Adaptive Participant Selection in Heterogeneous Federated Learning

Rana Albelaihi*, Xiang Sun*, Warren D. Craft*, Liangkun Yu*, and Chonggang Wang†
*University of New Mexico, Albuquerque, NM 87131, USA.
†InterDigital Communications, Conshohocken, PA 19428, USA.

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685077

***Abstract*—Federated learning (FL) is a distributed machine learning technique to address the data privacy issue. Participant selection is critical to determine the latency of the training process in a heterogeneous FL architecture, where users with different hardware setups and wireless channel conditions communicate with their base station to participate in the FL training process. Many solutions have been designed to consider computational and uploading latency of different users to select suitable participants such that the straggler problem can be avoided. However, none of these solutions consider the waiting time of a participant, which refers to the latency of a participant waiting for the wireless channel to be available, and the waiting time could significantly affect the latency of the training process, especially when a huge number of participants are involved in the training process and share the wireless channel in the time-division duplexing manner to upload their local FL models. In this paper, we consider not only the computational and uploading latency but also the waiting time (which is estimated based on an M/G/1 queueing model) of a participant to select suitable participants. We formulate an optimization problem to maximize the number of selected participants, who can upload their local models before the deadline in a global iteration. The Latency awarE pARticipant selectioN (LEARN) algorithm is proposed to solve the problem and the performance of LEARN is validated via simulations.**

I. Introduction

The vast amount of data generated by various devices, along with the unprecedented evolution of machine learning (ML), are dramatically changing our lives [1]. Traditional ML frameworks, which require data uploaded from devices to a centralized facility for training a related ML model [2], lead to concerns about data privacy and huge network traffic loads. In order to address these concerns, it is crucial to develop a distributed ML framework, where local data sets in different devices do not have to be transmitted and can be analyzed locally to derive an ML model. In response, we have seen the rise of Federated Learning (FL) [3], which has been widely adopted by many applications to train their ML models, such as smart home [4] and eHealth [5].

In FL, an FL server in each global iteration selects subsets of users, which would train their local models based on their local data sets. The derived local models are uploaded to the FL server, which derives the global model based on, for example, the FedAvg algorithm [3]. The global iteration continues until the derived global model converges. Users in FL do not share their local data, thereby providing more privacy [6]. It has been demonstrated that the performance of FL is comparable to the centralized ML method when the data sets are independent and identically distributed (IID) among the users [7], [8]. However, the latency of FL for training an ML model could be much longer than that of the centralized ML method [9]. This is because FL typically trains a model over a heterogeneous network, where users[1] have different computing and communications capacities, thus resulting in the straggler problem that tremendously increases the latency [10]. Hence, designing an optimal participant selection method is fundamental to reduce the latency of the training process for FL. The goal of the participant selection is to maximize the number of **qualified participants** in each global iteration, where a **qualified participant** refers to a user that has been selected and can upload its derived local models before a predefined deadline. Note that it has been demonstrated that having more qualified participants can speed up the convergence rate [11], thus accelerating the whole FL training process.

Consider the scenario in which a number of users are distributed in a base station (BS)'s coverage area and try to participate in the FL process. Typically, the latency of a selected participant in its training process includes 1) **computational latency**: the latency of deriving its local model, 2) **waiting time**: the latency of waiting for the wireless channel to be available (if time division duplexing (TDD) is applied), and 3) **uploading latency**: the latency of uploading the local model to the FL server via the BS[2]. Many participant selection methods have been designed [12]–[14]; however, none of them consider the waiting time, thus selecting inappropriate participants which are unable to finish the training process before the deadline. Other works apply frequency division duplexing (FDD) to allocate and reserve bandwidth resources for the selected participants [15]–[17], i.e., the total amount of bandwidth allocated to the selected participants should be no larger than the available bandwidth of the BS, which is not necessary (since participants may request uploading of their local models in different time slots and do not compete

This work is supported by the National Science Foundation under Award OIA-1757207.

[1]Users, devices, and participants are interchangeable in the paper.

[2]In the paper, we do not consider the latency of the BS in broadcasting the global model to the participants and the latency of the FL server in deriving the global model by aggregating the received local models since both latencies are the same for different users and do not change by selecting different users.

978-1-7281-8104-2/21/$31.00 ©2021 IEEE

for the bandwidth resources) and could tremendously reduce the number of selected participants. In this paper, we design a novel participant selection method in TDD-based FL to maximize the number of qualified participants by jointly considering the computational and uploading latency as well as the waiting time of a participant. The contributions of the paper are summarized as follows.

1) We propose to consider the waiting time in calculating the overall latency of a participant for its training process and apply the queuing model to estimate the waiting time.
2) We formulate the participant selection problem to maximize the number of qualified participants. We design a latency-aware participant selection (LEARN) algorithm to efficiently solve the problem.
3) The performance of LEARN is demonstrated via extensive simulations.

The rest of the paper is organized as follows. Section II briefly reviews related work. Section III introduces the system models and the participant selection problem. Section IV provides the solution to the problem. Section V presents the simulation results, and Section VI concludes the paper.

II. Related Work

The original FL framework randomly selects users to participate in the distributed training process [3], which has been demonstrated to incur a long delay in training an ML model in heterogeneous settings, where different users have different computing and uploading capabilities [12]. Many researchers have sought to improve the performance of participant selection in FDD-based FL. For instance, Shi *et al.* [16] jointly optimized bandwidth allocation and participant selection to maximize the convergence rate, which is equivalent to minimizing the overall latency of training a global model. Xu *et al.* [17] also jointly optimized bandwidth allocation and participant selection to maximize the weighted sum of selected participants. However, they proposed selecting fewer participants in early FL rounds and more participants in later FL rounds. The results indicate that the designed participant selection method can improve the model accuracy, and reduce the overall energy consumption of the participants. FDD-based participant selection aims to allocate and reserve bandwidth for the selected participants, which is not necessary and could reduce the number of the selected participants.

Other works have explored the participant selection solutions in TDD-based FL. Nishio and Yonetani [12] designed a new participant selection approach to jointly consider both computational and uploading latency of participants to maximize the number of selected participants in each global iteration. Similarly, Yang *et al.* [18] proposed a joint participant selection and beamforming approach to maximize the number of selected participants in each global iteration while satisfying the mean-squared-error requirement. Amiri *et al.* [19] designed a participant selection and resource allocation method to select participants and assign the wireless channel based on their channel conditions and the significance of their local model updates. Zhang *et al.* [13] proposed a participant selection method to handle the non-IID data set among participants, where participants having a lower degree of non-IID data sets would be more frequently selected to improve the accuracy of the derived global model. The works mentioned above, however, do not consider the waiting time for the selected participants, and generally assumed that participants can immediately upload their local models once the models have been derived. In this paper, we will consider not only the computational and uploading latency but also the waiting time of participants in selecting suitable participants.

III. System Models and Problem Formulation

Let $\mathcal{I}$ be the set of users in the BS's coverage area. Let x_i be the binary variable to indicate whether user i is selected to train an FL global model (i.e., $x_i = 1$) or not (i.e., $x_i = 0$).

A. Federated learning background

The goal of the FL is to derive the vector of parameters, denoted as $\boldsymbol{\omega}$, for a global model in order to minimize the global loss function $\boldsymbol{\mathcal{F}}(\boldsymbol{\omega})$, i.e.,

$$\arg\min_{\boldsymbol{\omega}} \boldsymbol{\mathcal{F}}(\boldsymbol{\omega}) = \arg\min_{\boldsymbol{\omega}} \sum_{i\in\mathcal{I}} \frac{|\boldsymbol{\mathcal{D}}_i|}{|\boldsymbol{\mathcal{D}}|} f_i(\boldsymbol{\omega}) x_i, \tag{1}$$

where $|\boldsymbol{\mathcal{D}}|$ is the size of the overall training data set, $|\boldsymbol{\mathcal{D}}_i|$ is the size of the training data set at user i (where $\boldsymbol{\mathcal{D}} = \bigcup_{i\in\mathcal{I}} (\boldsymbol{\mathcal{D}}_i \cdot x_i)$), and $f_i(\boldsymbol{\omega})$ is the local loss function of user i over $\boldsymbol{\mathcal{D}}_i$, i.e.,

$$f_i(\boldsymbol{\omega}) = \frac{1}{|\boldsymbol{\mathcal{D}}_i|} \sum_{n\in\boldsymbol{\mathcal{D}}_i} f(\boldsymbol{\omega}, \boldsymbol{a}_{i,n}, b_{i,n}). \tag{2}$$

Here, $(\boldsymbol{a}_{i,n}, b_{i,n})$ is the input-output pair for the nth data sample in user i's data set, and $f(\boldsymbol{\omega}, \boldsymbol{a}_{i,n}, b_{i,n})$ captures the error of the local model (with parameter $\boldsymbol{\omega}$) over $(\boldsymbol{a}_{i,n}, b_{i,n})$.

FL is used to solve Problem (1) in a distributed manner. In each global iteration k, there are four steps, i.e.,

1) The BS broadcasts the current global model, denoted as $\boldsymbol{\omega}^{(k)}$, to all the selected participants.
2) Each selected participant i (where $x_i = 1$) performs local computation on the received model to train its local model over local data set $\boldsymbol{\mathcal{D}}_i$ based on the gradient descent method, i.e., $\boldsymbol{\omega}_i^{(k+1)} = \boldsymbol{\omega}_i^{(k)} - \delta \nabla f_i\left(\boldsymbol{\omega}_i^{(k)}\right)$, where δ indicates the step size or learning rate.
3) After obtaining the local model $\boldsymbol{\omega}_i^{(k)}$, participant i uploads its local model to the BS.
4) The BS aggregates the local models from the selected participants to update the global model based on, for example, FedAvg [3].

The FL keeps updating the global model in each iteration until the global model does not change anymore.

B. Computational latency

The computational latency for user i to train the model on its local data samples can be estimated by [20]

$$t_i^{comp} = \frac{C_i |\boldsymbol{\mathcal{D}}_i| v \log_2(1/\eta)}{f_i}, \tag{3}$$

where f_i is the computational capacity of user i in Hz, C_i is the average number of CPU cycles required for computing

one data sample item, $|\boldsymbol{\mathcal{D}}_i|$ is the number of data samples at user i (*i.e.*, the size of data set $\boldsymbol{\mathcal{D}}_i$), and $v\log_2(1/\eta)$ gives the number of required iterations to achieve the desired accuracy η. Here, $v = \frac{2}{(2-L\delta)\delta\gamma}$, where δ is the step size, and L and γ are calculated based on the eigenvalues of the Hessian matrix for the loss function.

C. Uploading latency

Assume that the BS applies time-division duplexing (TDD) to receive the local model parameters from different users, and so the achievable data rate of user i is [21]

$$r_i = B\log_2\left(1 + \frac{p_i h_i^2}{N_0}\right), \tag{4}$$

where B is the total available bandwidth for the BS, p_i and h_i are the maximum transmission power and the channel response of user i, respectively, and N_0 is the background noise. Thus, the latency of user i in uploading its local model to the BS is

$$t_i^{upload} = \frac{s}{r_i}, \tag{5}$$

where s is the size of the local model.

D. Time consumption of a global iteration

Let $\boldsymbol{\mathcal{J}}$ be the set of selected participants, i.e., $\boldsymbol{\mathcal{J}} = \{\forall i \in \boldsymbol{\mathcal{I}} \,|x_i = 1\}$, and we calculate the indices of the participant in $\boldsymbol{\mathcal{J}}$ based on the increasing order of their computational latency, i.e., if $j_1 > j_2$, then $t_{j_1}^{comp} \leq t_{j_2}^{comp}$, where $j_1,\ j_2 \in \boldsymbol{\mathcal{J}}$. As mentioned before, the latency of a participant for training its local model in a global iteration, denoted as t_j, consists of three parts, *i.e.*, computational latency t_j^{comp}, waiting time t_j^{wait}, and uploading latency t_j^{upload}. For example, as shown in Fig. 1, the 3rd participant spends t_3^{comp} amount of time to obtain its local model. Before the 3rd participant uploads its local model, it has to wait for t_3^{wait} amount of time until the wireless channel is available (i.e., all the previous participants, who finished their local model updates before the 3rd participant, have completed their local model uploading). Once the wireless channel is available, the 3rd participant spends t_3^{upload} amount of time to upload its local model to the BS. Note that not all the participants has a waiting time.

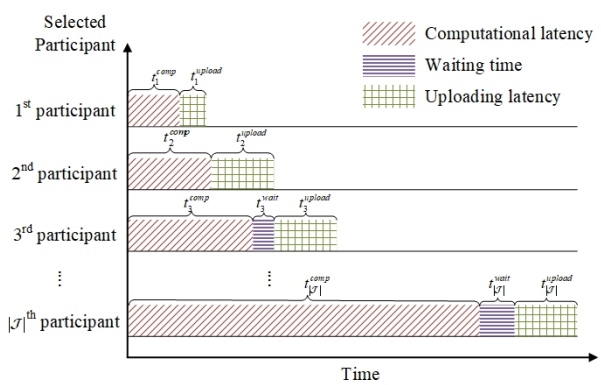


Fig. 1: Illustration of scheduling for the selected participants.

From Fig. 1, we can find that the total time consumption t of a global iteration is equal to the total time consumption of the last participant in $\boldsymbol{\mathcal{J}}$, who has the longest computational latency among all the participants. That is,

$$t = t_{|\boldsymbol{\mathcal{J}}|}^{comp} + t_{|\boldsymbol{\mathcal{J}}|}^{wait} + t_{|\boldsymbol{\mathcal{J}}|}^{upload}, \tag{6}$$

where $|\boldsymbol{\mathcal{J}}|$ is the index of the last participant in $\boldsymbol{\mathcal{J}}$, i.e., $|\boldsymbol{\mathcal{J}}| = \arg\max_{i\in\boldsymbol{\mathcal{I}}} \{t_i^{comp} x_i\}$. Here, $t_{|\boldsymbol{\mathcal{J}}|}^{comp}$ and $t_{|\boldsymbol{\mathcal{J}}|}^{upload}$ can be calculated based on Eq. (3) and Eq. (5), respectively. In order to estimate $t_{|\boldsymbol{\mathcal{J}}|}^{wait}$, we consider the process of different participants uploading their local models to the BS via the shared frequency band as an M/G/1 queueing model. Specifically, an arrival at the queue indicates a participant just completed its local model calculation and is requesting the wireless channel to upload its local model. A departure from the queue indicates the wireless channel is assigned to a participant for its local model uploading. In addition, the arrival rate (i.e., the rate of participants completing their local model calculations) is assumed to follow a Poisson distribution during the time period $\left[t_1^{comp}, t_{|\boldsymbol{\mathcal{J}}|}^{comp}\right]$, where the average arrival rate is

$$\lambda = \frac{\sum_{i\in\boldsymbol{\mathcal{I}}} x_i}{t_{|\boldsymbol{\mathcal{J}}|}^{comp} - t_1^{comp}}. \tag{7}$$

Moreover, the departure rate (*i.e.*, the rate at which participants obtain the wireless channel for their local model uploading) is assumed to follow a general distribution during the time period $\left[t_1^{comp}, t_{|\boldsymbol{\mathcal{J}}|}^{comp}\right]$, where the average service rate is

$$\mu = \frac{\sum_{i\in\boldsymbol{\mathcal{I}}} \frac{s}{r_i} x_i}{\sum_{i\in\boldsymbol{\mathcal{I}}} x_i}. \tag{8}$$

Therefore, the average waiting time of an M/G/1 queue, which is used to estimate $t_{|\boldsymbol{\mathcal{J}}|}^{wait}$, is

$$t_{|\boldsymbol{\mathcal{J}}|}^{wait} = \frac{\lambda E\left(\mu^2\right)}{2\left(1-\lambda/\mu\right)} = \frac{\sum_{i\in\boldsymbol{\mathcal{I}}} \left(\frac{s}{r_i}\right)^2 x_i}{2\left(t_{|\boldsymbol{\mathcal{J}}|}^{comp} - t_1^{comp} - \frac{\left(\sum_{i\in\boldsymbol{\mathcal{I}}} x_i\right)^2}{\sum_{i\in\boldsymbol{\mathcal{I}}} \frac{s}{r_i} x_i}\right)}, \tag{9}$$

where $E\left(\mu^2\right) = \frac{\sum_{i\in\boldsymbol{\mathcal{I}}} \left(\frac{s}{r_i}\right)^2 x_i}{\sum_{i\in\boldsymbol{\mathcal{I}}} x_i}$ is the expected value of μ^2. Based on Eq. (9), we can derive a tradeoff between minimizing the computational latency $t_{|\boldsymbol{\mathcal{J}}|}^{comp}$ and minimizing the waiting time $t_{|\boldsymbol{\mathcal{J}}|}^{wait}$. That is, if we want to reduce $t_{|\boldsymbol{\mathcal{J}}|}^{wait}$, we have to pick a participant with larger $t_{|\boldsymbol{\mathcal{J}}|}^{comp}$, and vice versa.

E. Problem formulation

We formulate the participant selection problem as follows.

$$\boldsymbol{P0}: \max \sum_{i\in\boldsymbol{\mathcal{I}}} x_i, \tag{10}$$

$$\text{s.t.} \quad t_{|\boldsymbol{\mathcal{J}}|}^{comp} + t_{|\boldsymbol{\mathcal{J}}|}^{wait} + t_{|\boldsymbol{\mathcal{J}}|}^{upload} \leq \tau, \tag{11}$$

$$|\boldsymbol{\mathcal{J}}| = \arg\max_{i\in\boldsymbol{\mathcal{I}}} \{t_i^{comp} x_i\}, \tag{12}$$

$$\forall i \in \boldsymbol{\mathcal{I}}, x_i \in \{0,1\}. \tag{13}$$

The objective is to maximize the number of selected participants in a global iteration. The first constraint implies the

total time consumption of a global iteration (which is the time consumption of the last participant in $\mathcal{J}$) should be less than the predefined deadline, denoted as τ. That is, all the selected participants should be qualified participants. The second constraint defines the last participant in $\mathcal{J}$. The third constraint implies that x_i is a binary variable.

IV. Latency aware participant selection

We design a heuristic algorithm, Latency awarE pARticipant selectioN (LEARN), to efficiently solve $\boldsymbol{P0}$.

Lemma 1. Given the last participant $|\mathcal{J}|$ in $\mathcal{J}$, the optimal user selection is to iteratively select a user i^* as a participant (i.e., $x_{i^*} = 1$), where user i^* is the one that has the maximum uploading data rate among all the users whose computation latency is no larger than the last participant $t^{comp}_{|\mathcal{J}|}$, i.e., $i^* = \arg\max_i \left\{ r_i \,\middle|\, i \in \mathcal{I}' \right\}$, where $\mathcal{I}' = \left\{ i \in \mathcal{I} \,\middle|\, t^{comp}_i \le t^{comp}_{|\mathcal{J}|} \ \&\ x_i = 0 \right\}$. The participant selection continues until all the users in $\mathcal{I}'$ have been selected as participants (i.e., $\mathcal{I}' = \emptyset$) or Constraint (14) is no longer met.

Proof: Plugging Eq. (9) into Constraint (11), we have

$$\frac{\sum_{i\in\mathcal{I}} \left(\frac{s}{r_i}\right)^2 x_i}{2\left(t^{comp}_{|\mathcal{J}|} - t^{comp}_1 - \frac{\left(\sum_{i\in\mathcal{I}} x_i\right)^2}{\sum_{i\in\mathcal{I}} \frac{s}{r_i} x_i}\right)} \le \tau - t^{comp}_{|\mathcal{J}|} - t^{upload}_{|\mathcal{J}|}$$

$$\Rightarrow \sum_{i\in\mathcal{I}} x_i \le \sqrt{\sum_{i\in\mathcal{I}} \frac{s\varphi_{|\mathcal{J}|}}{r_i} x_i}, \tag{14}$$

where

$$\varphi_{|\mathcal{J}|} = t^{comp}_{|\mathcal{J}|} - t^{comp}_1 - \frac{\sum_{i\in\mathcal{I}} \left(\frac{s}{r_i}\right)^2 x_i}{2\left(\tau - t^{comp}_{|\mathcal{J}|} - t^{upload}_{|\mathcal{J}|}\right)}. \tag{15}$$

Here, $\tau - t^{comp}_{|\mathcal{J}|} - t^{upload}_{|\mathcal{J}|} > 0$. Based on Eq. (14), maximizing $\sum_{i\in\mathcal{I}} x_i$ is equal to maximizing $\sum_{i\in\mathcal{I}} \frac{s\varphi}{r_i} x_i$. Thus, $\boldsymbol{P0}$ can be converted into

$$\boldsymbol{P1}: \max \sum_{i\in\mathcal{I}} \frac{s\varphi_{|\mathcal{J}|}}{r_i} x_i, \tag{16}$$

$$\text{s.t.} \quad \textit{Constraints } (12), (13), \textit{and } (14). \tag{17}$$

In order to guarantee Constraint (12), all the participants should be selected from the set $\mathcal{I}'$, where $\mathcal{I}'$ includes all the users whose computation latency is no larger than the last participant $t^{comp}_{|\mathcal{J}|}$, i.e., $\mathcal{I}' = \left\{ i \in \mathcal{I} \,\middle|\, t^{comp}_i \le t^{comp}_{|\mathcal{J}|} \ \&\ x_i = 0 \right\}$. Thus, $\boldsymbol{P1}$ can be converted into

$$\boldsymbol{P2}: \max \sum_{i\in\mathcal{I}'} \frac{s\varphi_{|\mathcal{J}|}}{r_i} x_i, \tag{18}$$

$$\text{s.t.} \quad \textit{Constraints } (13) \textit{ and } (14). \tag{19}$$

The optimal solution of $\boldsymbol{P2}$ is easy to derive, i.e., to iteratively select the user (denoted as i^*), which incurs the minimum value of $\frac{s\varphi_{|\mathcal{J}|}}{r_i}$ among all the users in $\mathcal{I}'$, as the participant (i.e., $x_{i^*} = 1$). The iteration terminates when the selected user i^* cannot meet Constraint (14). Note that picking the user with the minimum value of $\frac{s\varphi_{|\mathcal{J}|}}{r_i}$ is equivalent to picking the user with the maximum value of r_i since the value of $s\varphi_{|\mathcal{J}|}$ is the same for all the users. Thus, we have $i^* = \arg\max_i \left\{ r_i \,\middle|\, i \in \mathcal{I}' \right\}$. ■

Give the last participant $|\mathcal{J}|$, Lemma 1 provides the optimal participant selection. However, selecting a different last participant $|\mathcal{J}|$ may have a different value of $\varphi_{|\mathcal{J}|}$, thus leading to a different user selection. Intuitively, picking a last participant that incurs a smaller value of φ_i would select more users as the participants while satisfying Constraint (14). Therefore, the optimal last participant is the user that incurs the smallest φ_i among all the users that satisfy $\tau - t^{comp}_i - t^{upload}_i > 0$, i.e.,

$$|\mathcal{J}| = \arg\min_{i\in\mathcal{I}} \left\{ \varphi_i \,\middle|\, \tau - t^{comp}_i - t^{upload}_i > 0 \right\}, \tag{20}$$

where φ_i is calculated based on Eq. (15). It is difficult to derive the optimal last participant $|\mathcal{J}|$ since the value of $\sum_{i\in\mathcal{I}} \left(\frac{s}{r_i}\right)^2 x_i$ in Eq. (15) is unknown until all the participants are selected. Thus, we design the LEARN algorithm to iteratively choose a better last participant. Specifically,

1) The last participant $|\mathcal{J}|$ is initialized by picking the user that incurs the largest value of $t^{comp}_i + t^{upload}_i$ among all the users that satisfy $\tau - t^{comp}_i - t^{upload}_i > 0$, i.e., $|\mathcal{J}| = \arg\max_{i\in\mathcal{I}} \left\{ t^{comp}_i + t^{upload}_i \,\middle|\, \tau - t^{comp}_i - t^{upload}_i > 0 \right\}$.
2) In each iteration, given the last participant $|\mathcal{J}|$, a new group of participants is selected based on Lemma 1 (i.e., Steps 8–13 in Algorithm 1). Based on the selected participants, the last participant $|\mathcal{J}|$ is updated according to Eq. (20). The updated last participant will be used to determine the group of participants in the next iteration.
3) The iteration continues until the number of participants (i.e., $\sum_{i\in\mathcal{I}} x_i$) does not increase.

The LEARN algorithm is summarized in Algorithm 1.

V. Simulation

Assume that there are 200 users in the BS's coverage area, and the path loss between a user and the BS is estimated by $128.1 + 37.6 \log_{10} d$, where d is the distance between a user and the BS in kilometers. The available bandwidth at the BS is $B = 5$ MHz and TDD is applied at the BS to uploading data from the users. In addition, we set $\gamma = 2$, $L = 4$, $\delta = 0.1$, and $\eta = 0.1$ to calculate the computational latency of a participant based on Eq. (3). Other parameters are listed in Table I.

We compared LEARN with two other participant selection algorithms, i.e., Computational Aware paRticipant selectioN (CARN) and Frequency division duplex-based deadline Aware paRticipant selectioN (FARN). The basic idea of CARN is to iteratively select a user as a participant based on the increasing order of the computational latency of the users. CARN continues to select participants until the selected participant in the current iteration cannot satisfy $\left(t^{comp}_i + t^{upload}_i\right) x_i \le \tau$. On the other hand, FARN [15]–[17] applies FDD to upload

Algorithm 1: LEARN algorithm

Initialize $x_i = 0$, $\forall i \in \boldsymbol{\mathcal{I}}$ and $N = 0$.
Initialize the last participant $|\boldsymbol{\mathcal{J}}| = \arg\max_{i \in \boldsymbol{\mathcal{I}}} \left\{ t_i^{comp} + t_i^{upload} \middle| \tau - t_i^{comp} - t_i^{upload} > 0 \right\}$.
while $N < \sum_{i \in \boldsymbol{\mathcal{I}}} x_i$ **do**
  Set $x_i = 0$, $\forall i \in \boldsymbol{\mathcal{I}}$;
  $x_{|\boldsymbol{\mathcal{J}}|} = 1$;
  Obtain $\boldsymbol{\mathcal{I}}' = \left\{ i \in \boldsymbol{\mathcal{I}} \middle| t_i^{comp} \leq t_{|\boldsymbol{\mathcal{J}}|}^{comp} \ \& \ x_i = 0 \right\}$;
  while $\boldsymbol{\mathcal{I}}' \neq \emptyset$ & *Constraint* (14) *is met* **do**
    Calculate $i^* = \arg\max_i \left\{ r_i \middle| i \in \boldsymbol{\mathcal{I}}' \right\}$;
    $x_{i^*} = 1$; $\boldsymbol{\mathcal{I}}' = \boldsymbol{\mathcal{I}}' \backslash i^*$;
  end
  $N = \sum_{i \in \boldsymbol{\mathcal{I}}} x_i$.
  Update the last participant $|\boldsymbol{\mathcal{J}}|$ based on Eq. (20);
end

participants' local models to the BS, and so the total amount of bandwidth allocated to the participants should be no larger than B, i.e., $\sum_{i \in \boldsymbol{\mathcal{I}}} b_i x_i \leq B$ (where b_i is the amount of bandwidth allocated to user i). Meanwhile, each participant should satisfy $\left(t_i^{comp} + t_i^{upload} \right) x_i \leq \tau$.

TABLE I: Simulation Parameters

Parameter	Value
Size of data samples ($\lvert\boldsymbol{\mathcal{D}}_i\rvert$)	500 samples
Number of CPU cycles (C_i)	$U(5, 15) \times 10^4$ cycles/sample
Computation capacity (f_i)	$U(1.5, 2)$ GHz
Transmission power (p_i)	10 dBm/MHz
Noise (N_0)	-104 dBm/10MHz
Size of local model (s)	100 Kbits
Predefined deadline (τ)	1 second

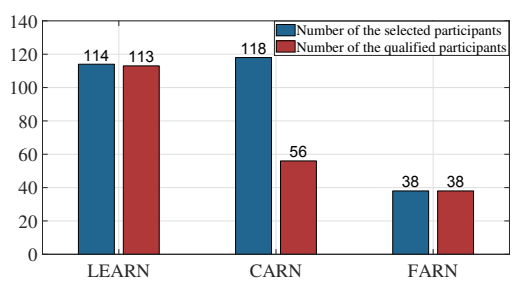


Fig. 2: Number of the selected and qualified participants.

Fig. 2 shows the results for the three algorithms in a global iteration, where blue and red bars indicate the number of selected participants and the number of qualified participants, respectively. As mentioned in the Introduction section, a qualified participant refers to a selected participant which can upload its local model before the deadline τ. From the figure, we can derive that although CARN selects the largest number of participants, most of the selected participants cannot upload their local models to the BS before the deadline (i.e., the number of qualified participants is less than half of the number of selected participants). This is because CARN underestimates the overall latency of a participant as it does not consider the waiting time. As a result, as shown in Fig. 3(*b*), most of the participants suffer from long waiting times, and are thus unable to upload their models before the deadline. On the other hand, FARN selects the fewest participants, but all the selected participants are qualified participants. This is because FARN applies FDD to reserve enough bandwidth to the selected participants such that these participants can upload their local models before the deadline. Thus, as shown in Fig. 3(*c*), all the selected participants in FARN have no waiting time as they can immediately upload their local models once the models are derived. Yet, the number of selected participants is limited by the available bandwidth B. LEARN selects fewer participants than CARN, but nearly all of the them can upload their local models before the deadline, and so LEARN has the most qualified participants than the other algorithms. This is because LEARN can accurately estimate the overall latency of a participant by considering the waiting time. As a result, as shown in Fig. 3(*a*), the waiting time for the selected participants are relatively low as compared to CARN.

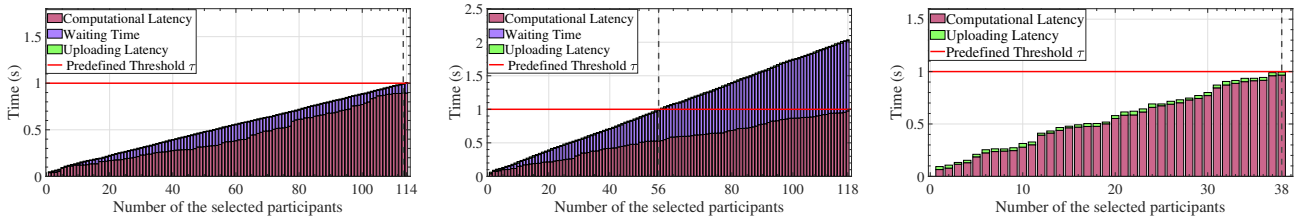
Fig. 4 shows the number of selected and qualified participants for different algorithms by varying the data set size $|\boldsymbol{\mathcal{D}}_i|$. Note that varying $|\boldsymbol{\mathcal{D}}_i|$ changes the computation latency of all the participants. From Fig. 4(*a*) and (*b*), we can see that LEARN selects slightly fewer participants than CARN, but has the most qualified participants. However, as $|\boldsymbol{\mathcal{D}}_i|$ increases, the performance gap (i.e., the difference of the number of qualified participants between LEARN and CARN/FARN) reduces. This is because the computational latency becomes the key factor to determine the overall latency of a participant as $|\boldsymbol{\mathcal{D}}_i|$ increases, and so the performance of LEARN, whose advantage is to consider the waiting time during the participant selection, could be similar with CARN/FARN if the computational latency is much larger than the waiting time and uploading latency of a participant.

Fig. 5 shows the number of selected and qualified participants for different algorithms by varying the amount of bandwidth B. Note that varying B changes the uploading latency of all the participants. As shown in Fig. 5(*b*), LEARN outperforms CARN and FARN, but the performance gap reduces as B increases. This is because the uploading latency and the waiting time of a participant reduce as B increases, and so the computational latency becomes the key factor to determine the overall latency of a participant, which, as we illustrated previously, would reduce the performance gap.

Fig. 6 shows the performance of the three algorithms under different deadlines τ. As the deadline relaxes, we have more selected and qualified participants for inclusion, regardless of the algorithms. However, LEARN always achieves the largest number of qualified participants as shown in Fig. 6(*b*).

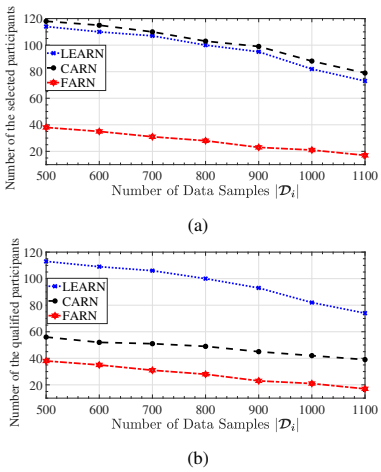
VI. Conclusion

In this paper, we have proposed that the waiting time of a participant should be considered as it may significantly affect the overall latency of a global iteration. We have proposed to use an M/G/1 queueing model to estimate the average waiting



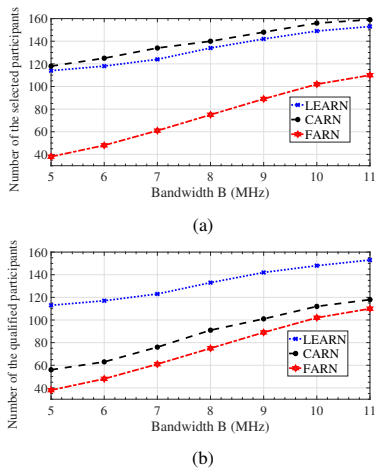
(a) LEARN (b) CARN (c) FARN

Fig. 3: The latency of the selected participants in a global iteration.



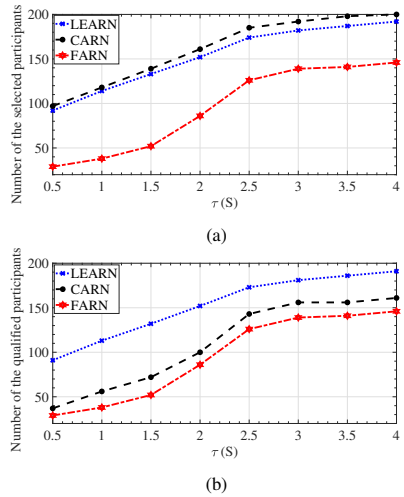
(a) (b)

Fig. 4: The number of selected and qualified users over $|\mathcal{D}_i|$.



(a) (b)

Fig. 5: The number of selected and qualified users over B.



(a) (b)

Fig. 6: The number of selected and qualified users over τ.

time of a participant, and then formulated a new participant selection problem and designed the LEARN algorithm to efficiently solve the problem.

References

[1] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, 2016.

[2] Y. Liu, S. Bi, Z. Shi, and L. Hanzo, "When machine learning meets big data: A wireless communication perspective," *IEEE . Veh. Technol. Mag.*, vol. 15, no. 1, pp. 63–72, 2020.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th Intl. Conf. on Artificial Intelligence and Statistics, AISTATS*, vol. 54, 2017, pp. 1273–1282.

[4] T. Yu, T. Li, Y. Sun, S. Nanda, V. Smith, V. Sekar, and S. Seshan, "Learning context-aware policies from multiple smart homes via federated multi-task learning," in *2020 IEEE/ACM Fifth Intl. Conf. on Internet-of-Things Design and Implementation*, 2020, pp. 104–115.

[5] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Intl. J. Medical Inform.*, vol. 112, pp. 59–67, 2018.

[6] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated Multi-Task Learning," *arXiv e-prints*, p. arXiv:1705.10467, May 2017.

[7] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv e-prints*, p. arXiv:1806.00582, Jun. 2018.

[8] M. Safa Ozdayi, M. Kantarcioglu, and R. Iyer, "Improving Accuracy of Federated Learning in Non-IID Settings," *arXiv e-prints*, p. arXiv:2010.15582, Oct. 2020.

[9] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7108–7123, 2020.

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

[11] S. U. Stich, "Local SGD Converges Fast and Communicates Little," *arXiv e-prints*, p. arXiv:1805.09767, May 2018.

[12] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019 - 2019 IEEE Intl. Conf. on Commun. (ICC)*, 2019, pp. 1–7.

[13] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, 2021.

[14] S. Zhai, X. Jin, L. Wei, H. Luo, and M. Cao, "Dynamic federated learning for gmec with time-varying wireless link," *IEEE Access*, vol. 9, pp. 10 400–10 412, 2021.

[15] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless . Commun.*, vol. 20, no. 1, pp. 269–283, 2021.

[16] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *2020 IEEE Intl. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[17] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1188–1200, 2021.

[18] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, 2020.

[19] M. M. Amiria, D. Gündüzb, S. R. Kulkarni, and H. V. Poor, "Convergence of update aware device scheduling for federated learning at the wireless edge," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2021.

[20] Z. Yang, M. Chen, W. Saad, C. S. Hong, M. Shikh-Bahaei, H. V. Poor, and S. Cui, "Delay Minimization for Federated Learning Over Wireless Communication Networks," *arXiv e-prints*, p. arXiv:2007.03462, Jul. 2020.

[21] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for internet of things," *IEEE Internet of Things Journal*, 2021, early access, doi:10.1109/JIOT.2021.3103715.