DAG-FL: Direct Acyclic Graph-based Blockchain Empowers On-Device Federated Learning

Mingrui Cao[1], Bin Cao[1], Wei Hong[2], Zhongyuan Zhao[1], Xiang Bai[3], and Lei Zhang[4]
[1]State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, 100876, China
[2]Beijing Xiaomi Mobile Software, Beijing, 100085, China
[3]China Electronic Technology Cyber Security Co., Ltd., Chengdu, 610041, China
[4]James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, U.K.

Abstract—**Due to the distributed characteristics of Federated Learning (FL), the vulnerability of global model and coordination of devices are the main obstacle. As a promising solution of decentralization, scalability and security, leveraging blockchain in FL has attracted much attention in recent years. However, the traditional consensus mechanisms designed for blockchain like Proof of Work (PoW) would cause extreme resource consumption, which reduces the efficiency of FL greatly, especially when the participating devices are wireless and resource-limited. In order to address device asynchrony and anomaly detection in FL while avoiding the extra resource consumption caused by blockchain, this paper introduces a framework for empowering FL using Direct Acyclic Graph (DAG)-based blockchain systematically (DAG-FL). Accordingly, DAG-FL is first introduced from a three-layer architecture in details, and then two algorithms *DAG-FL Controlling* and *DAG-FL Updating* are designed running on different nodes to elaborate the operation of DAG-FL consensus mechanism. The extensive simulations show that DAG-FL can achieve the better performance in terms of training efficiency and model accuracy compared with the typical existing on-device federated learning systems as the benchmarks.**

I. Introduction

In order to solve the data island problem caused by privacy and make the best use of distributed data on various devices, Federated Learning (FL) has recently drawn much attention. It is a distributed machine learning framework, and participants in FL transfer and communicate the model parameters without revealing user privacy to establish machine learning models [1, 2]. For wireless scenarios, on-device FL is one of the most typical application where participating nodes of FL are numerous mobile devices under a wireless network [3, 4]. Meanwhile, with the advent of 5G era, mobile devices would have sufficient communication bandwidth which makes it possible to establish an efficient FL system on mobile devices.

Although FL is widely considered to be a feasible way to enhance privacy and security in 5G wireless networks, it still faces many challenges during deployment [5]. The main two points are as follows.

- ***Device asynchrony***: Various nodes have different resources for FL in terms of computing, communication, caching, battery power, data, and training time, which would result in heterogeneity and it is a natural characteristic especially in wireless networks [5]. As a result, due to the limited capacity and ability of node, network and system, it is hard to coordinate FL process perfectly generating the device asynchrony. To this end, the traditional centralized and synchronous FL system like Google FL proposed in 2017 [1], a single node must wait for other nodes to complete their tasks and then enter the next iteration together after completing its own training task. However, this manner might generate the deteriorated cost incurred by the bottleneck node obviously, in which the worst case is that if a node shuts down during training, it may let an iteration of FL be invalid completely [6, 7].
- ***Anomaly detection***: Due to privacy concerns, the local data set and local operation process of a node are invisible by others, which makes FL suffer from abnormal actions of nodes easily. Especially considering massive participating nodes, the challenge of FL is to detect abnormal nodes and avoid adverse effects as much as possible. Abnormal nodes will reduce the overall efficiency of FL system as well as model accuracy by uploading abnormal parameters during the FL process, the machine learning model built by all nodes together would be very vulnerable, and thus anomaly detection in FL is necessary [8, 9].

To this end, some researches focus on asynchronous FL framework to solve device asynchrony, and anomaly detection strategy to mitigate the impact of abnormal node in FL system. Recently, considering the asynchronous system and security protection, blockchain becomes a nature design adopted in FL [10], the reasons are twofold. 1) Nodes can announce local model immediately without any asynchrony requirement. 2) Blockchain miners can collect and validate model parameters to encourage normal action while avoiding anomaly.

Although recent blockchained FL work [3, 11] have achieved some progress and advantages to address the mentioned challenges, there are still some problems that have not been investigated thoroughly. First, these work basically follow the synchronous framework of Google FL, in which the innovation is to use miners to replace the central servers. However, due to the synchrony, the mobile device acted as a miner should be associated with each other, which obeys the decentralization of blockchain while declining the performance of FL in terms of delay, convergence, accuracy and etc. Second, to maintain the blockchain operation, most work use PoW [12] which allows node acted as a miner consuming an amount of computing resource for consensus achievement. Meanwhile, in order to detect abnormal nodes, the miner should also verify the correctness of uploaded model parameters. As a result, the training/learning efficiency might not meet the expectation well caused by the blockchain cost since the overall resource is limited, especially for on-device FL under wireless networks.

Through the above observations, our concerned issue is whether blockchain is available to establish an efficient asynchronous on-device FL system without introducing too much extra resource consumption. Owing to the evolution of consensus mechanism, we notice that Directed Acyclic Graph (DAG) ledger technology promotes blockchain from synchronous to asynchronous bookkeeping without miner nor mining [13]. Inspired by these, we propose a DAG empowered asynchronous FL for purpose of efficiency and immunity, referred as DAG-FL. Specifically, with an asynchronous architecture, DAG-FL can well meet the device asynchrony and allow nodes participating in FL iterations without considering the state of others. Meanwhile, due to the voting consensus mechanism of DAG ledger technology, the workload of model validation is allocated to every node in DAG-FL, enabling anomaly detection and the immunity to abnormal nodes.

The rest of this paper is organized as follows. Section II provides some basic concepts in DAG-FL. Section III introduces DAG-FL in terms of architecture, consensus mechanism, and FL algorithm. Furthermore, Section IV eleborates the operation of DAG-FL. Then, Section V analyzes the performance of DAG-FL in the case of large-scale nodes through simulation experiments. Finally, Section VI gives a summary.

II. Fundamentals

In order to elaborate the proposed DAG-FL, the basic concepts of FL and DAG leger technology involved in DAG-FL are introduced briefly in this section.

A. Federated Learning

Traditional synchronous FL like Google FL is composed of a central server and numerous nodes, where the global model is maintained on central server and FL iterations is performed on nodes. To complete an FL iteration, the central server first selects several idle nodes to assign FL tasks, and the assigned nodes download the current global model from the central server. When the central server collects all local models trained by assigned nodes, it runs the *FederatedAveraging* algorithm [1] that combines all the local models on averaging to get an aggregated new global model. In fact, due to device asynchrony [5], synchronous FL cannot fit the on-device FL scenarios well, and thus asynchronous FL has been studied recently. Nodes in asynchronous FL can download the global model from the central server at any time, and update it immediately whenever it has trained a local model [6].

B. Blockchain and DAG technology

DAG-based blockchain is proposed to promote the synchronous blockchain to asynchronous bookkeeping. The principle of DAG-based blockchain is to attach the new transactions in a forking topology [13] without maintaining a main chain, and thus any new arrival transaction can be recorded in blockchain immediately without any coordination [14]. The consensus used in DAG-based blockchain can be treated as a voting mechanism, which requires nodes to validate and approve some early published transactions before publishing their owns. Transactions that are newly published and not approved yet on DAG are usually called as tips [15]. To publish a transaction on DAG, one node needs to go through three stages. In the first stage, the node selects some tips according to some algorithms or just randomly. In the second stage, the node validates the authentication and correctness of selected tips. In the final stage, a new transaction, composed of essential information and approvals to selected tips, is constructed and published on DAG. Through these three stages, the votes are stored in the published transactions, and unidirectional connections among transactions are built forming the DAG architecture through the approval relationship.

III. DAG-FL Overview

In this section, an overview of DAG-FL is provided in terms of three aspects which are architecture, consensus mechanism, and FL algorithm.

A. Asynchronous Architecture

DAG-FL is proposed as a decentralized asynchronous FL system, including application, DAG and Fl layers from top to down. To well elaborate, this hierarchical structure of DAG-FL is shown in Fig. 1, in which two mobile device nodes and an external agent are involved.

1) FL Layer: FL layer is the bottom layer to provide FL function. In order to obtain local models recorded as transactions on DAG, FL layer allows any participating node to use its own data to train the global model. The global model is formed by using *FederatedAveraging* algorithm to aggregate local models stored in transactions on DAG. After training the global model, a new trained local model would be processed and published as a transaction on DAG.

2) DAG Layer: In DAG layer, each node maintains a local DAG, where the transaction contains authentication information, local model parameters, and the approval connections. The local DAG can be updated by broadcasting or inquiring through wireless network using P2P technology, and thus the new transaction (or say the new model) can be spread throughout the DAG-FL network finally.

3) Application Layer: Application layer is deployed on the top of DAG-FL, which provides the interface to external agents by running smart contracts. Through the smart contract, external agents can release FL task to nodes, observe FL process, and obtain the target model as soon as FL is completed. When a specific FL task is released according to a smart contract, nodes in DAG layer can participate in this

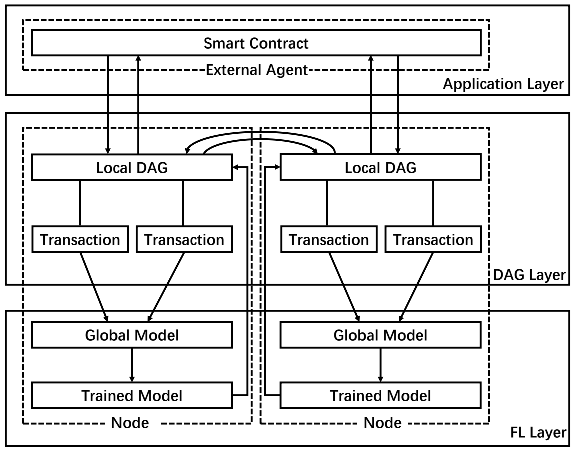

Fig. 1: Architecture of DAG-FL.

work based on an incentive mechanism to gain an amount of reward. And then, during FL process, the smart contract will observe transactions on DAG to determine whether a target model has been published.

Accordingly, the application layer provides an interface for external agents to deploy DAG-FL easily, the DAG and FL layer form an asynchronous platform for FL. Nodes with device asynchrony like smart phones and IoT devices in DAG-FL all need to maintain a local DAG to record transactions published by every node. The local DAG on each node is updated by communicating with adjacent nodes periodically using the P2P technology in blockchain field, and thus newly published transactions can be seen by all nodes in time. As there is no central server in DAG-FL, one node in DAG-FL constructs a global model from its local DAG to iterate FL instead of requiring a global model from the central server. This feature promises that nodes in DAG-FL can immediately participate in an iteration of FL whenever they are in idle state. When the node completes an iteration of FL and gets a new trained local model, the new local model can be published on its local DAG as a transaction immediately, and latter the new published transaction would be seen by all other nodes. In this manner, the operation of any node cannot affect the state of other nodes, which would satisfy the asynchrony of mobile devices.

B. Consensus based Anomaly Detection

We propose a DAG-FL consensus to keep the asynchronous FL platform stable, and present an effective way for anomaly detection in DAG-FL.

Traditional blockchained FLs often use miners to update the blockchain by running the consensus mechanism like PoW, enabling every node to observe transactions published on blockchain. Similarly, as there are no miners in DAG-FL, each node in DAG-FL should both perform FL tasks and update the DAG by running the DAG-FL consensus. Based on the voting mechanism of DAG ledger technology, DAG-FL consensus approves the nodes by validating both the authentication and local model correctness of tips. Whenever a node in DAG-FL performs one iteration of FL, it runs DAG-FL consensus and should first choose some tips on its local DAG to validate. The authentication of transactions can be validated by cryptography technology like RSA in blockchain fields, and the local model can be simply validated by computing the accuracy with a local test data set. Authenticated transactions with higher accuracy of local model would be chosen to construct the global model. The node then uses local data set to train the global model to get a local model. Finally, a new transaction that contains the newly trained local model is published and approves the tips which are used to construct the global model. Unlike the PoW consensus in other blockchained FL systems which consume resource on solving hash cryptography problems, DAG-FL consensus avoids the extra consumption of resource unrelated to FL.

DAG-FL consensus combines the voting mechanism of DAG ledger technology with the process of local model validation in FL, which can effectively detect abnormal nodes and mitigate their impact on FL. With the continuous extension of transactions on DAG, every approval of a transaction means that the local model on the approved transaction is selected to form a global model and influences the target model co-construction of FL. Consequently, the more approvals a transaction get, it will lead to the greater impact on FL, otherwise it will be isolated and has less impact on FL. Due to this unique consensus, the machine learning model in DAG-FL is always trained towards the direction that most nodes expect, and we assume that most nodes in DAG-FL are normal nodes while only a few nodes are abnormal. Abnormal transactions published by abnormal nodes usually have less prediction accuracy on test set than that of transactions published by normal nodes. Thus, compared with normal ones, the probability of abnormal transactions being approved by subsequent published transactions is much smaller. During the process of FL, abnormal transactions are isolated and their impact is minimized. In addition, nodes with too many isolated transactions can be detected by the DAG-FL as abnormal nodes, and then DAG-FL can react to these abnormal nodes.

C. FL Algorithm and Problem Formulation

DAG-FL is an asynchronous FL system without any central server, so we design a special FL algorithm to perform FL iterations. This part will give some numerical definition and introduce the FL algorithm of our DAG-FL.

Our DAG-FL is deployed on nodes which are mobile devices under a wireless network, such as the smart phones, wearable devices, and IoT devices. We assume that these nodes can communicate with each other considering an average communication bandwidth B under the wireless network. Let the set of mobile devices be denoted as $D = \{1, 2, 3, \cdots, N_D\}$ with $|D| = N_D$, where N_D is the number of nodes. D_i is the i-th node in D, and the set of training data on D_i is denoted as S_i with $|S_i| = N_i$, where N_i is the number of samples in S_i. Node D_i can create a local DAG g_i that is only visible to itself, and g_i can be periodically updated. Let the model stored in the transaction be denoted as ω. So the local model trained by node D_i at time t can be denoted as ω_i^t.

In order to build a common machine learning model with DAG-FL, e.g., a two-layer CNN model, usually thousands of FL iterations are requested. Initially, node D_i starts an FL iteration at t_0 by validating some tips on its local DAG first, and then choose k tips with local models $\omega_{d_1}^{t_1}$, $\omega_{d_2}^{t_2}$,..., $\omega_{d_k}^{t_k}$ ($t_1, t_2, ..., t_k \le t_0$, $d_1, d_2, ..., d_k \in D$) to aggregate a global model ω^{t_0} using the *FederatedAveraging* algorithm:

$$\omega^{t_0} = \sum_{i=1}^{k} n_i \omega_{d_i}^{t_i}, \tag{1}$$

where $\sum_{i=1}^{k} n_i = 1$, and n_i is the weight factor representing the importance of local models, and to simplify the FL algorithm of our DAG-FL, here we set $n_i = 1/k$ which means each local model is equally important.

After getting the global model, D_i extracts m samples from data set S_i as the mini-batch z_i to train the global model for β times. Once D_i gets a new local model $\omega_i^{t_0}$ trained by ω^{t_0}, a transaction with local model $\omega_i^{t_0}$ is published on g_i, and the FL iteration is completed. Samples in mini-batch z_i can be denoted as (x_i, y_i), where x_i is the feature set and y_i is the label set. Then the loss function in machine learning can be denoted as $f_{z_i}(\omega)$, where $f_{z_i}(\omega) = l(x_i, y_i; \omega)$, and is the prediction error of (x_i, y_i) in model ω.

In addition, due to the incentive mechanism of DAG-FL, D_i in DAG-FL expects to train global models in every participated FL iteration to get the local model ω_i, which can minimize $F_i(\omega_i)$ as follows.

$$\min F_i(\omega_i) = E_{z_i \sim S_i} f_{z_i}(\omega_i). \tag{2}$$

And for the whole DAG-FL system, external agents expect to get target model ω through smart contract after thousands of FL iterations to minimize $F(\omega)$ as follows.

$$\min F(\omega) = \frac{1}{N_D} \sum\nolimits_{i \in D} F_i(\omega_i). \tag{3}$$

IV. DAG-FL Operation

After a brief overview of DAG-FL, two algorithms involved in DAG-FL will be proposed in this section to introduce the operation process of DAG-FL in details.

The task publisher of FL in DAG-FL can be denoted as an external agent E, which can be regarded as an authoritative organization in the application layer holding a virtual machine to run the smart contract. Agent E can initialize the machine learning model in DAG-FL and update a local DAG periodically to detect whether to terminate the FL task. We regard a virtual DAG formed by all the published transactions in the DAG-FL network as the global DAG. As mentioned in the introduction of DAG-FL architecture, any node in DAG-FL updates its local DAG from adjacent nodes at a fixed time through wireless network. And this process can be seen as a node communicating with the global DAG to update its local DAG. When agent E publishes an FL task specifying the structure of the machine learning model with an expected final prediction accuracy ACC_0, then nodes in D are applied to participate in this FL task. We assume that the beginning time of an FL task is t_0, at t_0 agent E runs *DAG-FL controlling* algorithm to publish the initial transaction with the initial local model $\omega_0^{t_0}$ to the global DAG. After t_0, agent E updates its local DAG and gets global model to compute the current accuracy ACC_t periodically. If ACC_t is greater than or equal to ACC_0, agent E broadcasts an termination signal to all participating nodes in DAG-FL to terminate the FL, thereby the FL task is completed.

Algorithm 1 DAG-FL Controlling

```
1: External agent $E$ **executes:**
2: **Input** the framework of machine learning model and
   target accuracy $ACC_0$
3: Initialize the machine learning model with $\omega_0^{t_0}$
4: Publish initial transaction including $\omega_0^{t_0}$ to all dodes
5: **while** true **do**
6:    Update local DAG
7:    Choose $k$ tips to compute global model $\omega_0$
8:    Get accuracy $ACC_g$ by $\omega_0$
9:    **if** $ACC_g > ACC_0$ **then**
10:      Send end signal to all nodes
11:      **break**
12:   **end if**
13: **end while**
```

Algorithm 2 DAG-FL Updating

```
1: Any node $D_i$ **executes:**
2: **while** true **do**
3:    **if** end signal is recieved **then**
4:       **break**
5:    **else**
6:       Update local DAG
7:       **if** idle state **then**
8:          Validate no more than $\alpha$ tips on local DAG
9:          Choose $k$ tips with the highest accuracy rate to
            compute global model $\omega_i$
10:         Train $\omega_i$ with mini-batch $z_i$ for $\beta$ times to get local
            model $\omega_i^t$
11:         Publish a new transaction including $\omega_i^t$ and ap-
            prove $k$ transactions used to compute $\omega_i$ on local
            DAG
12:      **end if**
13:   **end if**
14: **end while**
```

After t_0, all the participating nodes run the *DAG-FL updating* algorithm to achieve DAG-FL consensus whenever they are in idle state. As nodes in DAG-FL are mobile devices, nodes may shutdown during one iteration of FL because of device asynchrony. And any node in DAG-FL which is in idle state should go through four stages to participate within one iteration:

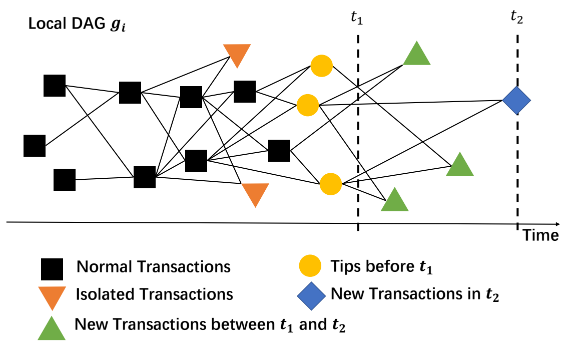

Fig. 2: Transaction alternations on DAG g_i.

1) Stage 1: The node selects some tips (no more than α) from local DAG randomly.

2) Stage 2: The node first validates the authentication of tips selected in stage 1. Then the node computes the prediction accuracy of the local models in the selected tips using its own test data set.

3) Stage 3: The node chooses k ($k < \alpha$) tips selected in the first stage with the highest accuracy to run *FederatedAveraging* algorithm and gets a global model. The node utilizes local data set to train the global model and gets a trained local model.

4) Stage 4: A new transaction is constructed. The new transaction contains authentication information, local model trained in the third stage, and the approval information to approve the k tips chosen in the third stage.

After the completion of the above four stages, the node then successfully finishes one iteration in DAG-FL and publishes the new transaction to the DAG, which will be soon observed by all other nodes in DAG-FL.

Here we take node D_i in DAG-FL as an example to illustrate the alterations of transactions on g_i during the process of running the *DAG-FL updating*. Suppose that node D_i is in idle state at t_1 and intends to perform an iteration of DAG-FL. As shown in Fig. 2, D_i selects α recently published and has not been yet approved tips from its local DAG g_i to validate them. Node D_i then computes the global model ω_i with k validated transactions by using the *FederatedAveraging* algorithm. After D_i trains ω_i with the mini-batch z_i of local data set S_i for β times, a new local model $\omega_i^{t_2}$ is built, at time t_2. Finally, a new transaction contained the local model $\omega_i^{t_2}$ and approval information is published to DAG g_i.

V. Simulation

We design a simulation platform pySimuFL that can evaluate the performance of the proposed DAG-FL as compared to Google FL [1] and Block FL [3]. For simulation, there are 100 nodes in our pySimuFL that share the same wireless network with a radius of 1km. In the following simulation experiments, the goal of DAG-FL is to build a CNN machine learning model which is also used in Google FL [1]. The used CNN model has two 5x5 convolution layers (the first with 32 channels, the second with 64, each followed with 2x2 max pooling), a fully connected layer with 512 units and ReLu activation, and a final softmax output layer (1,663,370 total parameters). We also use the MNIST set as the data set like Mecmahan did in [1]. To reflect the non-IId feature of local data on mobile devices, 60000 training set is equally divided into 200 groups, in which the samples in a group have same labels and every node has two groups of 600 samples of training data with two different labels. In order to enable models to validate transactions, 10000 validation data set is equally allocated to the 100 nodes with different labels for validation. Considering a quasi-static network environment, we set up a relatively conservative bandwidth B of wireless network, i.e., 20 Mbps. Nodes are set to be in idle state for FL at different times, thus enabling one node on average ready for a DAG-FL iteration per second. The results of Blcok FL and Google FL in the following figures are obtained by using the same nodes setting as DAG-FL with the fittest learning rates. The average time consumption per 100 iterations in our piSimuFL for the three FLs is shown in Table. 1.

TABLE I: Iteration delay

FL Systems	Average latency for 100 iterations/s
Google FL	132.06
Block FL	208.33
DAG-FL	103.05

We first evaluate the accuracy of DAG-FL compared with other FL systems in the ideal case shown as Fig. 3.(a). Note that the accuracy of local CNN is obtained by training CNN model on a single node with the whole MNIST data set. The increase rate of accuracy can indicate the convergence rate of the CNN model. According to Fig. 3.(a), we find that all the three systems can accelerate the convergence rate of CNN model. Although DAG-FL has a lower convergence rate than Block FL and Google FL in the early stage, DAG-FL finally achieves the highest accuracy. Furthermore, Block FL and Google FL have nearly the same accuracy performance with the increase of iterations. However, Block FL has the higher latency per iteration among the three FLs as shown in Table 2. This is because that Block FL should consume much time to run PoW consensus mechanism unrelated to FL. We can also notice that DAG-FL generates less latency than Google FL per iteration, this is because that DAG-FL is an asynchronous system and needs not to solve device asynchrony problem. In summary, we can conclude that DAG-FL is an effective FL system than Block FL and Google FL.

We consider normal and abnormal nodes, where abnormal nodes can be set as lazy nodes and malicious nodes. The lazy node reads the model parameters from an existing transaction and uploads them directly without training the global model with local data, so as to obtain potential rewards of FL and achieve the purpose of laziness. In contrast, the malicious node can train the global model with wrong data set to get poisonous transactions.

The accuracy of different FL systems with 10 percent lazy nodes are shown in Fig. 3.(b), and we find that DAG-FL is more immune to lazy nodes. It can be observed that Block FL

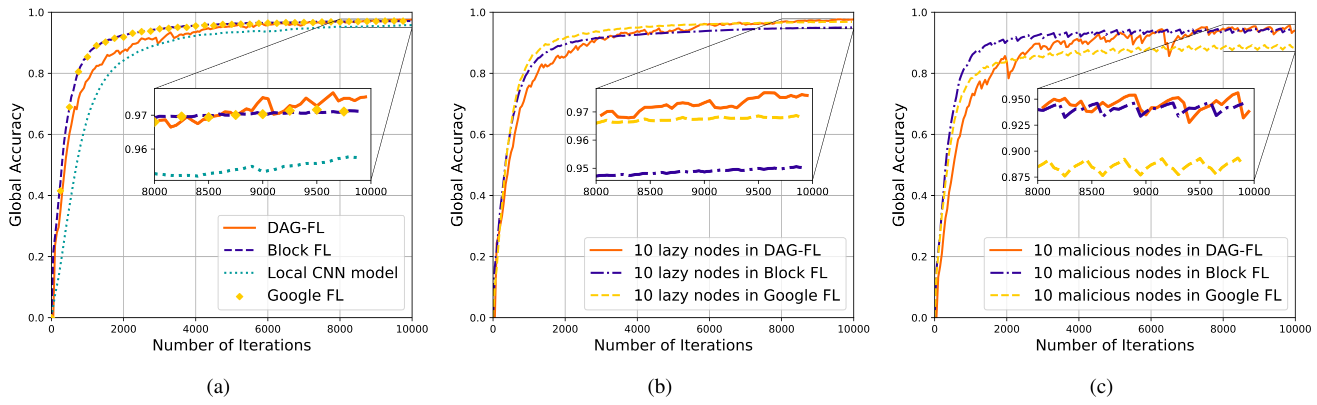

Fig. 3: (a) The accuracy of different federated learning systems in ideal case, (b) The accuracy of different federated learning systems with lazy nodes, (c) The accuracy of different federated learning systems with malicious nodes.

is significantly affected by the participation of lazy nodes in term of the convergence rate and the accuracy. The reason is that with more associated lazy nodes, the miner in Block FL usually has more time to run PoW to win the right to publish the next block. This means that more transactions published by normal nodes are dropped instead of being adopted for the global model updating.

We demonstrate the accuracy of different FL systems with 10 percents malicious nodes in Fig. 3.(c), and find DAG-FL and Block FL are both insensitive to the participation of malicious nodes. However, the accuracy of Google FL significantly reduces when malicious nodes are involved, this is due to the fact that Google FL is not capable of distinguishing malicious nodes. In contrast, Google FL can only use the average algorithm to dilute the harmful model parameters uploaded by malicious nodes, so as to achieve the purpose of mitigation.

VI. Conclusion

In this paper, we proposed a DAG empowered FL system named as DAG-FL, which combines DAG ledger technology to overcome the problems of mobile device asynchrony and abnormal nodes to complete the model co-construction task of FL efficiently. DAG-FL is constructed by a three-layer asynchronous architecture including FL, DAG, and application layers taking the responsibilities of model training, communicating and observing. In this manner, DAG layer could provide blockchain as a service (BaaS) establishing *DAG-FL controlling* and *DAG-FL updating* algorithms for operation process of DAG-FL. Finally, experimental results showed that DAG-FL has the higher system efficiency and a better target model with device asynchrony compared with other two benchmark FL systems. Meanwhile, DAG-FL is also proved to be insensitive to the impact of abnormal nodes and has the ability of anomaly detection.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th International Conference on Artificial Intelligence and Statistics*, vol. 54. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr. 2017, pp. 1273–1282.

[2] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[3] H. Kim, J. Park, M. Bennis, and S. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2020.

[4] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[5] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[6] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[7] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. 2018 35th International Conference on Machine Learning*, vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul. 2018, pp. 3043–3052.

[8] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. 23rd International Conference on Artificial Intelligence and Statistics*, vol. 108. PMLR, 26–28 Aug. 2020, pp. 2938–2948.

[9] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *Proc. 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, Rotorua, New Zealand, 2019, pp. 374–380.

[10] Y. Chen, W. Liu, Z. Niu, Z. Feng, Q. Hu, and T. Jiang, "Pervasive intelligent endogenous 6g wireless systems: Prospects, theories and key technologies," *Digital Communications and Networks*, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S235286482030242X.

[11] U. Majeed and C. S. Hong, "Flchain: Federated learning via mec-enabled blockchain network," in *Proc. 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2019, pp. 1–4.

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[13] Y. Li, B. Cao, M. Peng, L. Zhang, L. Zhang, D. Feng, and J. Yu, "Direct acyclic graph-based ledger for internet of things: Performance and security analysis," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1643–1656, 2020.

[14] B. Cao, Y. Li, L. Zhang, L. Zhang, S. Mumtaz, Z. Zhou, and M. Peng, "When internet of things meets blockchain: Challenges in distributed consensus," *IEEE Network*, vol. 33, no. 6, pp. 133–139, 2019.

[15] S. Popov, "The tangle," 10 Apr. 2018. [Online]. Available: https://www.iota.org/research/academic-papers