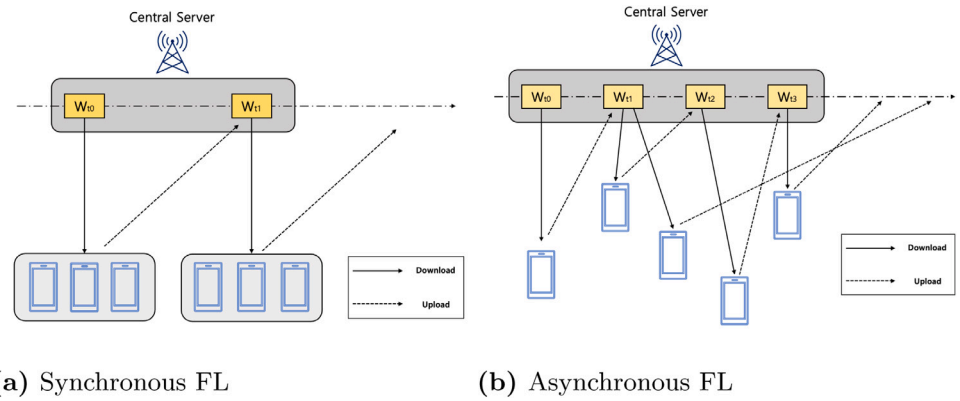

Fig. 3. Synchronous and asynchronous FL.

In a centralized FL framework, the local gradients of each client can be fully accessed by the server which performs the role of the aggregator. The situation increases the possibility of monitoring via so-called *inference attacks*. There are several possible attacks: *membership inference attack*, which determines whether certain data was used for training (Nasr et al., 2019); *property inference attack*, which learns the properties of the training data irrelevant to the original learning task (Melis et al., 2018); *distribution estimation attack*, which estimates the proportions of the training labels in the data (Salem et al., 2019); and even *reconstruction attack*, which actually reconstructs the training data (Salem et al., 2019).

To mitigate these kinds of attacks, a line of research has focused on improving *local gradient privacy* so that the server does not get to know the local gradients of the clients. These studies utilize cryptographic tools such as (distributed) differential privacy (DP)[1] (Kairouz, Liu, & Steinke, 2021), secure multi-party computation (MPC) (Fereidooni et al., 2021), and homomorphic encryption (HE) (Sav et al., 2020). The essence of these studies is to perform secure aggregation among the clients themselves or in an encrypted/obfuscated form without trusting the central server so that the server has access only to the aggregated results. However, in an asynchronous setting, a client and its exact local gradient value are linkable by design, as each local gradient is updated immediately without any aggregation process. That is, the server has access to who submitted which gradient. In this aspect, achieving local gradient privacy in an asynchronous FL scenario seems essentially impossible without modifying the framework. We leave designing a local gradient private asynchronous FL system as future work. Studying and applying an analog of secure shuffling and shuffled-mode DP (Cheu, Smith, Ullman, Zeber, & Zhilyaev, 2019) in the asynchronous setting might be a promising approach.

3.2. Proposed method for asynchronous FL classification

The straggler issue, which is also referred as the transaction staleness issue in asynchronous FL, is the main reason for attempting the asynchronous update of FL. The implementations of global model aggregation in asynchronous FL have been covered in various ways (Chen, Ning, Slawski, & Rangwala, 2020; Nguyen, Malik, et al., 2021; Shi et al., 2020; Sun, Lei, Wang, Liu, & Zhang, 2020; Wang, Li, Wang, Wang, & Zeng, 2021; Wu et al., 2020). To alleviate the transaction staleness issue, the aggregation server sorts out and selects locally trained models to some extent for aggregation instead of waiting for the entire locally trained models. It is crucial for asynchronous FL to decide which local models are aggregated and when the aggregation starts for the tolerance of the stale models meanwhile keeping up the performance of the global model.

An asynchronous FL system generates a provisional model that has not been accepted as a global model yet by all devices or an authorized entity such as a central server. The provisional model is often utilized for edge devices in their local training or regarded as a virtual global model. Particularly, when it comes to decentralized asynchronous FL, the various provisional models are named differently in each proposed architecture. Few studies have been conducted to clearly define a provisional model and to compare provisional models appearing in various literature. In this study, we propose two different types of models existing in a general FL scheme, a *reference model* and a *global model* to clarify the taxonomy of the global model and the provisional model in asynchronous FL. We consider the *reference model* as a single model, which is downloaded and referred to by each client for its local training. On the other hand, the *global model* stands for the latest trained model, which is globally approved by the authenticated entities or all devices participating in the entire process. In synchronous FL, these reference and global models are the same models shown in Fig. 1. In contrast, the reference and global models are different in most of the asynchronous FL systems.

Through this observation, both centralized and decentralized FLs fall between the synchronous and fully asynchronous FL models according to the scale of asynchrony. Depending on the system design of each architecture, there can be multiple variants of asynchronous FL such as pseudo and hybrid asynchronous FL. Although both hybrid and pseudo asynchronous FLs work similarly in terms of FL model training, we distinguish those due to the difference in the number of aggregation layers in each system design (e.g., single aggregation layer or multiple aggregation layers).

Motivated by the ambiguous definition of each type of FL especially in the decentralized FL literature, we propose a flowchart to classify FLs into four types in terms of asynchrony. As shown in Fig. 4, if the reference model and the global model are the same, a FL architecture is considered as a synchronous FL. Otherwise, if the frequency of the model aggregation is the same as the number of local model updates, the architecture is regarded as a fully asynchronous FL. In the case that the frequency of the model aggregation is smaller than the number of local updates, it can be either a pseudo or hybrid asynchronous FL depending on its system design and whether the system includes multiple aggregation layers or not.

Specifically, several existing studies (Cao et al., 2021; Hao, Zhao, & Zhang, 2020; Nguyen, Malik, et al., 2021; Shi et al., 2020; So, Ali, Güler, & Avestimehr, 2021; Stripelis & Ambite, 2021; Wu et al., 2020, 2021) utilize the *pseudo asynchronous* FL concept which has a cache or buffer before the aggregation by nature, so that the clients do not wait too long for the aggregation. It is important for *pseudo asynchronous*

[1] Other than the *distributed* DP, we can also utilize the *local* DP. In this setting, clients' data is processed with errors before sending them to the server. However, achieving local DP while maintaining a certain level of utility (accuracy) is known to be very challenging, especially with high-dimensional data (Kasiviswanathan, Lee, Nissim, Raskhodnikova, & Smith, 2011).

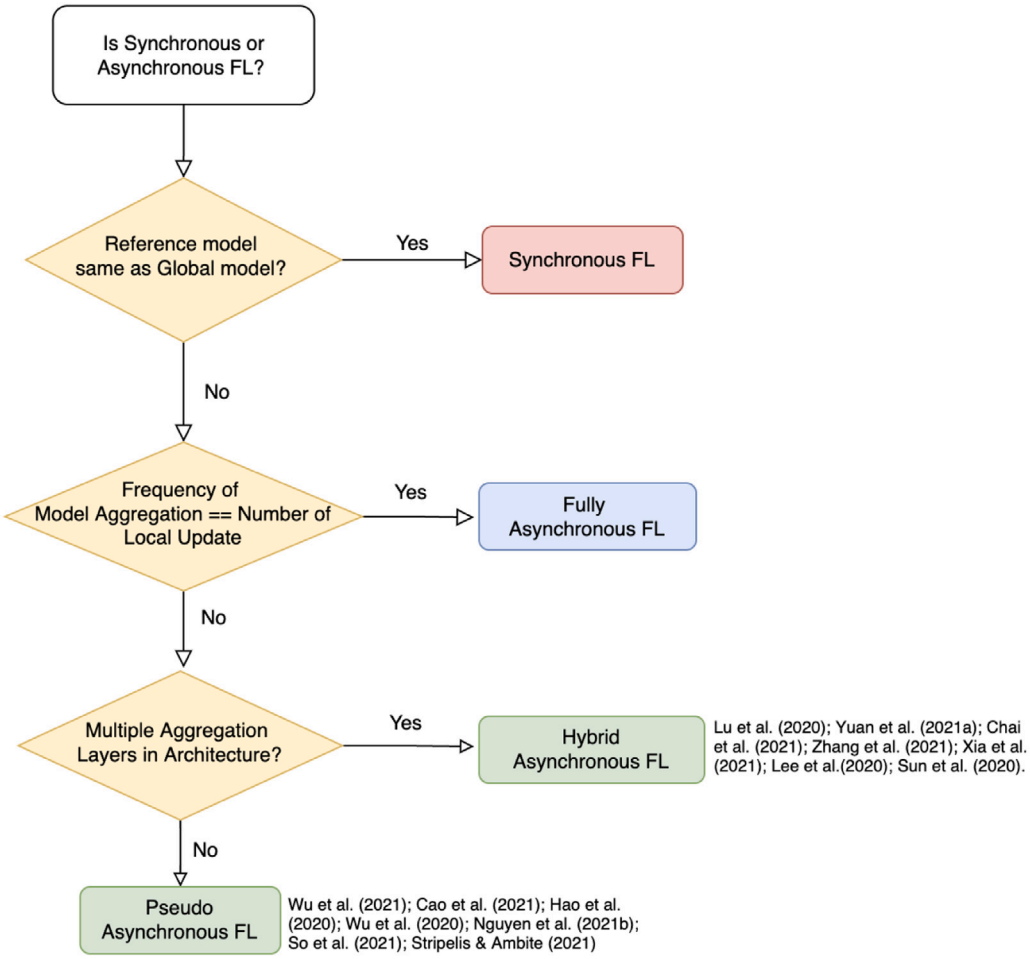

Fig. 4. Proposed flow chart to classify asynchronous/synchronous FL.

FLs to guarantee the convergence or performance of the global model while designing their global model aggregation process. Hao et al. (2020), Wu et al. (2020) proposed the evaluation function to select the locally trained model in their cached models according to large amounts of data or high computational power. Additionally, various mechanisms have been proposed to solve the overload problem due to many local models that have not yet been aggregated (Shi et al., 2020; So et al., 2021; Stripelis & Ambite, 2021). On the other side, the *hybrid asynchronous* concept stands for a mixture of different types of FL by separating into multiple layers (Chai et al., 2021; Lee, Oh, Shin, Lee, & Yoon, 2020; Lu et al., 2020; Sun et al., 2020; Xia, Li, Zhang, Wu, & Yuan, 2021; Yuan, Cao, Peng, & Sun, 2021; Zhang et al., 2021). Many studies have proposed criteria for classifying hierarchies as follows: metrics considering the gradient direction and model update delay (Xia et al., 2021), and metrics based on the data distribution and physical location to reduce global model loss and communication delay (Lee et al., 2020). We will utilize the four types of FL and the proposed algorithm to classify each blockchained-FL and AFL-DAG in the following sections.

4. Integration motivation and system model

In this section, we describe our motivation for the integration of asynchronous FL and DAG, which we call *AFL-DAG*. Then, we propose a generic system model of AFL-DAG for edge computing applications and provide a high-level overview of the entire operation.

4.1. Integration motivation

There are several reasons why DAG has drawn much consideration to be integrated with asynchronous FL in edge computing. First, DAG is often considered to be suitable for edge devices which lack hardware resources, such as CPU, storage, or network bandwidth, to run both FL tasks and mining. This is because unlike PoW-based blockchain, DAG does not require a heavy computation resource for block mining (Schmid et al., 2020). Without the heavy mining, DAG can facilitate anomaly detection without compromising performance because every node individually must validate its own previous model parameter and compare its accuracy to send its own transaction (Cao et al., 2021). Furthermore, DAG enables a shorter transaction time than blockchain, so that it enables instant micropayments or model aggregation to be utilized in various FL applications for IoT devices.

More importantly, the DAG's inherent property of network asynchrony (Zhao & Yu, 2019) naturally enables flexible integration of asynchronous FL without any additional hassles. A DAG network itself can tolerate the stale transaction, which approves outdated transactions by nature and is a critical prerequisite of asynchronous FL. For example, the linear blockchain maintains a single block at the tip height (even if there is a possibility of being forked), and if a miner created a new block that does not have the previous tip block hash information, it would be considered invalid. However, in a DAG-based ledger, the tip selection algorithm can tolerate the stale transaction, which would not be considered invalid, and every node can have different tips with each other because of its tree-based architecture (Schmid et al., 2020; Yuan,

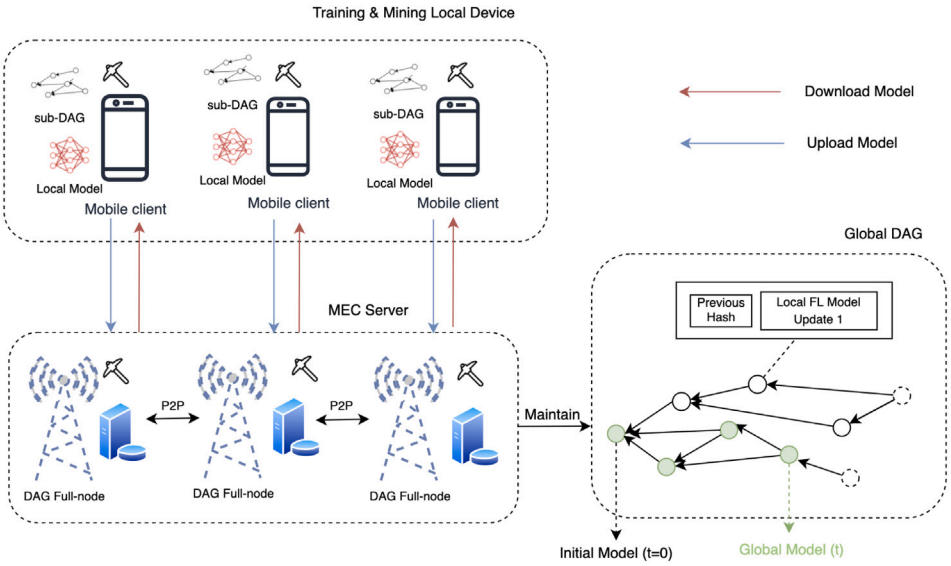

Fig. 5. A general tangle-based AFL-DAG architecture.

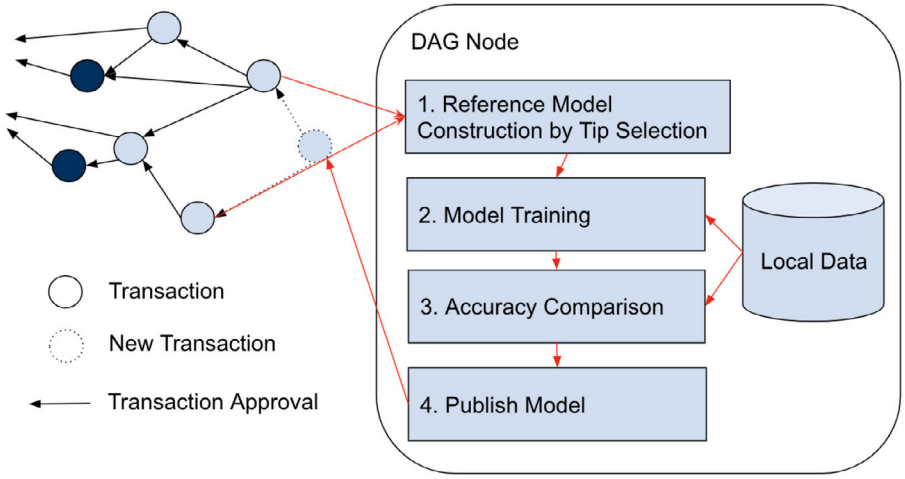

Fig. 6. AFL-DAG node training procedure.

Cao, Sun, & Peng, 2021). In other words, the fundamental asynchrony of DAG meets the crucial requirement of asynchronous FL, which can tackle the straggler issues in an IoT environment.

Because of the benefits of AFL-DAG, a few studies have proposed several DAG solutions to integrate with asynchronous FL. However, there is a huge research gap for a well-organized survey study of AFL-DAG. To fill such a research gap, we propose a generic system model of AFL-DAG, which covers the most important components in edge computing.

4.2. System model

Motivated by these unique benefits, we propose a generic AFL-DAG system architecture for edge computing applications. The system model presents the fundamental overview of how AFL-DAG works from end to end, which will be utilized in Section 5.

In the proposed AFL-DAG architecture in Fig. 5, there are two types of network entities: distributed edge devices and multi-access edge computing (MEC) servers. Due to the high computation and storage capacity, a MEC server behaves as a full node which stores all transactions and constructs the global DAG from the genesis transaction via a peer-to-peer network. On the other hand, the edge devices, which lack the hardware capacity, hold their sub-DAG and perform both training and validating by its local data. Then, the local device appends a new transaction that contains its local model-update to the peer-to-peer DAG network.

From the initial model published at $t = 0$, the global model at a certain time t is decided on by any type of predefined algorithm codified by a smart contract or a side chain so that the DAG network can agree on the global model in a decentralized way. The reference model aggregation is continuously performed by the tip selection in the edge devices. The detailed procedure of how each DAG node maintains the sub-DAG is shown in Fig. 6. In terms of DAG structure, the vertices are transactions, and the directed edges are approvals of the transactions. The general operation is summarized as follows:

1. Every edge device (DAG node) creates its own sub-DAG, which consists of transactions and the approvals of the transactions, via its peer-to-peer network. From the node-specific sub-DAG, each device selects its own tip transactions by the predefined tip selection algorithm, and builds the reference model, on which the device is going to train its local model based.
2. Then, the device starts training its local model by its privacy-sensitive data and builds the local model-update.

Table 2
A comparison of state-of-the-art blockchained/DAG-based FLs.

Scheme	Architecture	Consensus	Node type	FL classification
PermiDAG (Lu et al., 2020)	Hybrid Blockchain	DPoS (main), DAG consensus (sub)	Vehicles	Fully Asynchronous FL
DAG-FL (Cao et al., 2021)	DAG	DAG based voting	Mobile devices	Fully Asynchronous FL
ChainsFL (Yuan, Cao, Sun, & Peng, 2021)	Hybrid Blockchain	DAG consensus (main), Raft (sub)	Mobile devices	Hybrid Asynchronous FL
TangleFL (Schmid et al., 2020)	DAG	DAG consensus	IoT devices	Fully Asynchronous FL
BAFL (Feng et al., 2021)	Blockchain	Proof-of-Work	IoT devices	Pseudo Asynchronous FL
BlockFL (Kim et al., 2020)	Blockchain	Proof-of-Work	Network edges	Synchronous FL

3. The edge device validates its local model by comparing the accuracy of that with the existing reference model. Along with the model validation, the edge device performs the transaction validation such as the verification of previous transaction hash information for DAG network integrity.
4. Finally, the device broadcasts a new transaction which contains its local model-update and previous transaction information to its peer DAG node.

Due to the unique asynchrony property of DAG, there is no single tip status that is admitted by all participating nodes from the DAG system itself. Therefore, AFL-DAG comes as a promising approach to enable both decentralized and asynchronous FL in edge computing.

5. AFL-DAG in edge computing: Design and applications

Unlike an existing survey work solely on blockchained-FL (Nguyen, Ding, et al., 2021) or asynchronous FL (Xu et al., 2021), we focus on the survey studies of asynchronous FL and DAG (so-called AFL-DAG). In this section, we discuss the most important design aspects of AFL-DAG and analyze a representative application in various edge computing domains. Based on the design aspects, we analyze the latest AFL-DAG systems and compare those with blockchained-FL. The outlooks for future research are also presented.

5.1. AFL-DAG design aspects and analysis

In this section, we propose several key design aspects for AFL-DAG and present a general comparison table (Table 2) based on state-of-the-art AFL-DAG studies. For a comprehensive understanding of our survey, we added representative blockchained-FL studies as a comparison group. We cover four main categories for the design aspects of AFL-DAG: (1) global model/reference model construction, (2) FL classification, (3) consensus, and (4) security. In each category, we discuss the problem statement and key solutions of the latest studies.

5.1.1. Global/reference model construction

In Section 3.2, we distinguish the global model from the reference model for clarity. Based on the proposed terminology, we holistically compare how the prior research on AFL-DAG/blockchained-FL constructs their global and reference models, although these models are referred to differently by their own names in each study. Unlike synchronous FL where the reference model and the global model are the same, these two models can differ in asynchronous FL. Analyzing how each study constructs the global/reference model, gives us a comprehensive understanding of how the local devices participate in the entire machine learning process. The research also provides us with a prominent consideration factor for the FL classification, which will be further discussed in Section 5.1.2.

Most of the DAG-based asynchronous FL projects construct reference models by local tip selection and build global models via an external method such as a smart contract, a side chain, or a third party. For example, in PermiDAG (Lu et al., 2020), the reference model is continuously constructed via the local aggregation of a DAG–based subchain, while the global aggregation happens synchronously by a mainchain based on the permissioned setting. In ChainsFL (Yuan, Cao, Peng, & Sun, 2021) (see full version (Yuan, Cao, Sun, & Peng, 2021)), the smart contract published by a FL task requester keeps track of the global model by aggregating tips with the highest accuracy among some randomly chosen tips via the FedAvg algorithm, whereas the reference model is constructed in the DAG-based mainchain by miners of the subchain. DAG-FL (Cao et al., 2021) and TangleFL (Schmid et al., 2020) build the reference model via the DAG tip selection algorithm by local/edge clients, who train their own local models for better performance based on that model. Then, the global model is determined based on the performance by choosing sub-DAG, which has been generated by every local aggregation of the client. As an example, DAG-FL utilizes smart contracts to determine the global model.

In comparison with AFL-DAG, a blockchain based asynchronous FL project named blockchained-based asynchronous federated learning (BAFL) (Feng et al., 2021) proposes an architecture in which clients download reference models from miners and construct the global model by recording the best reference model into the actual block. On the other hand, BlockFL (Kim et al., 2020) proposes a synchronous blockchained-FL where every model aggregation step must be carried out after the global model update.

In short, it is summarized as follows:

- Most studies do not have precise descriptions about the provisional models generated due to the property of asynchronous FL. Therefore, a detailed discussion about the differences between the model referred to by individual devices for each local training and the model regarded as a result of the training is required.
- In general, AFL-DAG systems construct their reference models by the predefined tip selection algorithm, whereas the global model is built by introducing an external trusted method such as smart contracts or another side chain layer. In contrast, blockchained-FL generates both the reference and global models together via block generation with different consensus models.

5.1.2. FL classification

It is challenging to strictly divide FL into asynchronous FL and synchronous FL due to the heterogeneous system architecture and varied terminology of the global/reference model that is used in the literature. Therefore, we propose a classification method that can be applied horizontally to classify most of the representative AFL-DAG and blockchained-FL research into four types of FL in Section 3: (1) asynchronous, (2) pseudo asynchronous, (3) hybrid asynchronous, and (4) synchronous. Through our observation, the classification of FL in terms of asynchrony is an important factor to consider while designing AFL-DAG, which impacts not only the performance of the trained model but also the entire system functionality.

DAG-FL (Cao et al., 2021) and TangleFL (Schmid et al., 2020) are classified as a fully asynchronous FL in our framework. They have a different reference model and global model, and the number of aggregations is the same as the number of local updates because both utilize a DAG-based architecture for model aggregation and propagation. On the other hand, ChainsFL (Yuan, Cao, Sun, & Peng, 2021) belongs to

hybrid FL in our classification due to the multiple aggregation layers in its system design. In ChainsFL, DAG is used for the mainchain, and blockchain based on the Raft algorithm is used for the subchain (Yuan, Cao, Sun, & Peng, 2021). In contrast, PermiDAG (Lu et al., 2020) is classified as a fully asynchronous FL, which adopts a DAG-based subchain and a DPoS-based main chain. Therefore, the frequency of model aggregations is the same as the number of local updates owing to the DAG-based subchain.

As for blockchained-FL, block-FL (Kim et al., 2020) is a synchronous FL, because each client downloads the reference model from the tip height and synchronously aggregates the global model for the next tip height via miners. However, BAFL (Feng et al., 2021) is a pseudo asynchronous FL, which enables clients to download the reference model from miners whenever they want to reduce the waiting time for the aggregation.

In short, although some solutions (Feng et al., 2021) utilize PoW consensus and extra system design to be more asynchronous, most of the current solutions (Cao et al., 2021; Lu et al., 2020; Schmid et al., 2020; Yuan, Cao, Sun, & Peng, 2021) rely on the DAG architecture to implement asynchronous FL due to the consensus property. This is because the inherent design of the DAG-based ledger enables clients to aggregate whenever they submit a local update, which satisfies the rigid definition of a fully asynchronous FL described in Fig. 4.

5.1.3. Consensus

A consensus mechanism is a crucial consideration factor when introducing blockchain/DAG to FL. Each research study has adopted a consensus algorithm that is suitable for its own system among the various consensus types such as proof of work (PoW) (Nakamoto, 2008), proof of stake (PoS) (King & Nadal, 2012), delegated proof of stake (DPoS) (Larimer, 2014), Raft (Ongaro & Ousterhout, 2014), and DAG-based consensus algorithm (Popov, 2018). It would be necessary to clearly state which data is recorded in the block or transaction and how the optimal FL aggregation is achieved by the blockchain consensus algorithm, considering the inherent pros and cons and risk factors.

Most of the blockchained-FL and AFL-DAG studies have adopted the best machine learning model through the process of a consensus mechanism that selects the best one among accumulated blocks or transactions carrying parameters of the local trained model (Liu, Lan, Li, Miao, & Tian, 2021). When the process of the consensus mechanism is divided into a global and local consensus, one process can use a different consensus algorithm from the other.

As for the AFL-DAG studies, ChainsFL first determines the reference model synchronously through the Practical Byzantine Fault Tolerance (PBFT)-based consensus like Raft on the subchain and determines the global model asynchronously in the mainchain through the DAG-based algorithm (Yuan, Cao, Sun, & Peng, 2021). Similarly, but differently, PermiDAG utilizes the DAG-based consensus for choosing the reference model as a local consensus and DPoS for choosing the global model as a global consensus (Lu et al., 2020). On the other hand, TangleFL only uses the DAG-based consensus to decide on the global model among each transaction (Schmid et al., 2020). DAG-FL also selects the global model by the DAG-based voting mechanism (Cao et al., 2021). Regarding blockchained-FL, BAFL and BlockFL use the general PoW consensus for the mining of blocks containing the best global model (Feng et al., 2021).

To summarize, most of the studies have applied the existing consensus algorithms to their own system, such as PoW, PoS, DPoS, and DAG-based consensus. Consensus mechanisms have their own prerequisites for achieving consensus. For example, PoW assumes that no single identity can own more than half of the total hash power. We could accept the assumption because it is rarely possible to own more than 50 percent of the total hash power stemming from the decentralization characteristic of the blockchain. However, blockchain/DAG-based FL cannot guarantee that a network comprised of machine learning entities is decentralized enough. Therefore, it seems necessary for each FL model to discuss the process of reaching a consensus and the necessary prerequisites.

5.1.4. Security

As mentioned in Section 2, FL is susceptible to *poisoning* attacks, which can be classified largely into two categories: *untargeted* and *targeted* attacks. The goal of untargeted attacks (also known as *model degradation attacks*) is to merely disturb the training of the ML model and to degrade the overall model performance. In the asynchronous setting, we have the *transaction staleness* issue in which clients submit local gradients computed from an old (or stale) reference model due to a poor network environment or malicious intention.[2] We can regard submitting a stale gradient as a form of untargeted poisoning attack, even if it is unintentional. On the other hand, targeted attacks (also known as *backdoor attacks*) aim to impact a relatively small number of specific attacker-chosen inputs while not affecting the other inputs. It is critical to realize a secure AFL-DAG system that can defend against both the aforementioned attacks in the asynchronous FL and possible network attacks (such as a Sybil attack and an eclipse attack) in the decentralized FL.

As shown in Table 3, the prior AFL-DAG and blockchained-FL studies discuss how they mitigate the attack vectors in each system. Particularly, a transaction time limit is introduced to address the transaction staleness issue (Cao et al., 2021; Yuan, Cao, Sun, & Peng, 2021). DAG-FL introduces a max threshold of staleness during the tip selection algorithm to skip staled transactions (Cao et al., 2021). Similarly, ChainsFL proposes a virtual pruning that sets a waiting time, and the tips that were not selected during the freshness time can no longer be selected (Yuan, Cao, Sun, & Peng, 2021).

As for the model degradation attack, most of the AFL-DAG systems discuss that local model aggregation via the DAG tip selection algorithm is naturally effective to prune under-performing transactions, and thereby can mitigate such an attack. On the other hand, blockchained-FL architectures present that PoW consensus including the evaluation logic would somewhat help mitigate a model degradation attack. Regarding the backdoor attack, TangleFL applies multiple tip selection algorithms to mitigate such an attack, although it lacks experiments on its effectiveness (Schmid et al., 2020).

From the above analysis of existing literature, the following is highlighted:

- Most of the current AFL-DAG/blockchained-FL solutions do not cover network attacks. However, in the decentralized FL system, the security against different kinds of network attacks is a relatively paramount factor, unlike centralized FL. Therefore, defenses against those network attacks should be considered in the future work.
- More formal and broader studies on different attacks other than a poisoning attack and defenses against them are required in the framework of blockchained/DAG asynchronous FL. For instance, there is a class of inference attacks to consider (Section 3.1.2). Similarly, although some studies (Cao et al., 2021; Schmid et al., 2020) mention the backdoor attack in their paper, in-depth consideration against a backdoor attack is necessary for AFL-DAG.
- Regarding the transaction staleness issue, current solutions utilize a binary way to exclude staled transactions during the tip selection. To further circumvent the transaction staleness issue, a decay function in centralized asynchronous FL can be applied to enable a more flexible aggregation for staled transactions.

[2] The staleness issue can be seen as counterpart of the straggler issue of the synchronous FL.

Table 3
A security comparison of state-of-the-art blockchain/DAG based FLs.

	Against transaction staleness issue	Against model degradation attack	Against backdoor attack
PermiDAG (Lu et al., 2020)	N/A	Local aggregation by accuracy comparison	N/A
DAG-FL (Cao et al., 2021)	Setting a max threshold of staleness	Local model validation and transaction authentication	N/A
ChainsFL (Yuan, Cao, Sun, & Peng, 2021)	Setting a waiting time	Virtual pruning	N/A
TangleFL (Schmid et al., 2020)	N/A	Multiple tip selection algorithm	Multiple tip selection algorithm
BAFL (Feng et al., 2021)	N/A	Evaluation on validity of devices	N/A
BlockFL (Feng et al., 2021)	N/A	N/A	N/A

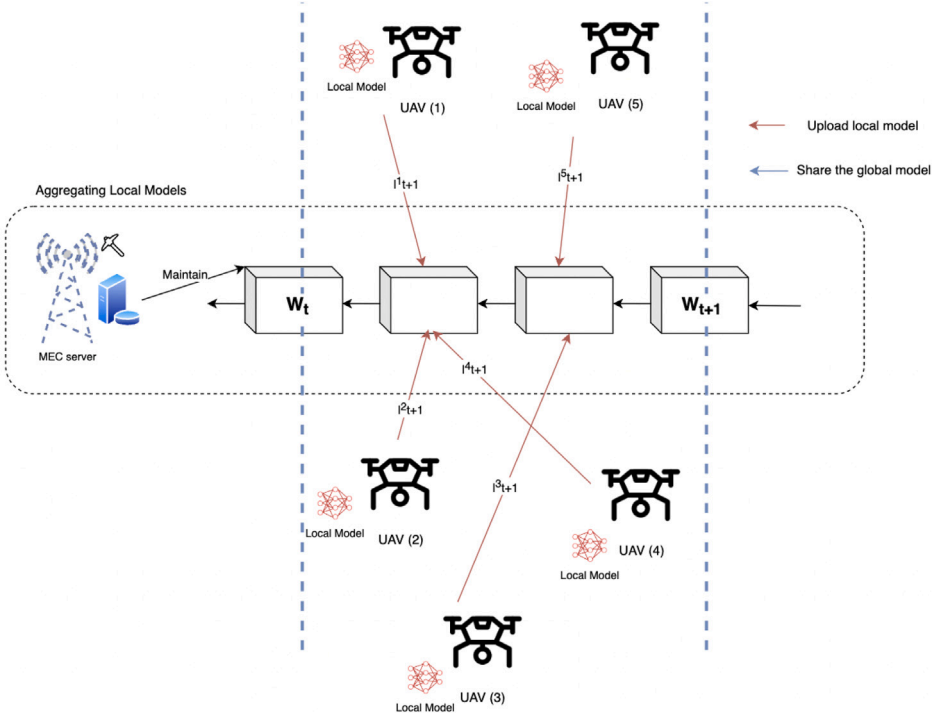

Fig. 7. Blockchain-based UAV crowdsourcing framework.

5.2. Application: On-device crowdsourcing system with AFL-DAG

In this section, we focus on the application of AFL-DAG in a popular domain, on-device crowdsourcing. In particular, we consider a mobile crowdsourcing system based on Unmanned Aerial Vehicles (UAVs) as a specific on-device crowdsourcing application (Zhao et al., 2021) to compare the AFL-DAG based crowdsourcing FL framework with the blockchain based crowdsourcing FL framework.

The UAV crowdsourcing system is a ML system that generates a global model aggregated from the local models trained by each UAV based on the collected data by itself. To aggregate local models, traditional AI/ML systems have introduced a single server with the authority to access to each UAV for gathering data and training models. However, the single server architecture is not optimal when the device network is large-scale and time-flexible. Additionally, the authorized central server can tamper with or maliciously use the information of individual UAVs. FL is an alternative used to enable highly scalable UAV crowdsourcing and can provide the improved security standards for private data usage by integration with a blockchain framework (Zhou et al., 2018). Wang, Su, Zhang, and Benslimane (2020) proposes a mobile UAV crowdsensing system on FLchain (Federated learning integrated with blockchain). In this system, the blockchain is utilized as a ledger to record the data training history or model exchanges among the UAVs and ensures the security for data tampering.

The UAV crowdsourcing framework based on FLchain includes a number of aerial vehicles carrying out ML tasks with the data gathered by themselves and a central coordinating server (MEC server) for the model aggregation (Pandey et al., 2020; Wang et al., 2020). When the crowdsourcer referred to as the MEC server wants to build a global model w, the MEC server shares the initial parameters w_0 to start the local training across the distributed vehicle clients. Local client k trains the local model parameters l^k to minimize the local loss function f^k and sends it back to the MEC server. The MEC server then updates the global model weights by aggregating the local trained weights and shares the updated weights with the local clients. The process continues iteratively until the global MEC server detects convergence of the global model.

Algorithm 1 Blockchain-based UAVs crowdsourcing framework

Input: Initialized weights w_0
Share initial weights with local clients $\forall k \in K$
while Model convergence **do**
  for $k \in K$ **do**
    Train local model to minimize their-own local loss function $f^k(l^k_{t-1})$ in parallel given the global model weight w_{t-1}
    Update local weights l^k_t and send back to MEC server
  end for
  Aggregation l^k_t for $k = 1, \cdots, K$ to update the global model
  Share the global model weight w_t
end while

Fig. 7 describes the UAV crowdsourcing FL framework based on the blockchain. MEC servers mine blocks and record the global model at every iteration to protect the model from tampering. Algorithm 1 summarizes the process of the blockchain-based crowdsourcing framework. Each client k has its local database D^k of the data size $|D^k|$, respectively, and trains their local loss function $f^k(l_t^k)$ at iteration t. Then, a central server aggregates every local models after each local training iteration. This step is important in determining the speed of the entire learning process, because an individual local machine cannot start the next iteration before this step is completed. This shows that the physical network of local UAVs is decentralized, but the process of global model aggregation is still somewhat centralized and can be classified as the hybrid or pseudo-asynchronous FL in the proposed classification framework. Blockchain-based frameworks have difficulty providing the advantages of a physically distributed network of local machines to asynchronous FL process.

Algorithm 2 DAG-based UAVs crowdsourcing framework

1: **Input:** Initialized weights w_0
2: MEC server shares initial weights with local UAVs $\forall k \in K$ and uploads the initial weights to the global DAG-based blockchain.
3: Every local UAV uploads the initial weights to their own DAG-based blockchain.
4: **while** Model convergence in the global DAG-based blockchain **do**
5: **for** $k \in K$ **do**
6: Train local model in parallel given the global model weight w_{t-1}.
7: Update local weights l_t^k
8: Save the local weights l_t^k on its own DAG-based blockchain and send back to MEC server.
9: MEC server update the global DAG-based blockchain asynchronously.
10: Local UAV k update its own DAG-based blockchain generated by selected local model parameters from the available recent global DAG-based blockchain.
11: **end for**
12: **end while**

Now, we describe the UAV crowdsourcing framework based on the DAG ledger. In this algorithm, we eliminate the steps of global model aggregation and distribution of MEC servers. Koo, Faseeh Qureshi, Siddiqui, Abbas, and Bashir (2020) shows that DAG structure provides an efficient data stream solution under the IoT industry. Therefore, all local clients maintain their own local models by utilizing DAG-based blockchains and update their owns after each learning process ends, which is the main difference from blockchain-based framework. MEC server maintains a global DAG-based blockchain that integrates the DAG-based blockchain of all local clients. In other words, sub DAG-based blockchains by local clients are a subset of global DAG-based blockchain. Therefore, each local client can refer to the central blockchain at that point in time at the end of each training session to update their own model without waiting for the next aggregation.

In the DAG-based UAV crowdsourcing framework, a MEC server irregularly gathers local model parameters whenever each UAV finishes its-own training step and transports the results to MEC server. In addition, each local UAV requires a global model parameter irregularly before it starts its new learning process. Each local client maintains its own sub-DAG, a database of the model parameters, including several models selected by itself without having to wait for the transmission of the global model parameters. Therefore, it can help the entire process eliminate the step of sharing the global model weights w_{t+1} at every iteration t. FedLin guarantees linear convergence to the optimal global model without assuming that the loss functions of local clients are similar (Mitra, Jaafar, Pappas, & Hassani, 2021). It also enables the heterogeneity of the system or objective, increasing the degree of freedom of the learning process of local clients (UAVs) and lowering the frequency of communication between the server and UAVs. We introduced FedLin algorithm to our DAG-based blockchain to take advantage of these features of the FedLin algorithm.

Algorithm 2 shows asynchronous FL to utilize the DAG-based blockchain for the UAV crowdsourcing framework. Fig. 8 illustrates the structure of the DAG-based UAV crowdsourcing framework. For example, UAV_1 maintains its-own sub-DAG that includes only model l_t^2 immediately before the training step l_t^1. After the training step, UAV_1 adds its next model l_t^1 to its-own sub-DAG. When the MEC server wants to create the next global model w_t, it aggregates the local models from a global DAG-based ledger including l_t^1, l_t^2, l_t^3, and l_t^4.

5.3. Comparative evaluation of AFL-DAG

As a summary, we compare the AFL-DAG with classic FL and blockchained-FL in edge computing as shown in Table 4. In this work, classic FL indicates both centralized and synchronous FL, such as Google FL (Konecný, McMahan, Ramage, & Richtárik, 2016; McMahan et al., 2017).

In the edge computing environment, the straggler problem can easily hinder the entire performance due to the heterogeneous capacities of devices and dynamic network circumstances. To address such an issue, AFL-DAG surpasses blockchained-FL and classic FL. Unlike blockchain in which a miner always needs to wait to aggregate the local model, AFL-DAG enables sequential model aggregation without any waiting process. By utilizing the DAG data structure, the AFL-DAG node can execute local training regardless of the global model aggregation, which mitigates the straggler problem eventually. Furthermore, AFL-DAG does not have a deterministic global model. In the blockchain architecture, the global model is determined by a certain consensus mechanism as a tip block. However, DAG nodes can have different tips, so that the global model is not determined as a single state at the same time. With this regard, AFL-DAG enables a more flexible global model construction, which could mitigate model degradation attack and be less affected by malicious nodes especially in untargeted attacks.

As for incentive learning, unlike classic FL, AFL-DAG and blockchained-FL can integrate a decentralized incentive mechanism with FL, which provides natural motivation for the network entities to participate in the model training. Most studies determine the global model through the blockchain consensus mechanism by assuming that their tip state results in a foremost global model. However, the outcome of the blockchain consensus may not be the same with the global model of highest accuracy from the viewpoint of FL. In general, in blockchain systems, miners are only concerned that transactions in a candidate block achieve consistency with the transactions of historical blocks. FL aims to create the best model that is not overfitted and possible to infer adequate results for new datasets. Therefore, consensus in AFL-DAG and blockchained-FL should include an additional incentive mechanism for miners or network entities to select a better performing model in FL and distribute rewards based on their contribution. However, most AFL-DAG studies do not address this aspect because DAG-based consensus does not have explicit miners and an incentive mechanism in the protocol layer. Therefore, the incentive mechanism, to assure that network entities choose the global model of highest quality, should be considered in future research to enable practical AFL-DAG applications.

Moreover, because of their decentralized property, AFL-DAG and blockchained-FL do not rely on an aggregation server and therefore are less vulnerable to the single-point-failure problem. In contrast, classic FL is dependent on the centralized aggregation server, which can be a scalability issue in the case of a high volume of transactions, especially in the IoT industry. On the other hand, there is a huge research gap on privacy concerns in blockchained-FL and AFL-DAG applications. Even with decentralized training, it could be more vulnerable by submitting the model parameter as a transaction to the peer node in terms of privacy.

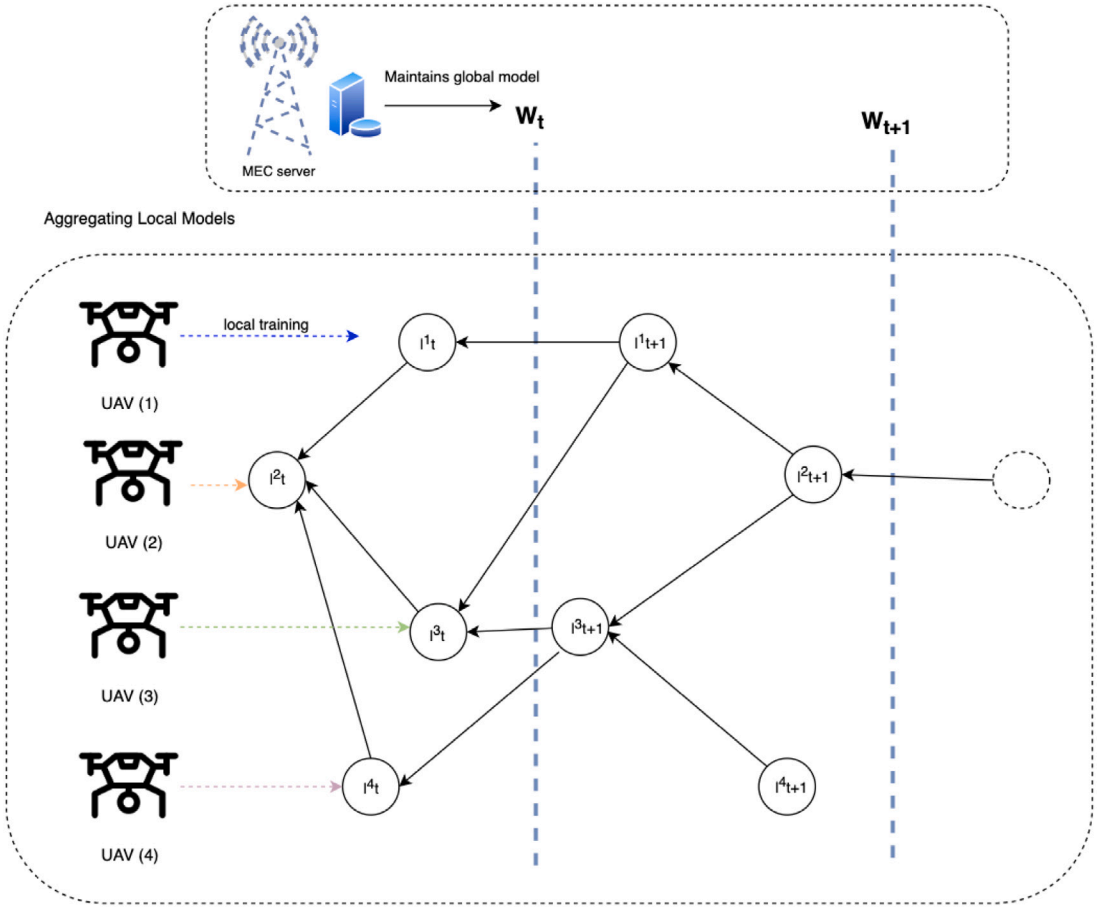

Fig. 8. DAG-based UAV crowdsourcing framework.

Table 4
A comparison between classic FL, blockchained-FL, and AFL-DAG.

	Classic FL	Blockchained-FL	AFL-DAG
Straggler problem	Vulnerable	Vulnerable	Less Vulnerable
Deterministic global model	Yes	Yes	No
Incentive learning	Centralized	Decentralized	Decentralized
Single-point-failure	Vulnerable	Less Vulnerable	Less Vulnerable
Privacy concern	Dependent on parameter server	Vulnerable	Vulnerable
Operation condition	N/A	It is assumed that miners would choose optimal model parameter.	It is assumed that coordinators who store a global DAG exist, and tip selection algorithm would pick the optimal model parameter.

6. Conclusion

This article mainly dealt with a survey of asynchronous FL integrated with a DAG-based ledger as a promising edge computing technology. We first proposed a concise terminology of global/reference model for an asynchronous FL definition and presented a general method to classify each type of FL in terms of asynchrony. Second, we elaborated on the reasons why a DAG-based ledger is suitable for asynchronous FL applications in an edge computing network and presented a generic system model of AFL-DAG based on MEC architecture. Third, we discussed the four main design aspects based on state-of-the-art AFL-DAG research and summarized the key solutions and lessons learned from the prior studies in each aspect. Finally, we analyzed the application of AFL-DAG in a popular edge computing domain, on-device crowdsourcing, and provided the high-level comparison of AFL-DAG with blockchained-FL. A discussion on several challenges and future directions to realize the MEC applications empowered by AFL-DAG is also highlighted.

Lastly, we provide some good starting points for discussion and further research as follows.

- We found that the research gap on how to decentralize the process of FL for providing secure and fair access to participants remains. Most related studies assume that the MEC server and local participants do not have equal authority to the learning process, regardless of which structure is utilized for the AFL in edge computing.
- How to handle data network transmission throughout the FL process should be considered for implementing AFL-DAG on the peer-to-peer network. The existing literature dealing with data transmission on the peer-to-peer network could be transformed for carrying out further research on transporting model parameter data securely for AFL-DAG.
- From existing studies related to AFL-DAG, there was not much research that discusses the scalability issue while model training

and aggregation when the number of devices increases. The performance of model training and whether it converges should be discussed as a future direction of AFL-DAG when the number of devices and transactions increases.

CRediT authorship contribution statement

Seoyoung Ko: Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Keewoo Lee:** Conceptualization, Methodology, Investigation, Writing – original draft. **Hyunhum Cho:** Investigation, Writing – original draft. **Yoonjae Hwang:** Visualization, Writing – original draft. **Huisu Jang:** Conceptualization, Supervision, Writing – review & editing, Funding acquisition, Formal analysis, Visualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: HUISU JANG reports financial support was provided by National research Foundation of Korea.

Data availability

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

Acknowledgments

This work was supported by the National Research Foundation of Korea Grant through the Korean Government (MEST) under NRF-2022R1F1A1074008.

References

Ali, M., Karimipour, H., & Tariq, M. (2021). Integration of blockchain and federated learning for internet of things: Recent advances and future challenges. *Computers & Security*, Article 102355.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., & Shmatikov, V. (2020). How to backdoor federated learning. In *International conference on artificial intelligence and statistics* (pp. 2938–2948). PMLR.

Baird, L. (2016). *The swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance, vol. 34*: *Swirlds tech reports SWIRLDS-TR-2016-01, Tech. rep.*.

Cao, M., Zhang, L., & Cao, B. (2021). Towards on-device federated learning: A direct acyclic graph-based blockchain approach. arXiv preprint arXiv:2104.13092.

Chai, Z., Chen, Y., Anwar, A., Zhao, L., Cheng, Y., & Rangwala, H. (2021). FedAT: a high-performance and communication-efficient federated learning system with asynchronous tiers. In *Proceedings of the international conference for high performance computing, networking, storage and analysis* (pp. 1–16).

Chen, Y., Ning, Y., Slawski, M., & Rangwala, H. (2020). Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE international conference on big data* (pp. 15–24). IEEE.

Cheu, A., Smith, A., Ullman, J., Zeber, D., & Zhilyaev, M. (2019). Distributed differential privacy via shuffling. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 375–403). Springer.

Feng, L., Zhao, Y., Guo, S., Qiu, X., Li, W., & Yu, P. (2021). Blockchain-based asynchronous federated learning for internet of things. *IEEE Transactions on Computers*.

Fereidooni, H., Marchal, S., Miettinen, M., Mirhoseini, A., Möllering, H., Nguyen, T. D., et al. (2021). SAFELearn: secure aggregation for private federated learning. In *2021 IEEE security and privacy workshops* (pp. 56–62). IEEE.

Hao, J., Zhao, Y., & Zhang, J. (2020). Time efficient federated learning with semi-asynchronous communication. In *2020 IEEE 26th international conference on parallel and distributed systems* (pp. 156–163). IEEE.

Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). *Mobile edge computing—A key technology towards 5G, vol. 11, no. 11*: *ETSI white paper*, (pp. 1–16).

Kairouz, P., Liu, Z., & Steinke, T. (2021). The distributed discrete Gaussian mechanism for federated learning with secure aggregation. arXiv preprint arXiv:2102.06387.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., et al. (2019). Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., & Smith, A. (2011). What can we learn privately? *SIAM Journal on Computing, 40*(3), 793–826.

Kim, H., Park, J., Bennis, M., & Kim, S.-L. (2020). Blockchained on-device federated learning. *IEEE Communications Letters*, *24*(6), 1279–1283. http://dx.doi.org/10.1109/LCOMM.2019.2921755.

King, S., & Nadal, S. (2012). *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, vol. 19, no. 1*: *Self-published paper, August*.

Konecný, J., McMahan, H. B., Ramage, D., & Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. arxiv abs/1610.02527.

Koo, J., Faseeh Qureshi, N. M., Siddiqui, I. F., Abbas, A., & Bashir, A. K. (2020). IoT-enabled directed acyclic graph in spark cluster. *Journal of Cloud Computing*, *9*(1), 1–15.

Larimer, D. (2014). *Delegated proof-of-stake (dpos), vol. 81*: *Bitshare whitepaper*, (p. 85).

Lee, J.-w., Oh, J., Shin, Y., Lee, J.-G., & Yoon, S.-Y. (2020). Accurate and fast federated learning via IID and communication-aware grouping. arXiv preprint arXiv:2012.04857.

Lewenberg, Y., Sompolinsky, Y., & Zohar, A. (2015). Inclusive block chain protocols. In *International conference on financial cryptography and data security* (pp. 528–547). Springer.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., et al. (2020). Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *22*(3), 2031–2063.

Liu, Y., Lan, Y., Li, B., Miao, C., & Tian, Z. (2021). Proof of learning (PoLe): empowering neural network training with consensus building on blockchains. *Computer Networks*, *201*, Article 108594.

Liu, Y., Yu, W., Ai, Z., Xu, G., Zhao, L., & Tian, Z. (2022). A blockchain-empowered federated learning in healthcare-based cyber physical systems. *IEEE Transactions on Network Science and Engineering*.

Liu, Y., Yuan, X., Xiong, Z., Kang, J., Wang, X., & Niyato, D. (2020). Federated learning for 6G communications: Challenges, methods, and future directions. *China Communications*, *17*(9), 105–118. http://dx.doi.org/10.23919/JCC.2020.09.009.

Lu, Y., Huang, X., Zhang, K., Maharjan, S., & Zhang, Y. (2020). Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles. *IEEE Transactions on Vehicular Technology*, *69*(4), 4298–4311.

Majeed, U., & Hong, C. S. (2019). FLchain: Federated learning via MEC-enabled blockchain network. In *2019 20th Asia-Pacific network operations and management symposium* (pp. 1–4). IEEE.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282). PMLR.

Melis, L., Song, C., Cristofaro, E. D., & Shmatikov, V. (2018). Exploiting unintended feature leakage in collaborative learning. arXiv:1805.04049.

Mitra, A., Jaafar, R., Pappas, G. J., & Hassani, H. (2021). Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. *Advances in Neural Information Processing Systems*, *34*, 14606–14619.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.

Nasr, M., Shokri, R., & Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy*. IEEE, http://dx.doi.org/10.1109/sp.2019.00065.

Nguyen, D. C., Ding, M., Pham, Q.-V., Pathirana, P. N., Le, L. B., Seneviratne, A., et al. (2021). Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet of Things Journal*, *8*(16), 12806–12825. http://dx.doi.org/10.1109/JIOT.2021.3072611.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Esmaeili, M. M., et al. (2021). Federated learning with buffered asynchronous aggregation. arXiv preprint arXiv:2106.06639.

Nguyen, T. D., Rieger, P., Miettinen, M., & Sadeghi, A.-R. (2020). Poisoning attacks on federated learning-based IoT intrusion detection system. In *Proc. Workshop Decentralized IoT Syst. Secur.* (pp. 1–7).

Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *2014 {USENIX} annual technical conference ({USENIX}{ATC} 14)* (pp. 305–319).

Pandey, S. R., Tran, N. H., Bennis, M., Tun, Y. K., Manzoor, A., & Hong, C. S. (2020). A crowdsourcing framework for on-device federated learning. *IEEE Transactions on Wireless Communication*, *19*(5), 3241–3256.

Poon, J., & Dryja, T. (2015). The bitcoin lightning network.

Popov, S. (2018). *The tangle, vol. 1, no. 3: White paper*.

Salem, A., Bhattacharya, A., Backes, M., Fritz, M., & Zhang, Y. (2019). Updates-leak: Data set inference and reconstruction attacks in online learning. arXiv:1904.01067.

Sav, S., Pyrgelis, A., Troncoso-Pastoriza, J. R., Froelicher, D., Bossuat, J.-P., Sousa, J. S., et al. (2020). POSEIDON: Privacy-preserving federated neural network learning. arXiv preprint arXiv:2009.00349.

Schmid, R., Pfitzner, B., Beilharz, J., Arnrich, B., & Polze, A. (2020). Tangle ledger for decentralized learning. In *2020 IEEE international parallel and distributed processing symposium workshops* (pp. 852–859). IEEE.

Shi, G., Li, L., Wang, J., Chen, W., Ye, K., & Xu, C. (2020). HySync: Hybrid federated learning with effective synchronization. In *2020 IEEE 22nd international conference on high performance computing and communications; IEEE 18th international conference on smart city; IEEE 6th international conference on data science and systems* (pp. 628–633). IEEE.

So, J., Ali, R. E., Güler, B., & Avestimehr, A. S. (2021). Secure aggregation for buffered asynchronous federated learning. arXiv preprint arXiv:2110.02177.

Sompolinsky, Y., & Zohar, A. (2013). Accelerating bitcoin's transaction processing. In *Fast money grows on trees, not chains*.

Stripelis, D., & Ambite, J. L. (2021). Semi-synchronous federated learning. arXiv preprint arXiv:2102.02849.

Sun, W., Lei, S., Wang, L., Liu, Z., & Zhang, Y. (2020). Adaptive federated learning and digital twin for industrial internet of things. *IEEE Transactions on Industrial Informatics*, *17*(8), 5605–5614.

Unal, D., Hammoudeh, M., Khan, M. A., Abuarqoub, A., Epiphaniou, G., & Hamila, R. (2021). Integration of federated machine learning and blockchain for the provision of secure big data analytics for Internet of Things. *Computers & Security*, *109*, Article 102393.

Wang, Q., Li, Q., Wang, K., Wang, H., & Zeng, P. (2021). Efficient federated learning for fault diagnosis in industrial cloud-edge computing. *Computing*, *103*(10), 2319–2337.

Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., & Qi, H. (2019). Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications* (pp. 2512–2520). IEEE.

Wang, Y., Su, Z., Zhang, N., & Benslimane, A. (2020). Learning in the air: Secure federated learning for UAV-assisted crowdsensing. *IEEE Transactions on Network Science and Engineering*, *8*(2), 1055–1069.

Wu, W., He, L., Lin, W., Mao, R., Maple, C., & Jarvis, S. (2020). Safa: a semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, *70*(5), 655–668.

Wu, W., He, L., Lin, W., Mao, R., Maple, C., & Jarvis, S. (2021). SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, *70*(5), 655–668. http://dx.doi.org/10.1109/TC.2020.2994391.

Xia, W., Li, Y., Zhang, L., Wu, Z., & Yuan, X. (2021). A vertical federated learning framework for horizontally partitioned labels. arXiv preprint arXiv:2106.10056.

Xie, C., Huang, K., Chen, P.-Y., & Li, B. (2019). Dba: Distributed backdoor attacks against federated learning. In *International conference on learning representations*.

Xie, C., Koyejo, S., & Gupta, I. (2019). Asynchronous federated optimization. arXiv preprint arXiv:1903.03934.

Xu, C., Qu, Y., Xiang, Y., & Gao, L. (2021). Asynchronous federated learning on heterogeneous devices: A survey. arXiv preprint arXiv:2109.04269.

Yuan, S., Cao, B., Peng, M., & Sun, Y. (2021). ChainsFL: Blockchain-driven federated learning from design to realization. In *2021 IEEE wireless communications and networking conference* (pp. 1–6). IEEE.

Yuan, S., Cao, B., Sun, Y., & Peng, M. (2021). Secure and efficient federated learning through layering and sharding blockchain. arXiv preprint arXiv:2104.13130.

Zhang, Y., Duan, M., Liu, D., Li, L., Ren, A., Chen, X., et al. (2021). CSAFL: A clustered semi-asynchronous federated learning framework. In *2021 international joint conference on neural networks* (pp. 1–10). IEEE.

Zhao, L., Saif, M. B., Hawbani, A., Min, G., Peng, S., & Lin, N. (2021). A novel improved artificial bee colony and blockchain-based secure clustering routing scheme for FANET. *China Communications*, *18*(7), 103–116.

Zhao, L., & Yu, J. (2019). Evaluating DAG-based blockchains for IoT. In *2019 18th IEEE international conference on trust, security and privacy in computing and communications/13th IEEE international conference on big data science and engineering* (pp. 507–513). IEEE.

Zhou, Z., Liao, H., Gu, B., Huq, K. M. S., Mumtaz, S., & Rodriguez, J. (2018). Robust mobile crowd sensing: When deep learning meets edge computing. *IEEE Network*, *32*(4), 54–60.

Zhu, L., & Han, S. (2020). Deep leakage from gradients. In *Federated learning* (pp. 17–31). Springer.