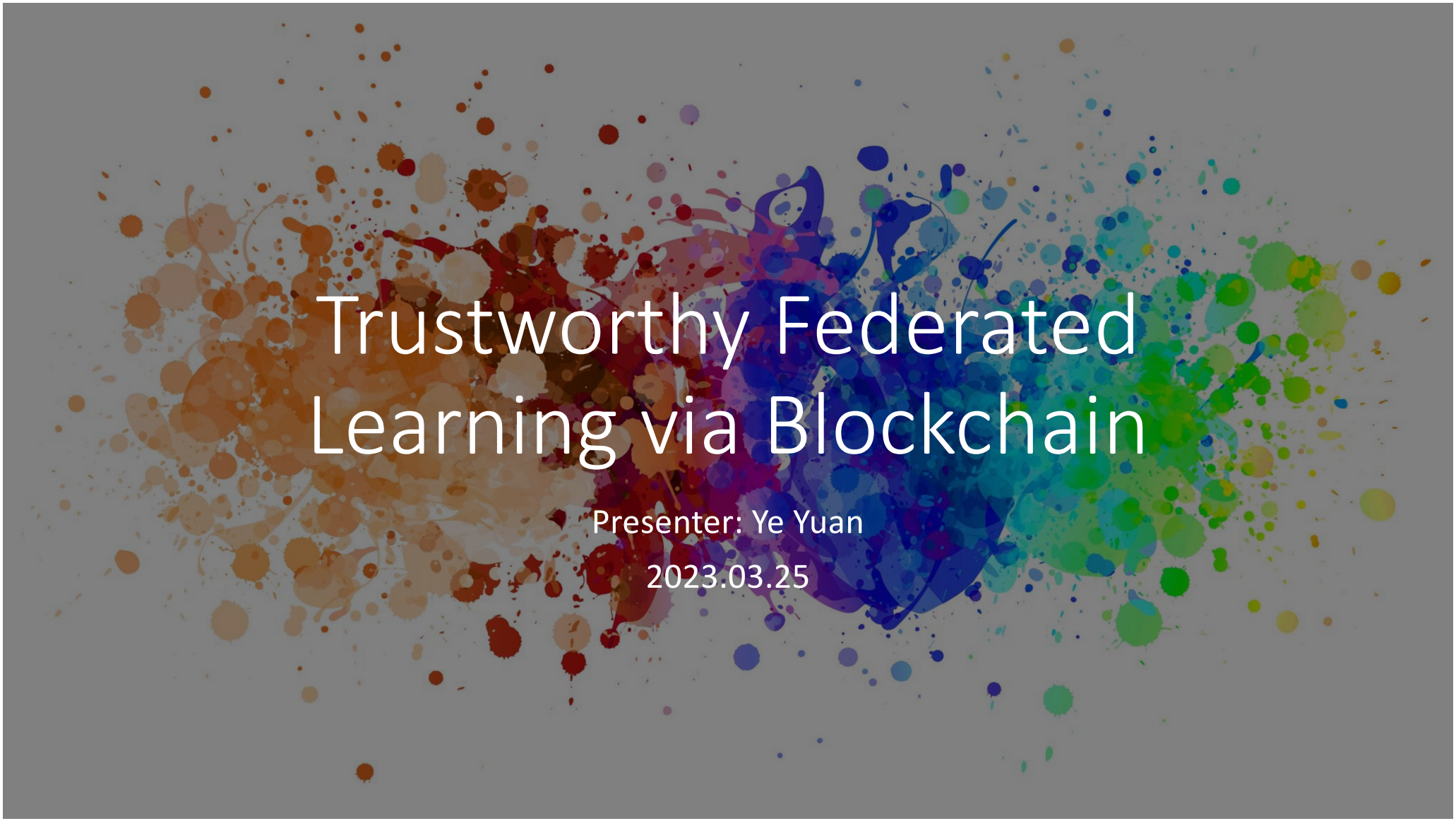


Trustworthy Federated Learning via Blockchain

Presenter: Ye Yuan

2023.03.25

Paper Background

- Venue: IEEE INTERNET OF THINGS JOURNAL, VOL. 10, NO. 1,
- Time: 1 JANUARY 2023

Abstract

***Abstract*—The safety-critical scenarios of artificial intelligence (AI), such as autonomous driving, Internet of Things, smart healthcare, etc., have raised critical requirements of trustworthy AI to guarantee the privacy and security with reliable decisions. As a nascent branch for trustworthy AI, federated learning (FL) has been regarded as a promising privacy preserving framework for training a global AI model over collaborative devices. However, security challenges still exist in the FL framework, e.g., Byzantine attacks from malicious devices, and model tampering attacks from malicious server, which will degrade or destroy the accuracy of trained global AI model. In this article, we shall propose a decentralized blockchain-based FL (B-FL) architecture by using a secure global aggregation algorithm to resist malicious devices, and deploying a practical Byzantine fault tolerance consensus protocol with high effectiveness and low energy consumption among multiple edge servers to prevent model tampering from the malicious server. However, to implement B-FL system at the network edge, multiple rounds of cross-validation in blockchain consensus protocol will induce long training latency. We thus formulate a network optimization problem that jointly considers bandwidth and power allocation for the minimization of long-term average training latency consisting of progressive learning rounds. We further propose to transform the network optimization problem as a Markov decision process and leverage the deep reinforcement learning (DRL)-based algorithm to provide high system performance with low computational complexity. Simulation results demonstrate that B-FL can resist malicious attacks from edge devices and servers, and the training latency of B-FL can be significantly reduced by the DRL-based algorithm compared with the baseline algorithms.**

***Index Terms*—Blockchain, federated learning (FL), long-term latency minimization, resource allocation, trustworthy artificial intelligence (AI).**

I. Introduction

- General Introduction
- A. Contributions
- B. Related Works
- C. Organization

General Introduction – part 1

ARTIFICIAL Intelligence (AI) has yielded a bloom of worldwide developments and profoundly changed human life, such as autonomous driving [1], Internet of Things (IoT) [2], smart healthcare [3], etc. However, recent research results have found that AI may cause potential vulnerabilities by, for example, leaking privacy data or making unreliable decisions under adversarial attacks (e.g., misdiagnosed disease in smart healthcare). Consequently, a new AI paradigm is emerging, named *trustworthy AI* [4], which aims at avoiding unfavorable impacts from AI and has garnered increased interest in both academia and industry. In particular, the trustworthy AI intends to achieve stable and sustained high learning accuracy under the considerations of robustness, privacy, accountability, fairness, interpretability, and environmental wellbeing [4]. As a promising framework of trustworthy AI, federated learning (FL) has recently been proposed to guarantee data privacy in the AI model training process, which collaboratively trains a global AI model by coordinating multiple devices [5], [6]. In the server–client architecture of FL, each device executes local training and uploads its updated local model to a server without transmitting the private raw data. The server then aggregates the collected local models for global AI model update, followed by disseminating the updated global model to devices [2]. Therefore, FL provides a privacy protection solution for privacy-sensitive intelligent applications, e.g., smart healthcare, financial industry, and IoT [7], [8], [9], [10], [11].

General Introduction – part 2

However, security challenges still exist in FL due to the potential malicious devices and malicious server, which will degrade the performance of AI model training or destroy the FL training process [7]. Although FL guarantees the privacy protection of edge devices, the global model can be attacked by malicious edge devices (e.g., model poisoning or adversarial attacks) [12]. Various secure model aggregation algorithms have been proposed to address this challenge, e.g., geometric median, trimmed mean, and multi-Krum [13]. In particular, the server–client architecture is vulnerable to server's malfunction, which includes a potential single point of failure or tamper of global model. The failure or tamper in the server will collapse the entire FL training. To address this issue, blockchain was adopted to establish a decentralized blockchain-based FL (B-FL) network [14], [15], [16], which leverages multiple servers to execute global model aggregation and confirm the correctness of global model using consensus protocols [e.g., Proof of Work (PoW), Proof of Stake (PoS), Raft, and practical Byzantine fault tolerance (PBFT)] [17], [18]. The

B-FL can thus resist failures or attacks of servers and devices by building trustworthy global model aggregation with secure model aggregation based on the blockchain consensus protocol among multiple servers [19], [20], [21], [22].

General Introduction – part 3

Moreover, the latency of the FL training process, one of the key metrics in the edge AI, is also crucial to improve the communication efficiency of FL due to the fading nature of wireless channels [23], [24], [25], [26]. To implement the FL over wireless networks, the edge devices transmit their local models and the edge server disseminates the shared global model over wireless links. This typically consists of computation latency and communication latency, for which theoretical analysis and resource allocation for FL latency have recently been provided [27], [28], [29], [30]. Specifically, Li *et al.* [27] characterized the delay distribution for FL over arbitrary fading channels via the saddle point approximation method and large deviation theory. Chen *et al.* [28] proposed a probabilistic device scheduling policy to minimize the overall training time for wireless FL. Ren *et al.* [29] formulated a training acceleration optimization problem developed the close-form expressions for joint batch size selection and communication resource allocation. Yang *et al.* [30] considered a delay minimization problem to obtain the optimal solution by a bisection search algorithm. However, the latency in wireless B-FL becomes much more complicated due to the additional multiple rounds of cross-validation among edge servers in blockchain consensus protocol. The computation latency includes the local training, global model aggregation, and block validation, which depends on the computation capability of edge devices and servers, and the size of data set and models. The communication latency includes the uplink and downlink transmissions for model updates and blockchain consensus protocol, which mainly depend on wireless communication techniques, bandwidth, and power budgets. It is thus critical to characterize the latency of the wireless B-FL system and optimize the network resources to reduce the overall learning latency.

General Introduction – part 4

In this article, we focus on designing a B-FL architecture to support trustworthy and low latency AI service under the consensus protocol of blockchain. Specifically, we utilize the PBFT consensus protocol [31] to achieve high effectiveness and low energy consumption compared with the existing consensus protocols such as PoW, and Raft with some loss of security [18]. The training latency in B-FL system is characterized by considering the computation latency and communication latency, which consists of local training, global model aggregation, consensus protocol, and model dissemination. We then propose to minimize the long-term average learning latency by allocating bandwidth and power resources. This long-term resource allocation problem cannot be equivalently transformed to multiple one-shot problems with respect to the channel of each round, due to the correlation in the long-term average power constraint [32]. Besides, as the optimization variables are coupled in the objective function and constraints, the conventional optimization-based algorithms have prohibitive computational complexity and may be inapplicable in the time-sensitive B-FL system [15], [33], [34]. To address the challenges, we propose to transform the long-term resource allocation problem to a Markov decision process (MDP), for which a deep reinforcement learning (DRL)-based algorithm is developed to achieve efficient and adaptive resource allocation with low computational complexity from a long-term perspective. The developed DRL-based algorithm will minimize the cumulative latency from a long-term perspective, which balances the latency in current round and that in the future rounds by dynamic resource allocation. Moreover, the DRL-based algorithm establishes a direct mapping from the current network information to resource allocation by deep neural network (DNN), which can significantly reduce the computational complexity.

A. Contributions

A. Contributions

In this article, we propose a PBFT-based wireless B-FL architecture to realize the trustworthy AI training process. To improve the communication efficiency of wireless B-FL, we further propose to minimize long-term average training latency by allocating the bandwidth and power resources. The long-term resource allocation problem is modeled as an MDP, followed by developing a DRL-based algorithm to achieve high system performance and reduce computational complexity. The major contributions of this article are summarized as follows.

1) We proposed a PBFT-based wireless B-FL architecture by combining the permissioned blockchain and wireless FL system at the network edge, which builds a trustworthy AI model training environment to resist the failures and attacks from malicious edge servers and malicious edge devices. The wireless B-FL utilizes PBFT consensus protocol of blockchain to validate the correctness of global model update, which achieves high effectiveness and low energy consumption. The detailed procedures of PBFT-based wireless B-FL system are presented, followed by characterizing the training latency by considering the communication and computation processes.
2) By jointly considering the consensus and training processes in the long-term process of B-FL, we formulate a long-term average latency minimization problem by optimizing bandwidth allocation and power allocation. We adopt long-term average power constraint for dynamic power allocation to assign more power in worse channel conditions and less power in better channel conditions. The long-term resource allocation problem turns out to be highly intractable due to the correlation in the long-term average power constraint and, thus, cannot be equivalently transformed to multiple one-shot problems.
3) To solve the long-term resource allocation problem, we propose to transform the resource allocation problem into an MDP, followed by developing a DRL-based algorithm to design efficient and adaptive resource allocation scheme with low computational complexity from a long-term perspective. We then utilize a twin delayed deep deterministic policy gradient (TD3) algorithm in DRL to tackle the continuous optimization variables.

The simulation results demonstrate the effectiveness of wireless B-FL to resist malicious attacks from edge devices and servers, which are simulated in the MINST data set for handwriting recognition and the heart activity data set for affect recognition. We further simulate the proposed DRL-based resource allocation scheme with various system parameters, and demonstrate the effectiveness for achieving significant reduction on training latency compared with other baseline algorithms.

B. Related Works

B. Related Works

The B-FL system has recently received significant interests in designing trustworthy AI by leveraging the consensus protocol of blockchain and a recent survey paper has presented the fundamental concepts and opportunities in the integration of FL and blockchain, which is called FLchain [20]. In particular, Qu *et al.* [35] and Ma *et al.* [36] proposed a B-FL architecture based on the PoW consensus protocol, which uses the miners in the blockchain as the centralized aggregator to replace the single dedicated server in the traditional FL system. Besides, Li *et al.* [37] proposed a B-FL framework with an innovative committee consensus mechanism to avoid malicious central servers and realize effective decentralized storage. The Biscotti in [38] designed a Proof-of-Federation (PoF) consensus protocol, which coordinates FL training between devices to generate blockchain by providing beneficial model updates or by facilitating the consensus process. Lu *et al.* [39] proposed a B-FL with delegated PoSs (DPoSs), which is developed in digital twin wireless networks. Based on the above architectures of B-FL with different consensus protocols, there are several works optimizing the network efficiency, e.g., latency and energy consumption. Kim *et al.* [40], Li *et al.* [41], and Pokhrel and Choi [42] analyzed an end-to-end latency model of B-FL based on PoW, characterized the close-form of optimal block generation rate in different scenarios, and formulated an edge association by jointly considering digital twin association, training data batch size, and bandwidth allocation, respectively. The paper [43] considered blockchain-enabled FL with PoW consensus protocol and achieved a certain model accuracy while minimizing the energy consumption and training latency with a reasonable payment by DRL approach. Nguyen *et al.* [44] employed a decentralized FL model aggregation algorithm in a peer-to-peer-based blockchain network and formulated a system latency minimization problem by optimizing communication and computation resources. Furthermore, to improve privacy protection Zhao *et al.* [45] jointly enforced differential privacy on the extracted features and proposed a new normalization technique to protect customers' privacy and improve the test accuracy in the B-FL system with Algorand consensus protocol. Moreover, several works are already deployed in the real blockchain platforms. For example, Zhang *et al.* [46] tackled the challenge of data heterogeneity in failure detection of industrial IoT and implement the B-FL system in Ethereum to evaluate the feasibility, accuracy, and performance. Kang *et al.* [47] presented efficient reputation management for mobile devices based on consortium blockchain and establish the reputation blockchain system on the Corda.

However, all the above works simply adopted the traditional PoW consensus protocol [36], [40], [41] or proposed some immature consensus protocols (e.g., committee consensus mechanism [37], and PoF [38]). In this article, we utilize the PBFT consensus protocol to build a permissioned blockchain, which achieve high effectiveness and low energy consumption compared with famous and widely used PoW [18].

C. Organization

C. Organization

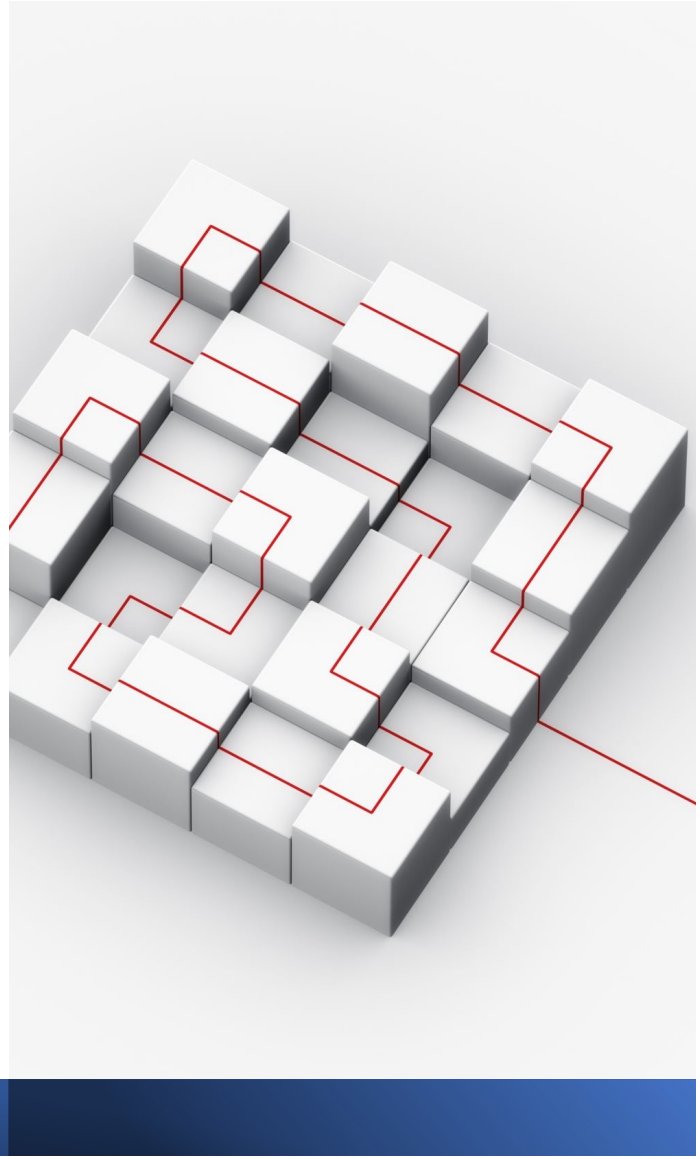
The remainder of this article is organized as follows. Section II introduces the wireless B-FL system architecture and the communication model. Section III analyzes the training latency of wireless B-FL and formulates the long-term average latency minimization problem. The TD3-based DRL algorithm is developed in Section IV. Section V presents simulation results of the proposed algorithm by comparing with other baseline algorithms. The notations in this article are listed in Table I.

TABLE I
LIST OF NOTATIONS

Notation	Description	Notation	Description
M	number of edge servers	$b^{\max}$	maximum system bandwidth
K	number of devices	$\bar{p}$	long-term average power constraint
$\mathcal{B}$	set of edge servers	$h^t_{D_k,B_m}$	channel gain between edge device D_k and edge server B_m at t-th round
$\mathcal{D}$	set of edge devices	$\boldsymbol{b}^t$	bandwidth allocation at t-th round
s_{D_k}	batch size of devices D_k	$\boldsymbol{p}_t$	transmit power of edge servers and devices at t-th round
$\mathcal{S}$	set of local datasets	R_{D_k,B_m}	achievable transmission rate from edge device D_k to edge server B_m
B_p	index of primary edge server	f_{B_m}	CPU frequency of edge server B_m
$\boldsymbol{w}_k$	local model of device U_k	σ	unit CPU cycle for executing secure model aggregation
$\boldsymbol{w}_g$	global FL model	δ	CPU cycle for training one sample
f	number of Byzantine edge servers	ρ	CPU cycle for generating or verifying one signature
$D(B)$	digest of block (e.g. hash value)	ϖ	size of transactions
H_B	block height in blockchain	S_B	size of blocks
N_0	power spectral density of AWGN	S_M	size of consensus messages
γ	discount factor	ϕ, θ_1, θ_2	parameters of online actor and critic networks
η_a, η_c	learning rate of actor and critic networks	$\phi', \theta_1', \theta_2'$	parameters of target actor and critic networks
κ	update proportion of target networks	$\mathcal{R}$	reply memory buffer
ϑ	update frequency of target networks	E	steps of exploration

II. System Model

- A. Architecture of Wireless Blockchain-Based FL
- B. Procedures of Wireless B-FL
  - 1) Local Training
  - 2) Upload Local Models
  - 3) Global Model Aggregation
  - 4) Preprepare
  - 5) Prepare
  - 6) Commit
  - 7) Reply
  - 8) Download Global Model
- C. Communication Model



A. Architecture of Wireless Blockchain-Based FL – part 1

A. Architecture of Wireless Blockchain-Based FL

In this article, we instead consider a wireless B-FL system consisting of M edge servers and a set of K edge devices, as shown in Fig. 1. Each edge server is equipped with a mobile edge computing (MEC) sever to execute computing tasks, and the edge servers are indexed by $\mathcal{B} = \{B_1, \ldots, B_m, \ldots, B_M\}$. Each edge server has enough computation and storage resources to execute global model aggregation and consensus protocol [14]. Meanwhile, K edge devices are indexed by $\mathcal{D} = \{D_1, \ldots, D_k, \ldots, D_K\}$. The edge devices with local data set $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_k, \ldots, \mathcal{S}_K\}$ have enough computing resources and storage resources to execute the local training process of B-FL. The edge servers communicate with each other over wireless channels [14], [48]. In summary, the B-FL architecture consists of three layers, i.e., the edge device layer to train local model, the edge server layer to execute secure mode aggregation by blockchain technique, and overlaided blockchain network layer, as depicted in Fig. 1.

The conventional server–client architecture for wireless FL consists of single edge server and multiple edge devices, which collaboratively train a shared global model without sharing private data of edge devices [49], [50]. However, the server–client architecture is vulnerable to edge server's malfunction, which includes a potential single point of failure and malicious global model poison [51]. A failure in the edge server may collapse the entire FL network. In this article, the proposed wireless B-FL combines the wireless FL with permissioned blockchain to execute global model aggregation based on the blockchain consensus protocol among multiple edge servers, which can resist failures or attacks of edge servers for trustworthy global model aggregation [19]. The permissioned blockchain provides an attractive solution for malicious server attacks in conventional single server wireless FL due to its features, such as decentralization, immutability, and traceability [22], [36]. In particular, the wireless B-FL system consists of multiple edge servers and multiple edge devices [52], which generates a blockchain in a form of blocks linked by cryptography under the control of a consensus protocol to confirm that the data in the block is correct and immutable. The block is a data structure consisting of all information of local models and global model. Therefore, the decentralized blockchain technology is adopted to provide a trustworthy model aggregation platform powered by distributed consensus protocol, which mitigates the single point of failure and malicious poison of edge server and ensures transparency and immutable.

A. Architecture of Wireless Blockchain-Based FL – part 2

Specifically, the B-FL system comprises one primary edge server and several validator edge servers to produce a new block and reach a consensus. In each training round, each edge device trains the local model based on its own data set followed by uploading the local model to the primary edge server. The primary edge server validates the identities of local models and aggregates them to the global model by executing the smart contract. The smart contract consists of lines of code, which is immutable and trackable to execute a function, e.g., secure global model aggregation. Then, the local models and global model are packed into a new block, which is broadcasted among validator edge servers to validate the correctness of global model. For example, the validator edge servers recalculate the global model and compare it with the global model from the primary edge server. The validator edge servers will reach a consensus on the correctness of global using consensus protocols (e.g., PoW and PBFT). B-FL will enter the next training round if the global model is correct. However, if the primary edge server computes incorrectly or tampers with the global model, the validator edge servers will ignore the global model from the malicious primary edge server and choose another primary edge server to restart the training round. After the validator edge servers reaching a consensus, the new block is stored in the blockchain to ensures transparency and traceability. The validated global model is then disseminated to edge devices. Therefore, the consensus protocol is the critical factor to ensure the uniformity and security of the decentralized blockchain system.

In this article, we adopt the PBFT consensus protocol in the proposed B-FL system to realize secure global model aggregation among edge servers. Compared with the famous and widely used PoW consensus protocol [35], [36], [40], [41], [42], [43], PBFT has weaker security performance. Generally, PoW is a permissionless consensus protocol, which means arbitrary edge sever can participate in the consensus and can tolerate 50% computing power attack. PBFT is a permissioned consensus protocol, which means only the authorized edge sever can participate in the consensus and can just resist 33% malicious edge servers. However, PBFT achieves more effective consensus and lower electrical energy consumption than PoW. In particular, the consensus rate of PBFT is hundreds of times higher than that of PoW and the PoW-based Bitcoin costs as much electrical energy as the whole of Switzerland [18]. To achieve high effectiveness and low energy consumption, we choose the PBFT consensus protocol in our proposed B-FL system.

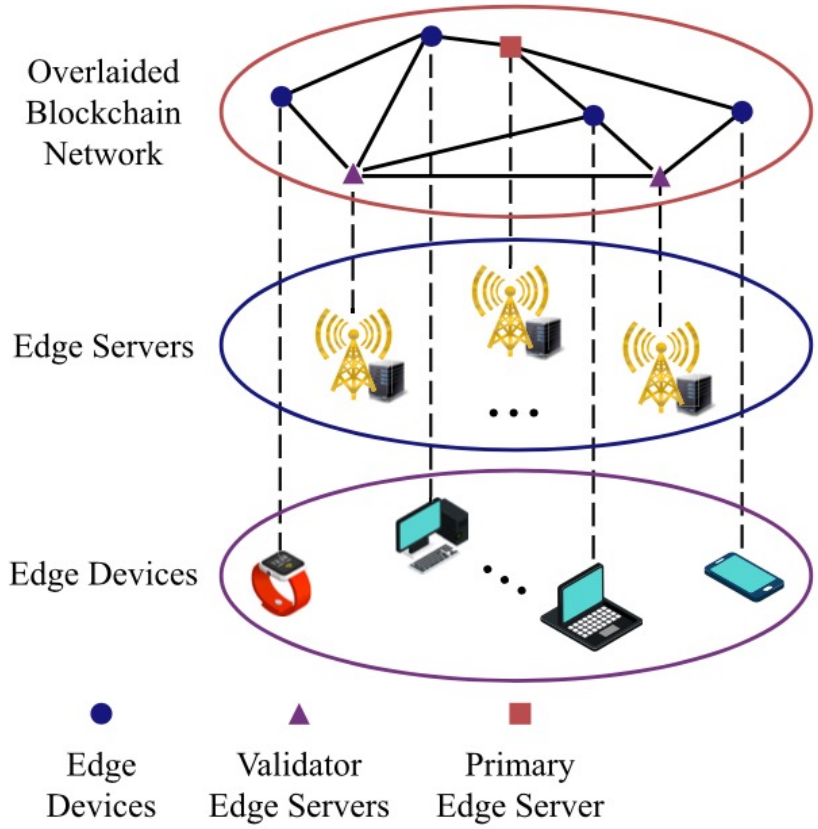


Fig. 1. Wireless B-FL system.

B. Procedures of Wireless B-FL

B. Procedures of Wireless B-FL

To train the ML models in the wireless B-FL system based on PBFT consensus protocol, eight steps are required in each round to achieve trustworthy global model aggregation to resist failures and attacks of malicious edge servers. The eight steps are also shown in Fig. 2.

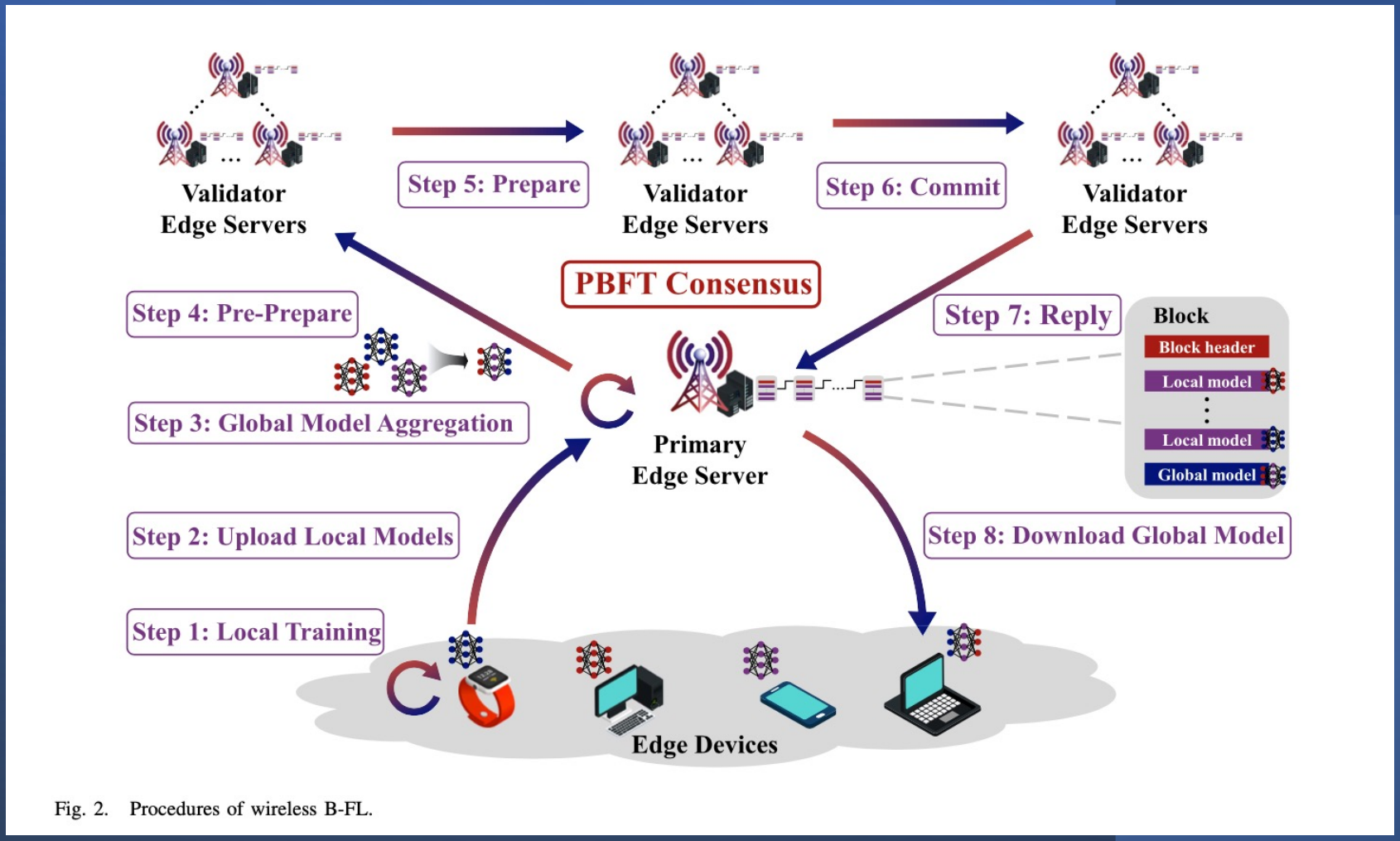


Fig. 2. Procedures of wireless B-FL.

1) Local Training

1) Local Training: Each edge device uses the local data set to train the local model through stochastic gradient descent. For each edge device $D_k \in \mathcal{D}$, the local loss function on the batch of samples $\mathcal{S}_k$ randomly from a local data set can be denoted by

$$F_k(\boldsymbol{w}; \mathcal{S}_k) = \frac{1}{s_{D_k}} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{S}_k} f(\boldsymbol{w}; \boldsymbol{x}_i, y_i) \tag{1}$$

where s_{D_k} denotes the batch size of edge device D_k, $\boldsymbol{x}_i$ and y_i are features and label, respectively, and $\boldsymbol{w}$ denotes the parameter of the ML model. In particular, $f(\boldsymbol{w}; (\boldsymbol{x}_i, y_i))$ is the sample-wise loss function, which depends on the ML model. Then, the local model parameter can be updated by the stochastic gradient descent algorithm, i.e.,

$$\boldsymbol{w}_k^t = \boldsymbol{w}_k^{t-1} - \eta \nabla F_k\left(\boldsymbol{w}_k^{t-1}; \mathcal{S}_k\right) \tag{2}$$

where $\boldsymbol{w}_k^t$ denotes the trained model parameters of edge device D_k in round t, $\nabla F_k(\boldsymbol{w}_k^{t-1}; \mathcal{S}_k)$ is the gradient of $F_k(\cdot)$ at point $\boldsymbol{w}_k^{t-1}$, and η is the learning rate. In this step, the computation latency is the local model parameter calculation at the edge devices.

2) Upload Local Models

2) Upload Local Models: With allocated wireless resources (e.g., bandwidth, transmit power), the edge devices upload the local models to the primary edge server as transactions $<\boldsymbol{w}_k, D_k> \forall D_k \in \mathcal{D}$, where $<\boldsymbol{w}_k, D_k>$ denotes a data packet consisting of local model parameter $\boldsymbol{w}_k$ and the data packet is signed by edge device D_k. The digital signature is a cryptology technique, which is used to validate the authenticity and integrity of data packets and guarantee that the information in data packet is not be tampered with others. The transaction is a concept in blockchain system involving cryptocurrency, contracts, records, or other information to record all behavior in blockchain. In this article, we assume that the data packets are encoded into digital signals (e.g., polar code) and transmitted by rate adaptation scheme to ensure error-free transmission. It is worth noting that the primary edge server is appointed before each training round (e.g., the primary edge server rotates among all the edge servers) and all edge devices and edge servers know the primary edge server before each round. In this step, the communication latency is the transaction uploading for all edge devices.

3) Global Model Aggregation

3) Global Model Aggregation: After collecting the data packets $<\boldsymbol{w}_k, D_k> \forall D_k \in \mathcal{D}$, the primary edge server validates the transactions to confirm the local model owner valid and executes smart contract to aggregate them into a global model. The primary edge server aims at minimizing the global loss function through global aggregation, which is defined as

$$F(\boldsymbol{w}) = \sum_{D_k \in \mathcal{D}} F_k(\boldsymbol{w}; \mathcal{S}_k). \tag{3}$$

Although FL guarantees the privacy protection of edge devices to a certain extent, the global model can be attacked by malicious edge devices (e.g., data or model poisoning) [12]. The proposed B-FL is compatible with all secure global aggregation algorithms (e.g., geometric median, trimmed mean, and Multi-Krum). We use multi-KRUM algorithm to realize Byzantine-resilient secure global model aggregation in this article, which outperforms other algorithm in convergence speed [13], [38], [53], i.e.,

$$\boldsymbol{w}_g^t = \text{multi\_KRUM}(\{\boldsymbol{w}_k^t \ \forall D_k \in \mathcal{D}\}) \tag{4}$$

where $\boldsymbol{w}_g^t$ is the global model in round t, and $\boldsymbol{w}_k^t$ is the local model from edge device D_k. The function multi_KRUM$(\cdot)$ is the multi-KRUM algorithm, which is defined in Algorithm 1. After that, the local models and global model are packed into a new block, which is a data packer denoted by $B = <\{<\boldsymbol{w}_1, D_1>\}_{D_k \in \mathcal{D}}, <\boldsymbol{w}_g, B_p>>$, where B_p is the notation of primary edge server. In this step, the computation latency is the transaction validation and model aggregation in the primary edge server.

4) Preprepare

4) Preprepare: After producing the new block, the primary edge server broadcasts the new block with a preprepare message, i.e., <PRE-PREPARE, H_B, $D(B)$, B, B_p>. H_B is the block height (i.e., the number of valid blocks in the blockchain), $D(B)$ is a digest of B (e.g., hash value) and the preprepare message is signed by primary edge server B_p. The validator edge servers, which are all edge servers except for the primary edge server, receive the new block and verify the digital signature of transactions and new block to confirm the valid digital signature of new block. Furthermore, the global model is recalculated to confirm that the primary edge server computes correctly. After verification, each validator edge server stores the new block to local. In this step, the computation latency is the digital signature validation of new block and recalculation of global model. The communication latency is the block transmission from the primary edge server.

5) Prepare

5) Prepare: After verification, the validator edge servers transmit the prepare messages <PREPARE, H_B, $D(B)$, B_m> $\forall B_m \in \mathcal{B}, B_m \neq B_p$ to others. The primary edge server and all validator edge servers receive prepare messages and confirm that a majority of validator edge servers have verified the new block and agree with the new block. In this step, the computation latency is the digital signature validation of prepare messages in all edge servers. The communication latency is the message transmission from the validator edge servers.

6) Commit

6) Commit: After the agreement of new block, the validator edge servers will broadcast the commit messages $<\text{COMMIT}, H_B, D(B), B_m> \forall B_m \in \mathcal{B}$ to others. The validator edge servers receive enough commit messages and confirm that most edge servers reach a consensus. In this step, the computation latency is the digital signature validation of commit messages in all edge servers. The communication latency is the message transmission from all edge servers.

7) Reply

7) Reply: After reaching the consensus, the new block becomes valid and all validator edge servers return the reply message $<\text{REPLY}, H_B, D(B), B_m> \forall B_m \in \mathcal{B}, B_m \neq B_p$ to the primary edge server, which indicates that they have stored the new block in the blockchain ready for the next consensus process. In this step, the computation latency is the digital signature validation of reply messages in the primary edge server. The communication latency is the message transmission from the validator edge servers.

8) Download Global Model

8) Download Global Model: The primary edge server sends back the validated global model to the edge devices, which promotes the next round of training. In this step, the communication latency is the global model downloading at the edge devices.

The steps of preprepare, prepare, commit, and reply are necessary in decentralized blockchain system to reach a consensus when each edge server does not know who is malicious. The consensus protocol ensures the correctness of global model aggregation by identity authentication and verification of calculation results at validator edge servers. The pseudocode of wireless B-FL is presented in Algorithm 1.

Algorithm 1 Pseudocode of Wireless B-FL

Input: Edge servers set $\mathcal{B}$, edge devices set $\mathcal{D}$, the dataset of edge devices $\mathcal{S}$.

Output: The global model $\boldsymbol{w}_g$ and a blockchain that contains training information.

Initialize the blockchain $\boldsymbol{B}$ and global model $\boldsymbol{w}_g^0$.
for each round $t = 1, 2, \ldots$ **do**
  Rotate primary edge server among edge servers.
  Allocate bandwidth and power resources.
  for each device $D_k \in \mathcal{D}$ **do**
    D_k executes local training on its local data $\mathcal{S}_k$ to generated local model $\boldsymbol{w}_k^t$.
    D_k uploads its local model to primary edge server.
  end for
  Primary edge server validates the local models $\{\boldsymbol{w}_k^t\}_{U_k \in \mathcal{U}}$ and aggregates them to a global model $\boldsymbol{w}_g^t = \text{multi\_KRUM}(\{\boldsymbol{w}_k^t\}_{U_k \in \mathcal{U}})$.
  Primary edge server packs local models and global model to a new block and broadcasts the block to the validator edge servers.
  Primary edge server and validator edge servers reach consensus by PBFT protocol consensus (including pre-prepare, prepare, commit, reply steps) to guarantee the correctness of global model and add verified block to the blockchain.
  Primary edge server broadcasts global model $\boldsymbol{w}_g^t$ to edge devices.
end for

function multi_KRUM($\boldsymbol{w}_{kD_k \in \mathcal{D}}^t$)
  Compute the Euclidean distances $s(D_k) = \sum_{D_j \neq D_k} \|\boldsymbol{w}_k - \boldsymbol{w}_j\|^2, \forall D_k \in \mathcal{D}$, where the sum runs over the $K - f - 2$ closest vectors and f is a specified parameter with respect to Byzantine tolerant.
  Select the $K - f$ lowest distance and corresponding local models as $\hat{\mathcal{D}} \subset \mathcal{D}$.
  Compute $\boldsymbol{w}_g = \frac{1}{|\mathcal{S}|} \sum_{D_k \in \hat{\mathcal{D}}} \boldsymbol{w}_k$ by averaging.
  return $\boldsymbol{w}_g$.
end function

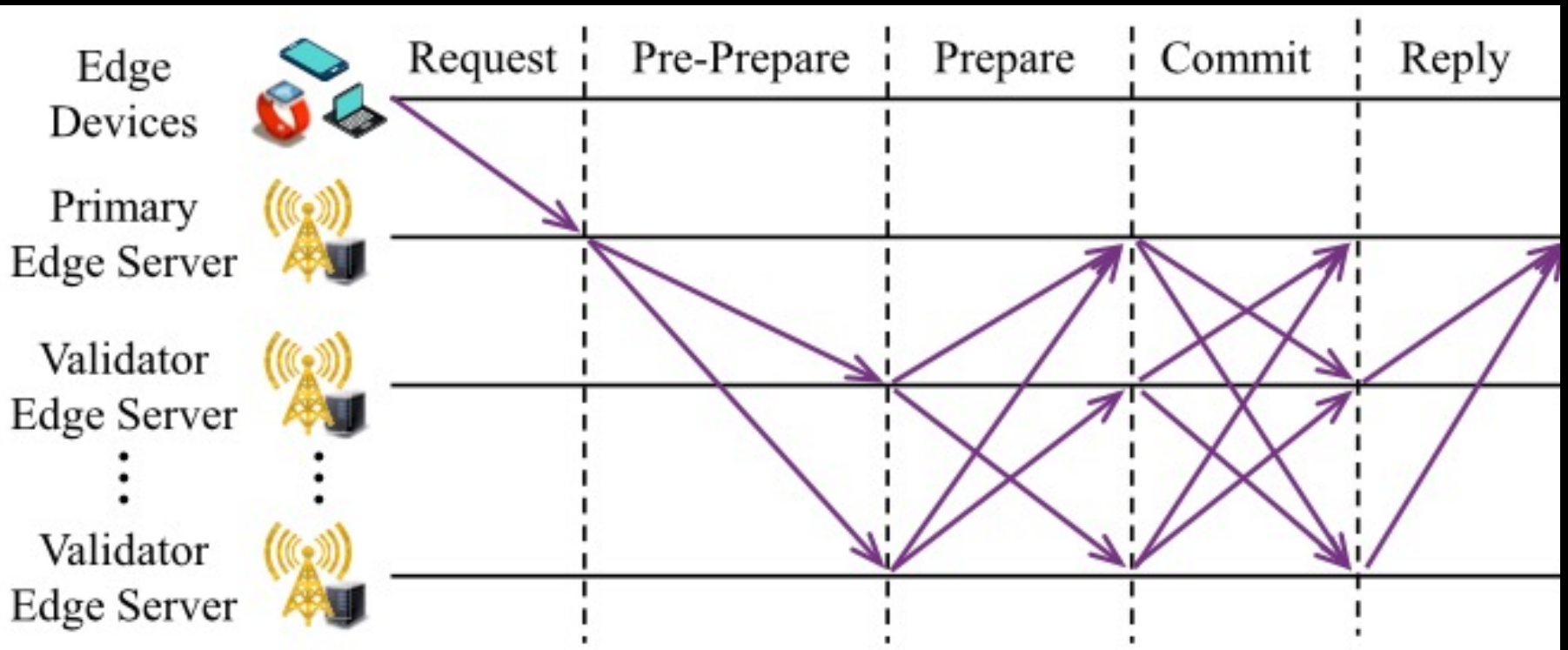


Fig. 3. PBFT consensus protocol.

C. Communication Model

The orthogonal frequency-division multiple access (OFDMA) technique is adopted in wireless B-FL system. All devices and edge servers are assumed to have a single antenna for simplification [14], [39], [44], which can be generalized to the multiantenna case by additionally allocating the power for each antenna of each server. We assume that the channel state information (CSI) is available for resource allocation and static in one time slot but vary from different time slots. To model the channel gain in each time slot, the channel gain between edge device D_k and edge server B_m of time slot s is defined by $h_{D_k,B_m}[s] = \zeta_{D_k,B_m}|g_{D_k,B_m}[s]|^2$, where ζ_{D_k,B_m} and $g_{D_k,B_m}[s]$ are large-scale path-loss attenuation and small-scale block fading between edge device D_k and edge server B_m, respectively. In particular, the large-scale path-loss attenuation is defined as $\zeta_{D_k,B_m} = d_{D_k,B_m}^{-\alpha}$, where d_{D_k,B_m} is the distance between edge device D_k and edge server B_m and α is the path-loss exponent. Based on Jakes' fading model [54], small-scale block fading $g_{D_k,B_m}[s]$ is modeled as first-order complex Gauss–Markov process, i.e.,

$$g_{D_k,B_m}[s] = \varrho g_{D_k,B_m}[s-1] + \sqrt{1-\varrho^2}\epsilon_{D_k,B_m}[s] \tag{5}$$

where $\varrho = J_0(2\pi f_d T_0)$ denotes the correlation coefficient in two consecutive time slots. $\epsilon_{D_k,B_m}[s]$ follows an independent and identically distributed (i.i.d.) circularly symmetric complex Gaussian distribution with unit variance. Here, $J_0(\cdot)$ is the zeroth-order Bessel function of the first kind, f_d represents the maximum Doppler frequency, and T_0 is the duration of one time slot.

In fact, the durations of time slots are very short (e.g., the frame time slot of the LTE protocol is 10 ms), but the duration of one B-FL training round is often in the time scale of second because of the high computation and communication complexity in the model training and consensus protocol. Therefore, each B-FL training round contains multiple time slots [27], [29]. Because the purpose of this article is minimization total latency from the long-term perspective with enormous rounds, the channel dynamics of multiple time slots in a single round is negligible. Thus, we employ the average channel gain instead of the instantaneous ones as the channel gain in one round to simplify the problem, i.e., $h^t_{D_k,B_m} = (1/S)\sum_{s=1}^{S} h_{D_k,B_m}[tS+s]$, which is channel gain between edge device D_k and edge server B_m in training round t and S is the total time slot in one round [29].

Thus, the achievable transmission rate from edge device D_k to edge server B_m is denoted by

$$R^t_{D_k,B_m} = b^t_{D_k}\log_2\left(1+\frac{h^t_{D_k,B_m}p^t_{D_k}}{b^t_{D_k}N_0}\right) \tag{6}$$

where $b^t_{D_k}$ is the transmission bandwidth assigned to edge device D_k at communication round t, and $h^t_{D_k,B_m}$ is the channel gain from edge device D_k to edge server B_m. Here, $p^t_{D_k}$ and N_0 represent the transmit power in edge device D_k and the power spectral density of the additive white Gaussian noise (AWGN), respectively. In this article, the maximum system bandwidth is denoted by $b^{\max}$, i.e., $\sum_{d\in\mathcal{D}\cup\mathcal{B}} b^t_d \le b^{\max}$, which means the edge servers and edge devices share the same bandwidth of broadband communication [14]. Therefore, the latency of transmitting data packet, whose size is ϖ, from device D_k to edge server B_m can be written as

$$T_{D_k,B_m} = \frac{\varpi}{R^t_{D_k,B_m}}. \tag{7}$$

In this article, we assume that the data packets are encoded into digital signals (e.g., polar code) and transmitted by rate adaptation scheme to ensure error-free transmission.

III. System Optimization

- A. Latency Analysis for Wireless B-FL
  - 1) Local Training
  - 2) Upload Local Models
  - 3) Global Model Aggregation
  - 4) Preprepare
  - 5) Prepare
  - 6) Commit
  - 7) Reply
  - 8) Download Global Model
- B. Problem Formulation and Analysis

A. Latency Analysis for Wireless B-FL

In the following, we give detailed steps and characterize the latency of each step in the tth round. We adopt synchronized transmission scheme in this article to ensure all edge servers and devices in the same step. Compared with the conventional wireless FL system without blockchain, the wireless B-FL system with blockchain needs to validate the digital signature of the local model and reaches a consensus of global model aggregation, which can be treated as the extra computation overhead. We assume that generating or verifying one digital signature requires ρ CPU cycles. The transmission processes of data packets in the PBFT consensus protocol are shown in Fig. 3, where the request step contains the upload local models step and global model aggregation step. The latency of eight steps in Section II-B is presented as follows.

1) Local Training: The number of CPU cycles for devices to execute backpropagation algorithm with one sample is denoted by δ. The local model weight calculation latency can be expressed as

$$T_{\text{train}}^{\text{cmp}} = \max_{D_k \in \mathcal{D}} \left\{ \frac{s_{D_k} \delta}{f_{D_k}} \right\} \tag{8}$$

where s_{D_k} is the batch size of edge device D_k and f_{D_k} is the CPU frequency of edge device D_k.

1) Local Training

2) Upload Local Models

2) Upload Local Models: After the local training, each updated local model is packed into a transaction, which consists of digital signature and local model. The computation latency of the digital signature is

$$T_{\text{up}}^{\text{cmp}} = \max_{D_k \in \mathcal{D}} \frac{\rho}{f_{D_k}}. \tag{9}$$

In this step, we define the average size of one transaction is denoted by ϖ, which is related to the AI model size. The transmission latency from the edge device to the primary edge server can be expressed by

$$T_{\text{up}}^{\text{com}} = \max_{D_k \in \mathcal{D}} \left\{ \frac{\varpi}{R_{D_k, B_p}^t} \right\} \tag{10}$$

where $B_p \in \mathcal{B}$ denotes the primary edge server, and R_{D_k, B_p} is the transmission rate from edge device D_k to the primary edge server B_p.

3) Global Model Aggregation

3) Global Model Aggregation: The transactions are submitted to the primary edge server. The primary edge server receives these transactions consisting of local models and verifies its digital signature to confirm the identity. Then, the primary edge server will execute smart contract to aggregate local models and generate global model. The smart contract consists of lines of code, which is immutable and trackable to execute a function. Therefore, the secure global model aggregation via the multi-KRUM algorithm can be programmed as a smart contract. After generating the global model, all the local model transactions and the global model transaction are packed into a new block.

The maximum number of transactions that can be packed in one block is $K+1$, where K is the number of edge devices, i.e., local model number, and one global model transaction. Thus, the block size can be presented as $S_B = (K+1)\varpi$ without considering the negligible size of the block header. In this step, the primary edge server needs to verify the digital signatures for K transactions. Then, the primary edge server executes the smart contract of secure global model aggregation, whose total CPU cycles can be defined as σ. Therefore, the total amount of computation at the primary edge server is $\Delta_{\text{req},B_p} = K\rho + \sigma$. Then, the computation delay at the primary edge server is

$$T_{\text{agg}}^{\text{cmp}} = \frac{\Delta_{\text{req},B_p}}{f_{B_p}}. \tag{11}$$

4) Preprepare

4) Preprepare: After generating the new block, the primary edge server will broadcast a preprepare message to all the validator edge servers for validation. The preprepare message consists of the digital signature of the primary edge server and the information of the new block. The validator edge servers, which are all edge servers except the primary edge server, need to validate that the new block are generated by the given primary edge server and validate the identities of all transactions to ensure the new block is valid. Besides, the validator edge servers are mainly responsible for the correctness validation of the global model, which can use an intuitive method by repeatedly calculating the global model through identified local models, and comparing the computation results with the global model of the primary edge server.

If the preprepare message is validated by $2f$ validator edge servers, it enters the next step. In particular, f is the number of hypothetical malicious edge servers, which satisfy $3f + 1 = M$ [31]. PBFT is designed to ensure the correctness of 2/3 edge servers, which is more efficient than validating all edge server but less secure. It is a tradeoff between efficiency and security.

In this step, the latency can be presented as

$$T_{\text{prep}}^{\text{com}} = \max_{B_m \in \mathcal{B}/B_p} \left\{ \frac{S_B}{R_{B_p,B_m}^t} \right\}. \tag{12}$$

The amount of computation at the validator edge servers is $\Delta_{\text{prep},B_m} = \rho + (K+1)\rho + \sigma$, where there is $B_m \neq B_p$. The computation latency in this step is

$$T_{\text{prep}}^{\text{cmp}} = \max_{B_m \in \mathcal{B}/B_p} \left\{ \frac{\Delta_{\text{prep},B_m}}{f_{B_m}} \right\}. \tag{13}$$

5) Prepare

5) Prepare: After accepting the new block generated by the primary edge server, each validator edge server will broadcast a prepare message to all the other edge servers consisting primary edge server and validator edge servers. Each edge server will validate the prepare message to ensure the consistent with the preprepare message.

In this step, the communication latency consisting of broadcasting the prepare message to all other validator edge servers, which is calculated by

$$T_{\text{pre}}^{\text{com}} = \max_{B_m \in \mathcal{B}/B_p, B_{m'} \in \mathcal{B}, B_m \neq B_{m'}} \left\{ \frac{S_M}{R_{B_m, B_{m'}}^t} \right\} \tag{14}$$

where S_M denotes the size of prepare message, which is smaller than S_B because it does not contain the new block.

For the computation, the primary edge server needs to validate $2f$ digital signatures from the other validator edge servers, which can be expressed by $\Delta_{\text{pre},B_p} = 2f\rho$. Each of validator edge servers needs to generate one digital signature for the prepare message. Then, $2f$ digital signatures are needed to be validated. Thus, the amount of computation at the validator edge servers $B_m (B_m \neq B_p)$ can be given by $\Delta_{\text{pre},B_m} = \rho + 2f\rho$. The computation latency in this step is

$$T_{\text{pre}}^{\text{cmp}} = \max_{B_m \in \mathcal{B}} \frac{\Delta_{\text{pre},B_m}}{f_{B_m}}. \tag{15}$$

6) Commit

6) Commit: After receiving $2f$ valid prepare messages from the other validator edge servers, each edge server will broadcast a commit message to all the other edge servers. The communication latency can be characterized by

$$T_{\text{cmit}}^{\text{com}} = \max_{B_m \in \mathcal{B}/B_p, B_{m'} \in \mathcal{B}, B_m \neq, B_{m'}} \left\{ \frac{S_M}{R^t_{B_m, B_{m'}}} \right\}. \tag{16}$$

In this step, each edge server needs to generate one digital signature to construct the commit messages. After receiving the commit messages, each edge server needs to verify $2f$ digital signatures. Thus, the amount of computation at each edge server is $\Delta_{\text{cmit},B_m} = \rho + 2f\rho$. The computation latency in this step is

$$T_{\text{cmit}}^{\text{cmp}} = \max_{B_m \in \mathcal{B}} \left\{ \frac{\Delta_{\text{cmit},B_m}}{f_{B_m}} \right\}. \tag{17}$$

7) Reply

7) Reply: After collecting $2f$ commit messages, the new block is ensured to a valid block, and it will be appended to the blockchain. A reply message will be transmitted to primary edge server. In this step, the communication latency is

$$T_{\text{rep}}^{\text{com}} = \max_{B_m \in \mathcal{B}/B_p} \left\{ \frac{S_M}{R_{B_m,B_p}^t} \right\}. \tag{18}$$

For the computation, each validator edge server needs to generate one digital signatures for the primary edge server, which can be given by $\Delta_{\text{rep},B_m} = \rho$. For the primary edge server, it needs to verify $2f$ signatures, the amount of computation of which is given by $\Delta_{\text{rep},B_p} = 2f\rho$. The computation latency in this step is

$$T_{\text{rep}}^{\text{cmp}} = \max_{B_m \in \mathcal{B}} \left\{ \frac{\Delta_{\text{rep},B_m}}{f_{B_m}} \right\}. \tag{19}$$

8) Download Global Model

8) Download Global Model: After local mode validation, global model aggregation, and the new block addition, the global model is transmitted to the edge devices to enter the next round. In this step, the communication latency is

$$T_{\text{down}}^{\text{com}} = \max_{D_k \in \mathcal{D}} \left\{ \frac{\varpi}{R_{B_p, D_k}^t} \right\}. \tag{20}$$

From the above analysis, the total latency in the t-th round consists of communication latency and computation latency, i.e.,

$$T(\boldsymbol{b}^t, \boldsymbol{p}^t) = T^{\text{com}} + T^{\text{cmp}}. \tag{21}$$

The communication time consumption can be calculated as

$$T^{\text{com}} = T_{\text{up}}^{\text{com}} + T_{\text{prep}}^{\text{com}} + T_{\text{pre}}^{\text{com}} + T_{\text{cmit}}^{\text{com}} + T_{\text{rep}}^{\text{com}} + T_{\text{down}}^{\text{com}} \tag{22}$$

and the computation time consumption is

$$T^{\text{cmp}} = T_{\text{train}}^{\text{cmp}} + T_{\text{up}}^{\text{cmp}} + T_{\text{agg}}^{\text{cmp}} + T_{\text{prep}}^{\text{cmp}} + T_{\text{pre}}^{\text{cmp}} + T_{\text{cmit}}^{\text{cmp}} + T_{\text{rep}}^{\text{cmp}}. \tag{23}$$

B. Problem Formulation and Analysis

In this article, we aim to minimize the long-term average latency of wireless B-FL by optimizing bandwidth allocation, and transmit power allocation from a long-term perspective. We assume that the wireless B-FL training process is finished in the τth training round. The long-term average latency minimization problem is given by

$$\begin{aligned} \underset{\{\boldsymbol{b}^t,\boldsymbol{p}^t\}_{t\in\mathcal{T}}}{\text{minimize}} \quad & \frac{1}{\tau}\sum_{t=1}^{\tau} T\left(\boldsymbol{b}^t,\boldsymbol{p}^t\right) \\ \text{subject to} \quad & \sum_{d\in\mathcal{D}\cup\mathcal{B}} b_d^t \le b^{\max}, b_d^t \ge 0 \ \forall t \in \mathcal{T} \qquad \text{(24a)} \\ & \frac{1}{\tau}\sum_{t=1}^{\tau}\sum_{d\in\mathcal{B}\cup\mathcal{D}} p_d^t \le \bar{p}, p_d^t \ge 0 \qquad \text{(24b)} \end{aligned}$$

where $\mathcal{T} = \{1, 2, \ldots, \tau\}$ denotes the training round. Here, (24a) restricts the maximum system bandwidth and (24b) denotes the long-term average power constraint. The long-term average power constraint can provide a dynamic power allocation scheme to assign more transmit power in worse channel conditions and less power in better channel conditions.

Due to the correlation in the long-term average power constraint, problem (24) is a long-term resource allocation problem, which cannot be equivalently transformed to multiple one-shot problems with respect to the channel of each round. Moreover, the optimization variables of the high-dimensional optimization problem are coupled in the objective function and constraints. As a result, the conventional optimization-based algorithms suffer from high computational complexity, which is unacceptable in the time-sensitive B-FL system [33]. Meanwhile, although the supervised learning, and unsupervised learning-based resource allocation schemes can solve the one-shot optimization problem to provide significant reduction on computational complexity, the suboptimal performance may occur in the long-term resource allocation problem [55].

In this article, we proposed a DRL-based algorithm to minimize the long-term cumulative latency by modeling problem (24) as analyze MDP. In particular, the DRL can optimize problem (24) from a long-term perspective by balancing the latency in current round and that in the future rounds while achieving significant reduction on computational complexity by establishing a direct mapping from the current network information to resource allocation decisions as DNN-based policy. Besides, the network information (i.e., channel conditions, current latency) for the decision making and policy update of DRL agents is obtained by interacting with the dynamic B-FL system, which enables the learned policy to keep track of the environment dynamic. To this end, we shall formulate the MDP for problem (24), and then we shall develop a TD3-based algorithm to make decisions in continuous action spaces, which will be described in detail in the next section.

IV. TD3-Based Resource Allocation Algorithm

- A. MDP Formulation
- B. TD3 Algorithm
- C. Complexity Analysis

In this section, we first formulate the optimization problem as an MDP, and then propose a TD3-based learning algorithm to solve the MDP problem efficiently.

IV. TD3-Based Resource Allocation Algorithm

A. MDP Formulation

A. MDP Formulation

To obtain optimal decisions under the stochastic environment, we reformulated the optimization problem (24) into an MDP, which can be solved by a DRL-based scheme. The key components of the MDP include the observed state, action, and reward, which are summarized as follows.

1) *State:* At each training round, the edge server collects CSI between edge servers and device devices and cumulative latency, which is necessary for resource allocation. Therefore, The state at time step t is determined by the cumulative latency and CSI in the wireless communication system, which is denoted by s^t. Particularly, the state contains the channel gain between edge devices and primary edge server, i.e., $\{h^t_{d,B_p}\}\ \forall d \in \mathcal{D}$, which has K entries. Besides, the state contains the channel gain between different validator edge servers, i.e., $\{h^t_{d_1,d_2}\}\ \forall d_1, d_2 \in \mathcal{B}, d_1 \neq d_2$, which has $M(M-1)$ entries. Therefore, the state is presented as

$$s^t = \left\{ \sum_{i=0}^{t-1} T(\boldsymbol{b}^i, \boldsymbol{p}^i), \left\{h^t_{d,B_p}\right\}\ \forall d \in \mathcal{D} \right. \\ \left. \left\{h^t_{d_1,d_2}\right\}\ \forall d_1, d_2 \in \mathcal{B}, d_1 \neq d_2 \right\}. \quad (25)$$

Thus, the dimension of the state space is $K + M(M-1) + 1$. The proposed DRL-based resource allocation algorithm can be directly applied when other correlated wireless channel models are considered. However, the uncorrelated channel (e.g., Rayleigh channel) will decrease the correlation of system state in two consecutive time, which may affect the estimation accuracy of the expected future reward and decrease the performance of resource allocation decisions [54].

2) *Action:* From the observed environment states, the agent chooses optimal action, i.e., making decisions based on the policy π. The action is constructed by the bandwidth allocation $\boldsymbol{b}^t \in \mathbb{R}^{M+K}$ and transmit power allocation $\boldsymbol{p}^t \in \mathbb{R}^{M+K}$ of all edge servers and edge devices at training round t, i.e.,

$$a^t = \{\boldsymbol{b}^t, \boldsymbol{p}^t\}. \quad (26)$$

Therefore, the dimension of the action space is $2(M+K)$.

A. MDP Formulation

3) *Reward:* After the action a^t has been taken by the environment, the DRL agent will obtain a reward from the environment and update the policy π to maximize the reward in the future. To minimize the latency in the wireless B-FL training process, the reward at the tth step is defined by negative latency, i.e., maximum reward means minimum latency, which is presented as

$$r^t = \begin{cases} -T(\boldsymbol{b}^t, \boldsymbol{p}^t), & \text{when (24a), (24b) are satisfied} \\ r_p, & \text{otherwise} \end{cases} \tag{27}$$

When the constraints (24a) and (24b) are satisfied, the immediate reward is the negative of the latency in this round. When the constraints (24a) and (24b) are not satisfied, the immediate reward is set to r_p. r_p is an extremely small value, which is regarded as a penalty. Therefore, the actions selected by the agent will satisfy the constraints (24a) and (24b).

4) *MDP Optimization Problem*: The action-value function is defined as

$$Q_{\pi_\phi}(s^t, a^t) = \mathbb{E}_{\pi_\phi}\left[\sum_{i=0}^{\infty} \gamma^i r^{t+i} | s^t, a^t\right] \tag{28}$$

where $\sum_{i=0}^{\infty} \gamma^i r^{t+i}$ denotes the total discount reward from the ith training round and $\gamma \in [0, 1]$ represents the discount factor. Therefore, the action-value function expresses a "myopic" evaluation if $\gamma \to 0$ or "far-sighted" evaluation if $\gamma \to 1$. The action-value function can evaluate the action selected by a policy π_ϕ with parameters ϕ. Moreover, the action-value function can be rewritten as the Bellman equation, i.e.,

$$Q_{\pi_\phi}(s^t, a^t) = \mathbb{E}_{\pi_\phi}\left[r^t + \gamma Q_{\pi_\phi}\left(s^{(t+1)}, a^{(t+1)}\right) | s^t, a^t\right]. \tag{29}$$

Therefore, the optimal policy can be obtained by maximizing the action-value function to solve the MDP problem, i.e.,

$$\pi_* = \arg\min_{\pi_\phi} Q_{\pi_\phi}(s^t, a^t). \tag{30}$$

The resource allocation problem (24) is reformulated as an MDP without having state transition probabilities $P(s^{t+1}|s^t, a^t)$ in prior due to the complexity and dynamics of wireless B-FL systems. Therefore, we resort to adopting model-free reinforcement learning based on value function approximation, which does not require the transition probability distribution to make decisions.

B. TD3 Algorithm

From the definition of the MDP problem, the state space and action space are continuous, which indicates that infinite dimension of action cannot be solved by Q-learning or deep Q network. Recently, the deep deterministic policy gradient (DDPG) algorithm has achieved considerable performance for solving the MDP problem with continuous action variables. Moreover, the enhanced version of DDPG, i.e., TD3 [56], can reduce the overestimation bias of the Q value and avoid suboptimal policy updates, which improves the policy learning speed and performance in continuous control domains. The TD3 is widely used to solve complex resource allocation problem in real time. We thus propose a TD3-based resource allocation algorithm to solve the MDP problem of B-FL system in this article. The TD3 algorithm is presented in Algorithm 2 in detail.

Particularly, the actor–critic architecture is adopted in TD3 algorithm, which combines the value-based and policy-based reinforcement learning algorithm and is shown in Fig. 4. In the actor–critic architecture, the actor will output the selected actions in continuous space from the input observed state, i.e., the policy $a^t = \pi_\phi(s^t)$. The critic is used to approximate the action-value function, which outputs the Q value from the inputs selected action and current state, i.e., $Q(s^t, a^t)$. Due to the complexity of the mapping function of actor and critic, and vector structure of state and action, the actor and critic are implemented by DNNs, which consist of several full-connected layers of neurons to construct the feature extractor and the output layer. Then, we introduce the training processes of actor–critic architecture and update rules of the TD3 algorithm as follows.

1) Critic Network Training: The critic network aims to estimate the real Q table in continuous state and action spaces as shown in Fig. 5(a), which approximates the mapping function from state and action values to Q value, i.e., $Q_\theta(s^t, a^t)$. Therefore, the loss function of the critic network is the mean square error (MSE) between the estimated value and the target value, i.e.,

$$L(\theta) = \mathbb{E}\Big[\big(y - Q_\theta(s^t, a^t)\big)^2\Big] \tag{31}$$

where y is the target value and $Q_\theta(s^t, a^t)$ is the estimated value under DNN parameter θ. Especially, y is generated by the target critic network, i.e.,

$$y = r^t + \gamma Q_{\theta'}\left(s^{t+1}, a'^{t+1}\right) \tag{32}$$

where θ' is the DNN parameter in the target network of critic network and γ is the discount factor. θ' is the periodically updated parameters of the target critic $Q_{\theta'}(s^t, a'^t)$, and a'^{t+1} is the action taken in the next state s^{t+1} selected by the target actor $\pi_{\phi'}$.

However, the update of the value function with the above target value have an overestimate bias, which will also affect the accurate update of the actor [56]. To address the bias problem, the TD3 uses two approximately independent target critics $\{Q_{\theta_1}, Q_{\theta_2}\}$ to estimate the value function, in which Q_{θ_1} is a biased estimator and Q_{θ_2} is an estimator with less overestimate bias. The minimum of the two estimators is used in the update of the value function, i.e.,

$$y = r^t + \gamma \min\left\{Q_{\theta_1'}\left(s^{t+1}, a'^{t+1}\right), Q_{\theta_2'}\left(s^{t+1}, a'^{t+1}\right)\right\}. \tag{33}$$

Under the loss function in (31), the update rules of the two critic network can be written as

$$\theta_i = \theta_i - \eta_c \nabla_{\theta_i} L(\theta_i)\ \forall i = 1, 2 \tag{34}$$

where η_c is the learning rate of the critic network, and $\nabla_{\theta_i} L(\theta_i)$ is the gradient of the ith critic network, i.e.,

$$\nabla_{\theta_i} L(\theta_i) = -\mathbb{E}_{s^t}\big[\big(y_j - Q_i(s_j, a_j|\theta_i)\big)\nabla_{\theta_i} Q_i(s_j, a_j|\theta_i)\big]. \tag{35}$$

B. TD3 Algorithm

2) Actor Network Training: The actor network aims to select optimal actions in continuous state and action spaces as shown in Fig. 5(b), which approximates the mapping function from state to action, i.e., $a^t = \pi_\phi(s^t)$. The output layer of the actor network is designed to have neurons for bandwidth allocation and power allocation. The first set of output neurons is implemented by a softmax function, which guarantees the sum of output is equal to 1 to represent the percentage of bandwidth allocation. The second set of output neurons is implemented by Sigmoid function, which guarantees each output is between 0 and 1 to represent the percentage of the maximum transmit power. The optimal action is selected to achieve the maximum $\mathbb{E}[Q_1(s^t, a^t|\theta_1)]$, which utilizes the DNN parameter of the first critic network to update the actor network. Accordingly, the actor can be updated using the deterministic policy gradient algorithm (DPG), i.e.,

$$\nabla_\phi J(\phi) = \mathbb{E}_{\pi_\phi} \nabla_a \left[Q_{\pi_\phi} \left(s^{t+1}, a^{t+1} \right)|_{a=\pi_\phi(s^t)} \nabla_\phi \pi_\phi(s^t) \right] \quad (36)$$

where $\pi_\phi(s^t)$ is the actor network with DNN parameter ϕ. Therefore, the update rule of the actor network is

$$\phi = \phi - \eta_a \nabla_\phi J(\phi) \quad (37)$$

where η_a is learning rate of actor network.

3) Target Networks Updating: The two critic networks and one actor network have corresponding target networks, which are called target critic networks and target actor network. The target critic networks aim to generate target values, which approximates real Q values. The target actor network is used to select actions. In other words, the critic networks and actor network, also known as online networks, are used to update the DNN parameters in each step, and the target networks are assumed as a real Q value and optimal action. The DNN parameters of target networks are updated based on the online networks with delayed method, i.e., the target networks update after several steps of online networks. In general, the update rule of target networks is

$$\theta_1' = \kappa\theta_1 + (1-\kappa)\theta_1' \quad (38)$$

$$\theta_2' = \kappa\theta_2 + (1-\kappa)\theta_2' \quad (39)$$

$$\phi' = \kappa\phi + (1-\kappa)\phi' \quad (40)$$

where κ is update proportion, i.e., the proportion of DNN parameters of online networks.

In summary, there are two types of networks, i.e., two critic networks and one actor network. Each of them contains two subnets, i.e., online network and target network. These six DNNs consist various layers and all layers contain their corresponding parameters. Actor network parameters $\{\phi, \phi'\}$ denote the online network parameter and target network parameter, respectively. Critic network parameters $\{\theta_1, \theta_2, \theta_1', \theta_2'\}$ indicate online network parameter and target network parameter of the two critic networks, respectively.

Besides, the TD3 uses a reply memory buffer $\mathcal{R}$ to store the experience (s^t, a^t, r^t, s^{t+1}) from the interaction between agent and environment. In the training process, a minibatch of experience is randomly selected from the reply memory buffer $\mathcal{R}$ to update the parameters of the online networks. Particularly, the online critic networks are updated by minimizing the loss function (31) and the online actor network is updated based on (37). After every ϑ steps, the parameters of targets networks are updated (38)–(40) with update proportion δ. The pseudocode of the TD3-based resource allocation algorithm is presented in Algorithm 2.

We assume the number of edge servers and edge devices is invariable during the FL training process. The B-FL can still be applied when the edge servers or edge devices exit the system before the completion of FL, but suffer from system security and global model accuracy degradation. The DRL-based resource allocation algorithm is compatible with the case of moving edge servers or edge devices. When the edge servers or edge devices exit the system, the relevant parameters in the states and actions are set to be null to cope with the smaller number of edge servers or edge devices [14]. However, if the number of edge servers and edge devices increases, the DRL-based resource allocation algorithm needs to be retrained, because the output layer dimension of the actor network increases.

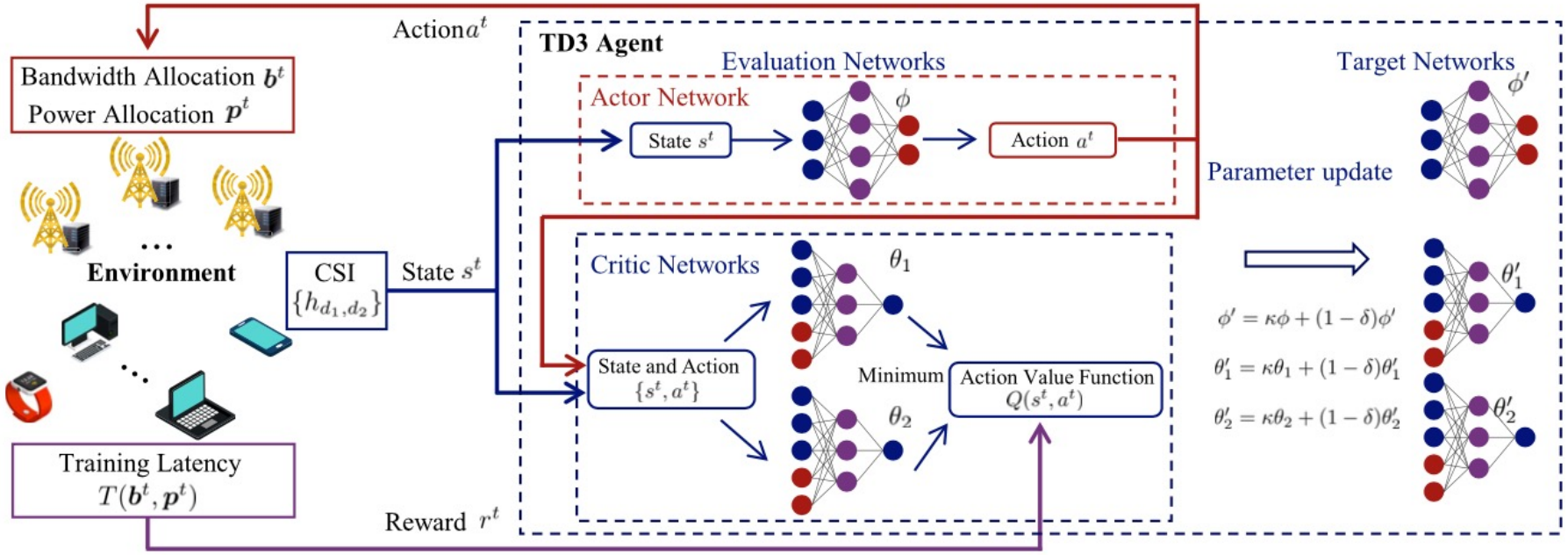


Fig. 4. Framework of the TD3 algorithm.

Algorithm 2 TD3-Based Bandwidth and Transmit Power Allocation Algorithm

Output: Actor network π_ϕ and critic networks $\{Q_{\theta_1}, Q_{\theta_2}\}$

Initialize critic networks $\{Q_{\theta_1}, Q_{\theta_2}\}$and actor network π_ϕ with random parameters $\theta_1, \theta_2, \pi_\phi$.
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$.
Initialize replay buffer $\mathcal{R}$.
Initialize noise range c, noise variance σ_1^2, σ_2^2, update proportion δ, discount factor γ.
Explore E time steps with random policy, obtain the reward r^t and the next state s^{t+1}, and storage the transition (s^t, a^t, r^t, s^{t+1}) of the E steps into $\mathcal{R}$.
while time step $\leq P$ **do**
    Select action with exploration noise $a^t = \pi_\phi(s^t) + n_1, n_1 \sim \mathcal{N}(0, \sigma_1^2)$.
    Obtain bandwidth allocation $\boldsymbol{b}^t$ and transmit power allocation $\boldsymbol{p}^t$ from action a^t, and assign them to edge servers and device devices.
    Edge devices and edge servers begin B-FL training process and get the latency in one round.
    Calculate reward r^t, update state s^{t+1}, and storage the transition (s^t, a^t, r^t, s^{t+1}) into $\mathcal{R}$.
    Sample a mini-batch of transitions from $\mathcal{R}$.
    Obtain target action $a'^{t+1} = \pi_{\phi'}(s^{t+1}) + n_2, n_2 \sim \text{clip}(\mathcal{N}(0, \sigma_2^2), -c, c)$.
    $y = r^t + \gamma \min_{i=1,2} Q_{\theta_i'}(s^{t+1}, a'^{t+1})$.
    Update critics $\theta_i \leftarrow \arg\min_{\theta_i} \mathbb{E}\Big[(y - Q_{\theta_i}(s^t, a^t))^2\Big]$.
    Update ϕ using the deterministic policy gradient algorithm.
    if step mod $\vartheta = 0$ **then**
        Update the target networks:
        $\theta_1' = \kappa\theta_1 + (1-\kappa)\theta_1'$,
        $\theta_2' = \kappa\theta_2 + (1-\kappa)\theta_2'$,
        $\phi' = \kappa\phi + (1-\kappa)\phi'$.
    end if
    $t \leftarrow t + 1$
end while

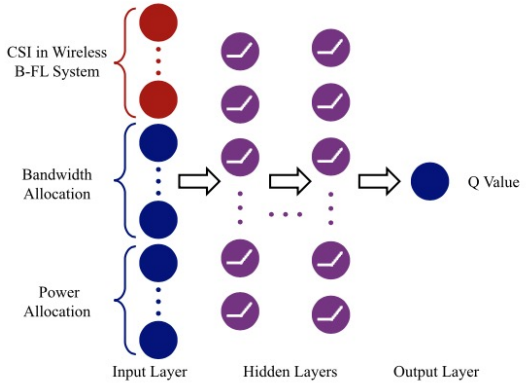


Fig. 5. Proposed structures of (a) critic and (b) actor networks in the TD3 algorithm.

C. Complexity Analysis

C. Complexity Analysis

Based on the description of the TD3 algorithm, the training process of DRL for the proposed B-FL systems can be performed offline with simulated channel states to train the actor network and critic network. Then, the trained actor network models are deployed online to allocate the wireless resources based on the system state to minimize the long-term latency of B-FL. Furthermore, the experience (s^t, a^t, r^t, s^{t+1}) generated by an online actor network can be collected, which is used to train and update the actor network.

In the training phase, two actor networks and four critic networks need to be trained, the most significant complexity is caused by backpropagation. Hence, the computational complexity is $\mathcal{O}(2\sum_{l=1}^{L^a} z_l^a z_{l+1}^a + 4\sum_{l=1}^{L^c} z_l^c z_{l+1}^c)$, where L^c is the number of layers in the critic network. Let z_l^c denote the number of neurons in the lth layer of the critic network. Hence, $z_1^c = K+M(M-1)+1+2(M+K)$ is the number of neurons in the input layer and $z_L^a = 1$ is the number of neurons in the output layer. Since the proposed algorithm selected Q minibatch transitions to train the actor networks and critic networks and takes P training step in total, therefore, the total computational complexity is $\mathcal{O}(PQ(2\sum_{l=1}^{L^a} z_l^a z_{l+1}^a + 4\sum_{l=1}^{L^c} z_l^c z_{l+1}^c))$.

In the deployment phase, for each single decision-making, the complexity only caused by the actor network, which can be calculated as $\mathcal{O}(\sum_{l=1}^{L^a} z_l^a z_{l+1}^a)$, Let L^a be the number of layers in the actor network. z_l^a denotes the number of neurons in the lth layer of the actor network. Hence, $z_1^a = K+M(M-1)+1$ is the number of neurons in the input layer and $z_L^a = 2(M+K)$ is the number of neurons in the output layer.

V. Numerical Results and Analysis

- A. Settings and Benchmarks
- B. Performance of Wireless B-FL
- C. Convergence of TD3-Based Resource Allocation Algorithm
- D. Performance Impact of Different System Settings

V. Numerical Results and Analysis

V. Numerical Results and Analysis

In this section, various simulation results are presented to show the performance of the wireless B-FL and related TD3-based resource allocation algorithm. We first introduce the simulation settings and then show the results of the proposed algorithms.

A. Settings and Benchmarks

A. Settings and Benchmarks

In the following simulations, we consider a wireless B-FL system consisting of several edge servers and edge devices to execute the ML task, which guarantees security and privacy in the training process based on the blockchain technique. The details of the system model and simulation scenario are defined as follows.

1) System Settings: In our simulation, we consider a circular wireless network with $M = 4$ edge server serving for $K = 10$ edge devices, and the location of each edge server or edge device is uniformly distributed in a circle with a radius of 100 m. The duration of time slot T_0 is set as 10 ms according to the LTE standard and Doppler frequency f_d is 5 Hz due to the assumption of low mobility in B-FL system. The path-loss exponent α is 2.5. In general, we assume that the maximum CPU frequencies of edge servers and devices are 2.4 and 1 GHz, respectively. The maximum system transmission powers are set as 24 dBm and the maximum system bandwidth is 100 MHz. The noise power spectral density is $N_0 = -174$ dBm/Hz.

2) MINST Data Set for Handwriting Recognition: To simulate the performance of FL and B-FL, we consider training a convolutional neural network (CNN) model based on the MINST data set, which is a standard data set used for testing the ML algorithm and consists of 60 000 training handwritten image samples and 10 000 test handwritten image samples. The handwritten image samples are 28×28 grayscale images denoting a handwritten number within 0–9. The data set gives a typical and proper complexity in ML tasks. We design the CNN model by two 5×5 convolutional layers (10 channels and 20 channels, respectively), two 2×2 max-pooling layers, two fully connected layers, and two dropout layers. The rectified linear unit (ReLU) activation is used in each layer and the softmax activation is employed in output layer. Therefore, each device randomly has 6000 training samples and 1000 test samples. In the training process, each device executes two local training epochs with 128 batch size and the learning rate is set as 0.01.

3) CIFAR-10 Data Set for Image Classification: To verify the performance of B-FL in the practical real-word data set, we utilize a more popular and standard CIFAR-10 data set and train the AlexNet model. The CIFAR-10 consists of 50 000 training 32×32 color images in ten classes and 10 000 test images. The AlexNet is a typical CNN model with five convolutional layers and max pooling and three fully connected layers. Therefore, each device randomly has 5000 training samples and executes one local training epoch with 128 batch size and the learning rate is set as 0.01.

4) Heart Activity Data Set for Affect Recognition: More and more researches have shown great interest to improve the health of individuals via prevention and diagnostics by using sensors of Internet of Medical Things (IoMT). In particular, affect recognition of stress is remarkable to improve the health of individuals based on the biomedical informatics collected from wearable IoT devices. In this article, we utilize the preprocessed data set of 26 individuals from the paper [57]. The data set consists of heart activity samples range from 60 to 125, which means a nonindependent identically distributed (non-iid) data set. Each sample has a 16 dimension feature vector and a label to indicate low stress and high stress, which means it is a 2-class stress-level classification. We design the fully connected neural network (FNN) model by two hidden layers with ReLU activation and one output layer with sigmoid activation. Each hidden layer is composed of 100 neurons. In the training process, we set that the batch size is 32, the local learning rate is 5e-6 and local training epoch is 1. Furthermore, we partition the 26 individuals as 20 training edge devices to train a shared global model and six test edge devices to evaluate the accuracy of global model.

A. Settings and Benchmarks

5) TD3 Algorithm Settings: Based on the details of the TD3 algorithm in Section IV, we implement the proposed algorithm by TensorFlow in a general computer with CPU Intel Xeon E5-2643 v4 and GPU Nvidia GeForce GTX 1080 Ti. The actor network is designed by five hidden layers, which has 512, 1024, 2048, 1024, and 512 neurons with ReLU activation, respectively. The output layer is a softmax layer to output action with constrains. The critic network is constructed by four hidden layers, which has 512, 1024, 512, and 512 neurons with ReLU activation, respectively. The output layer is a linear layer to output the Q value. The hyperparameters in the TD3 algorithm are set as follows, i.e., the size of the replay buffer is 10^6, update proportion $\kappa = 5 \times 10^{-3}$, discount factor $\gamma = 0.99$, steps of exploration $E = 512$, update frequency $\vartheta = 2$, and the learning rate of the actor and critic networks is $\eta_a = \eta_c = 1 \times 10^{-4}$. We train the networks with max steps 5000, which can achieve stable performance in our experiments.

6) Benchmarks: To verify the effectiveness of our proposed TD3-based resource allocation algorithm, we choose three other algorithms to compare as follows.

1) *Random Allocation:* The random allocation scheme allocates the bandwidth and transmit power to edge servers and devices from uniformly random distribution.
2) *Average Allocation:* The average allocation scheme uniformly allocates the bandwidth and transmit power to edge servers and devices, i.e., all edge servers and edge devices are assigned with the same resources.
3) *Monte Carlo Algorithm:* The Monte Carlo algorithm randomly samples C allocation solutions and choose the best one, i.e., the latency is the smallest. Therefore, the best allocation solution is close to the global optimal solution if the C is large enough. We set $C = 10^6$ in this article.

B. Performance of Wireless B-FL

B. Performance of Wireless B-FL

We simulate the processes of B-FL following Algorithm 1. The malicious edge device is assumed to upload local models with the random DNN parameter, which follow normal distribution $\mathcal{N}(0, 1)$ and will obviously influence the convergence performance of FL training [13]. Besides, we use the FedAvg algorithm as a global model aggregation algorithm in the FL training process [2].

In the handwriting recognition task, we set that the percentage of malicious edge devices ranges from 0% to 100% and train CNN in 100 global epochs. The results are shown in Table II, which shows the malicious edge devices will obviously decrease the accuracy. The proposed B-FL in this article shows robustness for the malicious edge devices, which cannot be affected below 50%. The partial loss and accuracy of training rounds are shown in Figs. 6 and 7. The results show that the proposed B-FL with 40% malicious devices and FL with 0% malicious devices have the same performance, which is because the secure global aggregation based on the multi-KRUM algorithm can eliminate partial local models from malicious devices. Therefore, the global model is not affected by malicious local models. Furthermore, the influence of the number of aggregated local models is inconspicuous in this simulation settings.

In image classification task, we set the percentage of malicious edge devices as 0%, 20%, and 40%. The test loss and test accuracy are shown in Fig. 8, and Fig. 9, which indicate that the malicious devices yield significant influence on the FL training process. However, the proposed B-FL with 40% malicious edge devices can still maintain the similar accuracy with FL with 0% malicious edge devices, which is mainly because the proposed B-FL utilizes secure global aggregation based on the multi-KRUM algorithm to eliminate local models from malicious devices.

In the affect recognition task, we assume the FL and B-FL systems have 10% malicious edge devices to attack global model, respectively, and present the performance of centralized FL and proposed B-FL in different settings. The loss and accuracy of training rounds are shown in Figs. 10 and 11. The

B. Performance of Wireless B-FL

proposed B-FL can conspicuously resist the attack of malicious edge devices and achieve excellent performance. We just plot the scenario of 10% malicious edge devices in this task, which is because the test loss is diverging when the number of malicious edge devices continues to increase.

On the other hand, the situation of malicious edge servers is also considered in this article. It is obvious that the FL training process will be discontinued or destroyed completely when the only edge server is malicious. However, this will not happen in B-FL system, which has multiple edge servers to reach consensus and produce a global model. When the primary edge server is malicious, the other validator edge servers will not accept its block and global model and will choose another new primary edge server. If less than 1/3 validator edge servers are malicious, the consensus cannot be destroyed because of the mechanism of PBFT. If more than 1/3 validator edge servers are malicious, the consensus will be destroyed and the ML training process will be discontinued or destroyed completely. However, the situation of more than 1/3 malicious validator edge servers is almost impossible because the authorization is generally required for validator edge servers to participate in the consensus.

TABLE II
ACCURACIES OF FL AND B-FL WITH THE MINST DATA SET OVER DIFFERENT PERCENTAGE OF MALICIOUS DEVICES

Malicious edge devices percentage	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
FL with FedAvg	97.92	95.59	92.38	89.67	86.42	11.35	11.35	11.35	11.35	10.28	9.74
B-FL with multi-KRUM	97.93	97.63	97.93	97.80	97.90	93.74	88.14	11.35	11.35	9.80	9.74

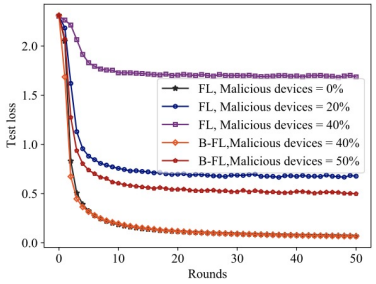


Fig. 6. Test loss of FL and B-FL with malicious devices for handwriting recognition.

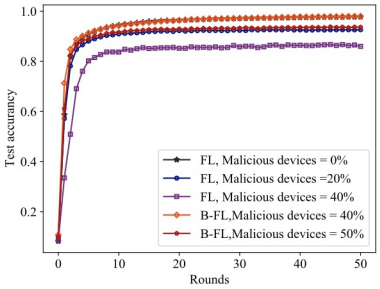


Fig. 7. Test accuracy of FL and B-FL with malicious edge devices for handwriting recognition.

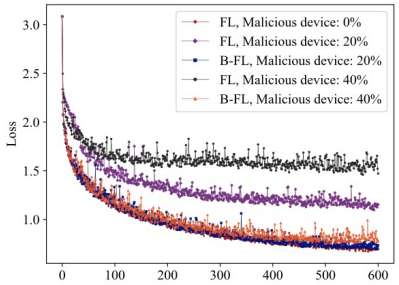


Fig. 8. Test loss of FL and B-FL with malicious edge devices for the CIFAR10 data set.

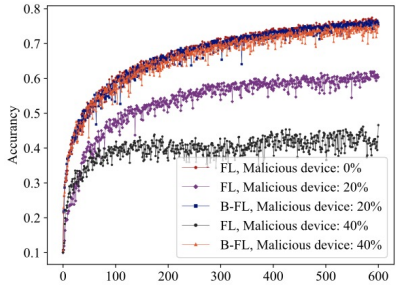


Fig. 9. Test accuracy of FL and B-FL with malicious edge devices for the CIFAR10 data set.

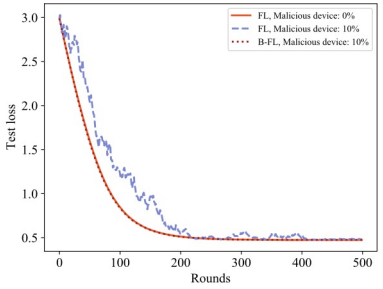


Fig. 10. Test loss of FL and B-FL with malicious devices for affect recognition.

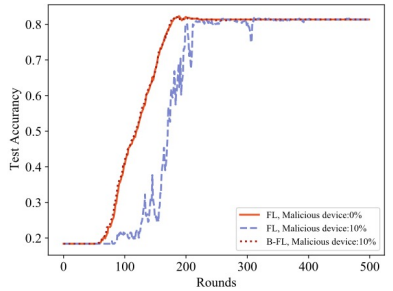


Fig. 11. Test accuracy of FL and B-FL with malicious edge devices for affect recognition.

C. Convergence of TD3-Based Resource Allocation Algorithm

C. Convergence of TD3-Based Resource Allocation Algorithm

To evaluate the effectiveness of our proposed TD3-based resource allocation algorithm, the convergence behavior of TD3 is shown in Fig. 12. The performance is presented by instant rewards following the training steps. We also illustrate the performance over different learning rate, i.e., $\eta_a = \eta_c = \{1e-4, 8e-5\}$. We choose the two learning rates because of the brief and clear performance improvement. It can be seen that the rewards will converge with the increase of time step t. The convergence rate will be promoted with larger learning rate.

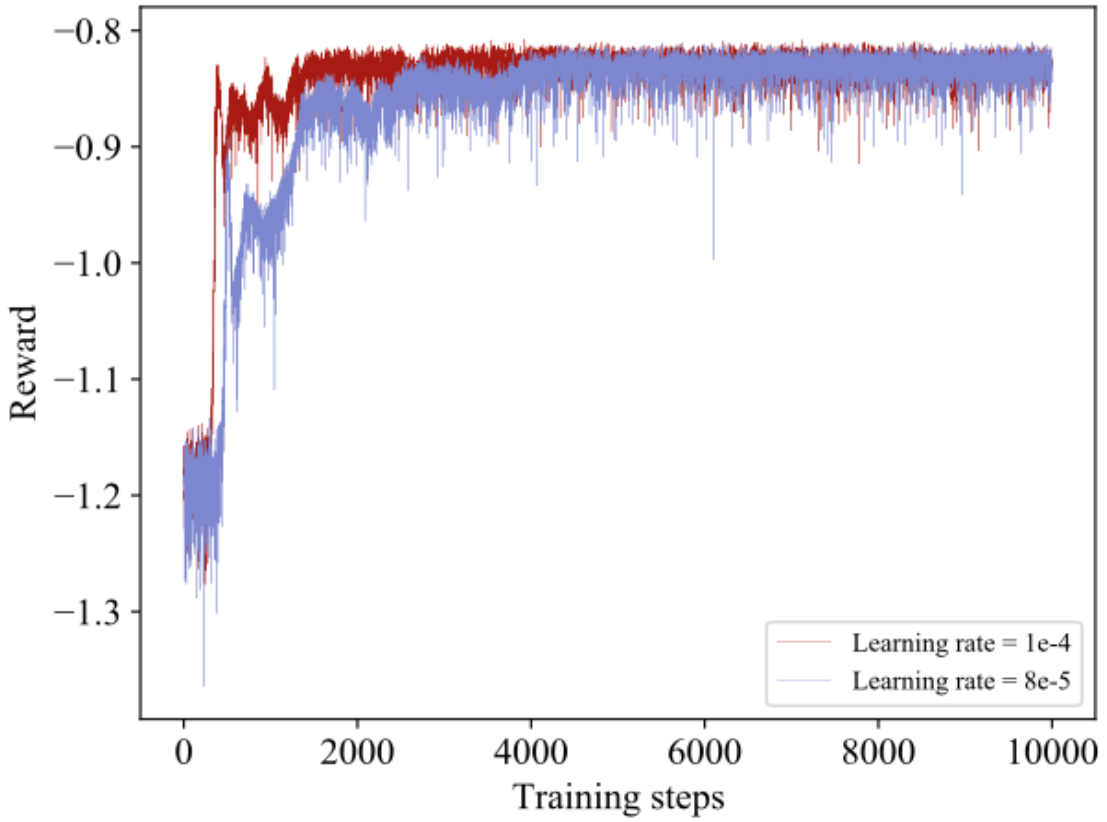


Fig. 12. Convergence performance of the TD3-based resource allocation algorithm.

D. Performance Impact of Different System Settings

D. Performance Impact of Different System Settings

To evaluate the performance of the proposed TD3-based resource allocation algorithm, we compare it with three benchmark algorithms, i.e., random allocation, average allocation, and Monte Carlo algorithm. Thus, we plot three figures to indicate the performance of our proposed algorithm and impact of different system settings, i.e., maximum system bandwidth, maximum system transmit power, and the number of edge devices. We average the results over 500 realizations of the system in our experiments to mitigate the randomness of devices' locations and channel states.

Fig. 13 illustrates the influence of maximum system bandwidth $b^{\max}$ to the long-term latency during B-FL training process. First, one observation is that our proposed TD3-based algorithm has the similar result with the benchmark Monte Carlo algorithm, and is better than the random allocation and average allocation schemes. In particular, the DRL-based algorithm is a little better than the Monte Carlo algorithm, because the Monte Carlo method is computationally expensive and can only obtain a suboptimal solution by using a finite number of samples. We take this result to present the performance of our TD3-based algorithm, which chooses more efficient and better resource allocation scheme. Second, the long-term latency can significantly decrease with the increasing maximum system bandwidth. Similarly, we present the impact of the maximum system transmit power in Fig. 14.

Fig. 15 plots the long-term latency to present the scalability to deal with different network scale of edge devices in the system. The number of edge devices varies from 10 to 60 and the other system settings are fixed. In the simulation, we find that TD3-based algorithm achieves greater gaps than three benchmarks with increasing number of edge devices. The long-term latency is increasing over more edge devices, which is because the average resources decrease and the size of new block increase.

For practical implementations, the B-FL system can be implemented by using the opensource blockchain frameworks, such as Ethereum, Hyperledger Fabric, Corda, and FISCO BCOS. Based on the blockchain framework, the local model can be submitted as a transaction and the global model aggregation and validation algorithms can be deployed as smart contracts. Particularly, Zhang *et al.* [46] and Kang *et al.* [47] have already deployed FL in the Ethereum framework and Corda framework, respectively.

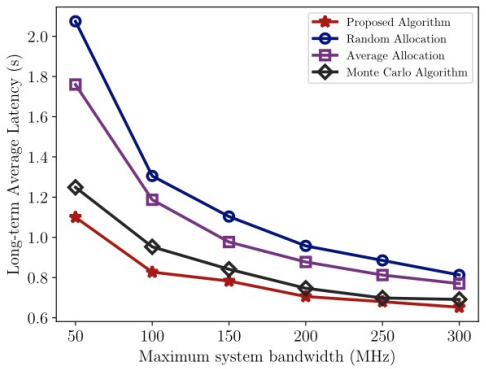


Fig. 13. Latency (s) versus maximum system bandwidth (MHz).

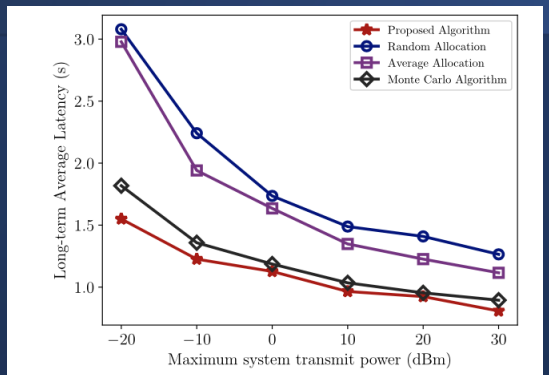


Fig. 14. Latency (s) versus maximum system transmit power (dBm).

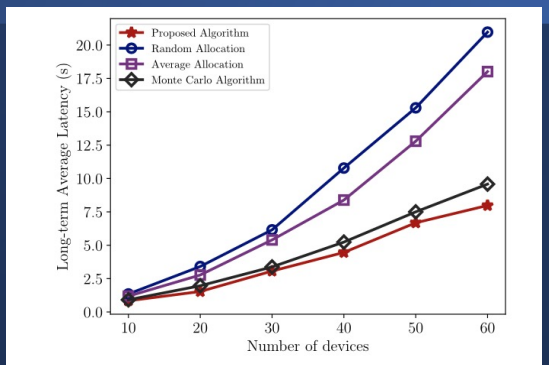


Fig. 15. Latency (s) versus number of edge devices.

VI. Conclusion

VI. Conclusion

In this article, we developed a B-FL architecture to ensure the security and privacy, which utilizes secure global aggregation and blockchain technique to resist the attacks from malicious edge devices and servers. We utilized PBFT consensus protocol in B-FL to achieve high effectiveness and low energy consumption for trustworthy FL. The procedures of PBFT-based wireless B-FL was presented first, and the training latency was then analyzed. We have formulated an optimization problem that jointly considers bandwidth allocation and transmit power allocation to minimize the long-term average training latency. To solve this network optimization problem, we derived a TD3-based algorithm to achieve long-term resource allocation and low computational complexity. Finally, we simulated the learning performance of wireless B-FL and the efficiency of the DRL-based resource allocation algorithm, which is compared with baseline algorithms (i.e., random allocation, average allocation, and Monte Carlo algorithm). Our simulation results shown that the wireless B-FL architecture can resist the attacks from malicious servers and malicious devices. Furthermore, the training latency of wireless B-FL can be significantly reduced by the developed TD3-based adaptive resource allocation scheme.

Thanks for listening

- Discussion and Comments