Democratizing the Federation in Federated Learning

Ningxin Su, *Student Member, IEEE,* Baochun Li, *Fellow, IEEE,* Bo Li, *Fellow, IEEE*

***Abstract*—Federated learning (FL) is a widely acknowledged distributed training paradigm that preserves the privacy of data on participating clients, and has become the *de facto* standard for distributed machine learning across a large number of edge devices. Conventional FL, however, has a rather rigid design, where the server is the *dominant player* that selects a subset of its *clients* to participate in each communication round, and clients are merely *followers*, and are not offered the freedom to accept or decline invitations from the server to participate. In addition, clients may become unavailable or very slow due to a wide variety of reasons, yet it may take an excessive amount of time for a conventional FL server to recognize that a particular client is unavailable.**

In this paper, we advocate for a more pragmatic paradigm in federated learning, called *democratic federated learning*, to offer more freedom to both servers and clients with respect to the ability to accept or decline requests, and to explicitly request to participate. In contrast to conventional federated learning, our paradigm allows (1) both the server and clients to participate and withdraw from the federated learning process at any time; (2) the server to decide whether to reject clients' updates based on the current model convergence steps, i.e., after satisfying the minimum required clients' updates received; and (3) the clients to adjust the local epochs based on their own training and communication time. Our experimental results on a variety of datasets and models have confirmed that *democratic federated learning* not only accelerates the convergence process but also improves the accuracy of converged models, and serves as a foundation for future explorations into client-centric models within the FL ecosystem.

I. Introduction

In recent years, federated learning (FL) [1] has gained prominence due to the advent of machine learning and the growing emphasis on privacy. Federated learning represents a paradigm of distributed machine learning, in which clients perform local training iteratively in rounds, and the server aggregates their model updates to obtain a shared global model to be distributed back to the clients in the next round. With federated learning, data is private to each client and not disclosed to the server. In essence, federated learning employs a client-server architecture, where the server is in charge of selecting the subset of clients for training the shared model in each round, and the clients are assumed to take a passive role and to start training on-demand when they are selected. Such a strong assumption prevents clients from declining or disconnecting when they are chosen in each training round, and disregards the heterogeneity and hardware performance of clients. Additionally, once the server has selected the clients in each training round, it must wait for all the clients to provide their training results before proceeding to the next round.

Unfortunately, such a stringent assumption in conventional FL is not typically valid in practice. *First*, clients may become disconnected at any time owing to network or energy issues, and the server cannot guarantee that it will receive the same number of updates from all the selected clients in each round. *Second*, the hardware heterogeneity across devices causes their local training speed to vary substantially, and synchronous designs with an equal number of local training epochs result in an excessive amount of waiting time. *Third*, data on various clients are diverse, and the statistical heterogeneity induced by non-i.i.d. data causes the phenomenon of "model drift," where the trained model diverges across different clients. This phenomenon will be worse if the server cannot select clients based on historical information.

With the fundamental assumption that clients take a passive role in conventional FL, clients do not enjoy any "rights" in the working relationships between the clients and the server, which are heavily skewed in favour of the server. Naturally, if clients are required to use their local data and computing resources forcibly without any reward, they are less likely to participate and engage in the FL training process. In practice, a large proportion of clients may not prefer to participate, due to a lack of local computation resources or available data. However, with appropriate incentive mechanisms in place [2], [3], some clients may also be willing to join the FL training session. Despite the fact that defense mechanisms have been designed against a variety of data reconstruction attacks, and attempts have been made to optimize FL-related workflows so as to expedite the convergence of FL models, these improvements have never taken the "rights" of clients into consideration.

In this paper, we propose the concept of *democratic federated learning*, which provides clients with the option to join an FL training session on their own initiative, or decline the FL server's invitation to engage in the training session. Having the freedom to join or to decline invitations from the server will maximize the assurance that the clients have full autonomy throughout the FL training session. If the server receives an acceptance response from the client, it will return the current global model, indicating that the client has joined training. Although Anarchic FL [4] also provided more "rights" to the clients, it only permits them to adjust their local steps, rather than having the complete freedom of declining to participate in the session.

In order to provide servers and clients with equal privileges,

Ningxin Su and Baochun Li are with the Department of Electrical and Computer Engineering, University of Toronto. Their email addresses are ningxin.su@mail.utoronto.ca and bli@ece.toronto.edu, respectively.

Bo Li is with the Department of Computer Science, Hong Kong University of Science and Technology. His email address is bli@cse.ust.hk.

The research was supported in part by a RGC RIF grant under the contract R6021-20, RGC TRS grant under the contract T43-513/23N-2, RGC CRF grants under the contracts C7004-22G, C1029-22G and C6015-23G, and RGC GRF grants under the contracts 16200221, 16207922 and 16207423.

and to allow clients to join the training session proactively, we wish to allow the server to have the liberty of declining such requests from a client as well. This is determined by a client selection algorithm on the server, which may improve the convergence speed and the converged accuracy of the model [5], [6]. With the same degrees of freedom from both perspectives of clients and the server, the training session is inherently *asynchronous*: the server should not wait for all of its selected clients to submit model updates before initiating its aggregation. Instead, it should start aggregating as soon as enough clients have reported their model updates.

Additionally, *democratic federated learning* employs variable training epochs on the client side to further reduce the time required for a client to complete one training round. It has been observed that in federated learning, there are clients with exceptionally long training times, which may be due to limited computational power or insufficient memory. In synchronous federated learning, the round completion time would be extended due to these slower clients, as the server must wait for all clients to return their model updates. However, this issue can also arise in asynchronous federated learning because it tolerates stale model updates, meaning the server will aggregate the model updates from several rounds prior. Thus, clients that require more extended training times may provide more outdated model updates, which might not significantly aid the convergence of the global model. Dynamically reducing the training epochs for these slower clients, while maintaining the epochs for faster clients, could intuitively accelerate the convergence process of federated learning.

In the context of related work, the original contributions in this paper are as follows.

- We propose a new paradigm, referred to as *democratic federated learning*, that fundamentally changes the rules that govern how FL clients and the server engage and interact with each other. Clients can actively ask to join the FL session and refuse the server's request to join; likewise, the server can invite or decline clients that actively join the FL session based on their performance. Thus, the clients and server are no longer required to accept arriving requests, and can opt to accept or decline based on their own current conditions.
- We propose a new algorithm that governs how both the clients and the server process incoming requests in the *democratic federated learning* paradigm. With our new algorithm, the clients or server will *asynchronously* proceed with training or aggregation, respectively, in a parallel fashion, with the objective of optimizing the amount of time it takes to converge and the accuracy after convergence.
- We implement and evaluate our proposed algorithm in the paradigm of *democratic federated learning* on PLATO, an open-source framework for FL research that excels at the task of comparing performance across-the-board. Our evaluations span a variety of datasets and models, and show that our algorithm outperforms state-of-the-art *asynchronous* and *anarchic* FL algorithms with respect to the wall-clock time needed for convergence, yet supporting more realistic assumptions that both the clients and the server are fully autonomous and driven by self-interests.

II. PRELIMINARIES AND RELATED WORK

To date, only a small fraction of work has discussed the relationship between clients and servers in federated learning. Anarchic FL [4] was the first work that inspired researchers to pay more attention to the right of clients to join FL freely. Although it proved that the active participation of clients in the asynchronous case helps convergence, it did not consider the right of the server to select and reject clients. That is, if an unsuspecting client wants to join an FL training session actively, the anarchic server has no way of rejecting it. In contrast, the work of Ruan *et al.* [7] has rigorously analyzed the cases of clients' refusal to join, dropouts and early departures. However, Ruan *et al.* maintains the performance of model convergence in synchronous FL by quickly notifying other clients to restart.

Flexible worker participation is an alternative approach for addressing client heterogeneity [8]. Flexible worker participation allows the server to select clients to participate in training based on the client's data distribution in each round. This is able to filter out those clients that may negatively affect the convergence of the shared global model, and allows the shared global model to converge more quickly and to a higher accuracy. Oort [5] shows that if we select clients with higher local losses more favourably with higher probabilities, the global model may converge faster during the training process. Pisces [6], in addition to data utility, also incorporates client staleness as a condition for client selection.

In order to further relax the strong assumptions on the server side, many techniques can nevertheless guarantee model convergence under the premise that the system is asynchronous [9]. FedAsync [10] is the first to propose the concept of asynchronous federated learning, but it aggregates each incoming client update with the global model too aggressively, and convergence is no longer guaranteed. Since then, numerous works have focused on reducing the wall-clock time of global model convergence. FedBuff [11] stores clients arriving at different times in a buffer and controls the length of the buffer to control the degree of asynchrony, which corresponds to the minimum number of clients accumulated at the server before asynchronous aggregation commences. Port [12] not only considers the minimum number of clients accumulated, but also assigns weights to clients with varied model similarities during the aggregation process. Although this step increases the aggregation bias, it decreases the global model bias from the local model aggregation of non-i.i.d. data.

III. DEMOCRATIC FEDERATED LEARNING

In democratic federated learning (FL), we wish to provide the server and clients with equal privileges. In particular, we allow both clients and servers to accept and decline

membership in each communication round within the FL training session. In this section, we integrate client selection into the asynchronous FL training paradigm, and design a highly flexible, practical democratic FL algorithm that offers much more freedom to both the server and clients.

Client-Server Interaction. How could democratic FL be implemented so that servers and clients can freely participate and exit the training session at any time? Most of the existing research on federated learning, beginning with McMahan *et al.* [1], assumed that each communication round between the client and the server is fully synchronous. With conventional synchronous FL, the server selects a subset of clients, assigns them a global model and a fixed number of local training epochs, and the clients are required to proceed with local training.

In our proposed algorithm, the server has three possible ways of interacting with a client: sending an invitation proactively, accepting the client's join request, or rejecting its request. The first two actions cause the client to automatically join a server-maintained list of participating clients. On the other hand, clients also have three alternatives for providing feedback: accepting an invitation from the server, applying to join the FL session proactively, or declining to join the FL session. When a client responds with a refusal to join, the server is unable to add the client to its list of participants. With this algorithm, before the server adds a client to the list of participating clients, both the server and the clients must agree that the other party may join the FL session. Similar to the celebrated *stable matching* algorithm, the essence of our algorithm is that the server and its clients would asynchronously match each other's requests, and clients must be incentivized to join the FL session.

If a client's join request is accepted by the server, or if a client accepts a join invitation from the server, the server then sends the shared global model to such a client, who subsequently joins the current round of federated learning. Our algorithm on the client side is more formally described in Algorithm 2. Such communication regarding invitations and requests must take place continuously between clients and the server; therefore, in order to implement democratic federated learning, the FL session must progress asynchronously.

Client Selection. When the list of participating clients reaches a predetermined threshold capacity, the server initiates client selection and sends the initial global model. The democratic federated learning algorithm ensures a swift initial client-server interaction, eliminating wait times in subsequent rounds. In an asynchronous FL session, client training times are crucial influential factors of the speed of model convergence. A client's training time is determined by the wall-clock time elapsed since it last received the global model, which includes the time it takes to communicate with the server. The server records each client's start time (when it sends the initial global model) and completion time (when it receives the client's update), maintaining a list of client training times. This enables accurate determination of client training times upon their arrival.

Additionally, in democratic FL, the data quality of clients is integral to the model's training performance. High-quality data from clients significantly boosts model accuracy by supplying more informative inputs, which assist the global model to capture patterns in the data distribution more effectively. Data quality is measured using its statistical utility [5], defined as the expected reduction in the global model loss with the inclusion of a client's data in the training process. Therefore, with our algorithm, the server preferentially selects clients with not only shorter training times but also higher data quality in each training round, thereby optimizing the training process and reducing the overall convergence time of the shared global model.

Asynchronous System Settings. By default, the entire democratic FL process is asynchronous. Because the server must keep two lists in the FL process, allowing clients to be able to join the FL session at any moment. In asynchronous FL, there are two crucial indicators, the first of which is the previously mentioned staleness, which represents the communication and processing speed of a client. Specifically, staleness is defined as a metric for the local model's outdatedness, presenting how much the client update lags behind the current global model. An update from a client with an outdated model is less useful than updates from clients with a smaller amount of staleness, and may even damage the global model [12] due to the staleness of its update in asynchronous FL. Another indicator is the minimal number of required client updates before server aggregation in each round. The server in FedAsync [10], for example, aggregates a client update once it is received; this is way too aggressive and it may not be able to converge to a global model with acceptable accuracy. In democratic FL, the experimental results of Port [12] can be directly utilized to determine the ideal minimum amount of client updates.

Dynamic Number of Epochs on the Clients in Each Round. To optimize for system heterogeneity in democratic FL, we allow the server to vary the number of epochs at the clients. This is achieved by limiting the staleness of clients as much as possible while allowing more clients to participate. This allows more data to be used for training, thereby avoiding a potential degradation of accuracy in the aggregated global model. To do this, the server should be capable of sending a customized proposal to the slower clients, so that they can stop training as soon as the current epoch completes. Such a customized proposal to slower clients not only speeds up the server's aggregation process in each round, but also prevents stale clients from wasting computational resources. With this way of varying the number of local training epochs at each of the clients, even slow clients can participate in the training process actively, expanding the total volume of training data used for democratic FL.

Algorithm Summary. The democratic FL algorithm begins by engaging a set of C clients, where in each round τ, the server distributes the global model ω_τ and clients work on their local models ω_τ^k. Clients possess datasets D^k, aggregating to total D, essential for determining each client's statistical util-

Algorithm 1 Democratic FL: server-side algorithm.

Require: The global model $\boldsymbol{w}_\tau$ in communication round τ; the local model $\boldsymbol{w}_\tau^k$ of client k at round τ; the total clients C; the number of clients for start-up FL C'; the minimum number of clients required for each round of aggregation K; the selected clients per round K'; the number of invited clients V; the number of received client updates R; the minimum number of clients for sending proposal S'; the number of proposals that server sending S; the total number of data samples D^k on each client k; and the total number of data samples D across all clients; the list for stores all clients' updates $\{\mathcal{R}_\tau\}$.

Wait for clients responses throughout entire FL mechanism...
while global accuracy $<$ target accuracy **do**
  Send the invitation to V clients
  ▷ $V \geq C'$
  Receive invitation response or join request from k
  if no recorded training times and data utility N^k **then**
    Add $N^k = None$ to ordered agreed clients set $\{\mathcal{C}'\}$
    ▷ Clients in $\{\mathcal{C}'\}$ are sorted by the normalized summation of clients' duration and statistical utility [5].
  end if
  Start democratic FL session
  Selected top K' clients to $\{\mathcal{K}'\}$ from start-up client set $\{\mathcal{C}'\}$. ▷ $|\{\mathcal{K}'\}| \leq C'$
  for client k in $\{\mathcal{K}'\}$ **do**
    Send global model $\boldsymbol{w}_\tau$
  end for
  if receives client updates $\boldsymbol{w}_\tau^k$ **then**
    Add $\boldsymbol{w}_\tau^k$ to $\{\mathcal{R}_\tau\}$
    Update N^k to $\{\mathcal{C}'\}$
  end if
  if $|\{\mathcal{R}_\tau\}| \geq S'$ **then** ▷ $S' < K$
    for client k in $\{\mathcal{K}' - \mathcal{R}_\tau\}$ **do**
      Send proposals to the top S clients from $\{\mathcal{K}'\}$
    end for
  end if
  if $|\{\mathcal{R}_\tau\}| = K$ **then**
    Do $\boldsymbol{w}_{\tau+1} := \sum_{k \in K} \frac{|D_k|}{|D|} \boldsymbol{w}_\tau^k$
    ▷ Aggregate K updates
  end if
end while

Algorithm 2 DEMOCRATIC FEDERATED LEARNING — CLIENT SIDE

Require: The client k; the global model $\boldsymbol{w}_\tau$

Wait server's invitation or send join request to the server...
if Received $\boldsymbol{w}_\tau$ **then**
  Do local training
  if Received server's proposal **then**
    Finished current training epoch
  end if
  Return update w_τ^k to the server
end if

ity. Initially, C' clients are selected randomly, and invitations are sent to V clients, with those accepting joining the *training clients* pool $\mathcal{C}'$. The server later ranks these clients by their statistical utility and training experience, sharing the global model with top K' clients for the next round.

It should be noted that for clients lacking any prior interaction with the server, and thus without existing data on the statistical utility or training times, selection occurs randomly via a uniform distribution. This continues until the number of clients which have recorded statistical utility and training times reaches C'. Subsequently, after the global model has been distributed to the training clients selected by the server, the server waits for their updates. After a while, when the server received updates to meet the number of clients' updates N' to trigger sending customized proposals, it chooses another top N clients out of all those who have not yet returned updates in the current round, and sends a customized proposal to remind them to return their own updates once their current local epoch is finished.

Finally, when the accumulation of client updates hits the minimum number of request aggregation K, assumed here as 10 according to FedBuff [11], the server initiates the aggregation process. Therefore, the first round of FL only aggregates 10 clients, triggering the commencement of the second round. In this round, the server employs the normalized summation of clients' statistical utility and training times to select $k' - K$ clients from $\{\mathcal{C}'\}$. The number of clients who were chosen (K') for training in the second round now equals 20 once more. The described process repeats iteratively, with the server fine-tuning the selection of clients based on their responsiveness and utility until the global model's accuracy reaches the predetermined target. This methodology not only streamlines the training process but also incorporates a degree of flexibility and client autonomy, characteristic of the democratic FL approach.

Key Differences from Related Work. Unlike previous efforts, our proposed democratic FL gives servers and clients the ability to freely enter and exit an FL training session. This not only allows the server to select clients but also enables both the server and clients to actively and simultaneously reject requests to join. In practical scenarios, many clients may not choose to engage due to insufficient computational resources or data at their disposal. Although strategies have been developed to shield against various attempts at reconstructing private data on the clients, and efforts have been made towards speeding up model convergence, existing strategies often overlook the clients' interests and choices.

In order to ensure the final convergence of the global model in such a free market, a well-designed algorithm must allow the system to operate asynchronously. In democratic FL, the status of clients is more diverse than in traditional FL. For instance, client A requests to join FL, and the server agrees; client B receives an invitation from the server and accepts it.

Both A and B appear to eventually contribute to the global model, but the server has historical data on client B prior to sending invitations to it, whereas client A may be an unknown client. The likelihood of the server rejecting A's join request is greater than that of B in future rounds.

Anarchic FL [4] also allows clients to join actively and it gives clients the right to choose different local epochs, but does not account for the fact that the server, in practice, may refuse the joining of stale clients for the convergence speed of the global model. Also, it does not confer upon the server the authority to alter the training epochs of clients; the server's role is solely to facilitate and expedite the convergence of the global model. Therefore, from the server's perspective, we have made modifications to dynamic local epochs that are more aligned with the server's objectives, making democratic FL a "win-win" solution for ensuring the interests of both servers and clients.

IV. Democratic FL: Convergence Analysis

To allow both servers and clients the flexibility to participate in the FL process, we first need to demonstrate the degree of flexibility that democratic FL can permit for clients and servers. Our intuition is that there is a bound for model convergence; when the number of samples decreases significantly, the model tends to fail to converge. However, convergence within asynchronous FL is more complex, involving factors such as epochs, staleness, and aggregation algorithms. In democratic FL, both epochs and staleness are variable, and the aggregation method involves averaging the received model parameters. Based on the above, we first outline the assumptions needed for our analysis.

Assumption 1 (Smoothness)**.** *Each objective function f_k of the client k is L-smooth. Thus its gradients are Lipschitz continuous with constant L, i.e., $\| g_k^r(\boldsymbol{w}) - \tilde{g}_k^r(\boldsymbol{w}') \| \leq L \| \boldsymbol{w} - \boldsymbol{w}' \|, \forall k \in [C]$, where g_k^r denotes the gradient ∇f_k and $\tilde{g}_k^r = E[g_k^r]$.*

Assumption 2 (Unbiased Local Gradient)**.** *Let ξ^k be sampled from the k-th device's local data uniformly at random. The local stochastic gradient is unbiased, i.e., $\mathbb{E}\left[g_k^r(\boldsymbol{w}, \xi^k)\right] = g_k^r(\boldsymbol{w}), \forall k \in [C]$, where $\boldsymbol{w}$ denotes trainable parameters.*

Assumption 3 (Bounded Local and Global Variances)**.** *Two non-negative constants, denoted as σ_L and σ_G, can be found such that the squared variance of each local stochastic gradient estimator is bounded by $\mathbb{E}\left[\left\|g_k^r(\boldsymbol{w}, \xi^k) - g_k^r(\boldsymbol{w})\right\|^2\right] \leq \sigma_L^2, \forall k \in [C]$; and the global gradient is bounded by $\|g_k^r(\boldsymbol{w}) - g(\boldsymbol{w})\|^2 \leq \sigma_G^2, \forall k \in [C]$.*

In our exploration, we utilize the general partial participation framework from Horváth & Richtárik [13]. Within a client set, a subset is chosen through a probabilistic method called sampling ($\mathbb{S}$). This mechanism selects from the potential 2^n subsets of $[n]$. Corresponding to each $\mathbb{S}$ is a matrix ($\mathbf{P}$), where the element $\mathbf{P}_{ij}$ indicates the probability that clients i and j are both in the sample. We can derive a vector, $p = (p_1, \ldots, p_n)$, from $\mathbf{P}$'s diagonal, representing individual client inclusion probabilities. A sampling is termed proper if every $p_i > 0$. The expected number of participants in a communication round, represented by b, is either the sum of vector values in p or the trace of $\mathbf{P}$. With probabilities ranging in [0,1] for each client, the choice to include a client in $\mathbb{S}$ is independent of others, a method labeled as independent sampling. Thus, we set p_r^k as the participating probability of the client k in round r, and there are C^r participating clients. In addition, let τ_r^k denote the staleness for client k at round r. And, q and j denote the index of the local epoch while Q is the total local epochs. Thus, Q_r^k is local epochs for client k at round r. Before performing the convergence analysis on $E[f(\boldsymbol{w}_{r+1})] - f(\boldsymbol{w}_r)$, where $f(\boldsymbol{w}_r) = \sum_k p_k f\left(\boldsymbol{w}_r^k\right)$ is the global objective and $\boldsymbol{w}_r$ is the global model in round r, we first introduce the following lemmas.

Lemma 1. *The expected divergence between the accumulated local gradients $G_k^r\left(\boldsymbol{w}_r^k\right)$ and $\tilde{G}_k^r\left(\boldsymbol{w}_r^k\right)$ has an upper bound $Q_r^k \sum_{q=0}^{Q_r^k-1} \eta_q^2 \sigma_L^2$, where η_q is the learning rate. After considering the p^k as the weight for each client, the upper bound becomes $\sum_{k \in C_r} \left(p_r^k\right)^2 Q_r^k \sum_{q=0}^{Q_r^k-1} \eta_q^2 \sigma_L^2$, where $G_k^r\left(\boldsymbol{w}_r^k\right) = \sum_{q=0}^{Q_r^k-1} \eta_q g_k^r\left(\boldsymbol{w}_{r,q}^k\right)$, $E\left[G_k^r\left(\boldsymbol{w}_r^k\right)\right] = \tilde{G}_k^r\left(\boldsymbol{w}_r^k\right) = \sum_q \eta_q \tilde{g}_k^r\left(\boldsymbol{w}_{r,q}^k\right)$.*

Proof. Firstly, $E\|\triangle\|^2 = E\left\|\sum_{q=0}^{Q_r^k-1} \eta_q \left(g_k^r - \tilde{g}_k^r\right)\right\|^2$, where $\triangle = G_k^r\left(\boldsymbol{w}_r^k\right) - \tilde{G}_k^r\left(\boldsymbol{w}_r^k\right)$. With Cauchy-Schwartz inequality, the bound becomes $Q_r^k \sum_{q=0}^{Q_r^k-1} \eta_q^2 E\|g_k^r - \tilde{g}_k^r\|^2$. The proof can be completed after using Assumption 3. Similarly, we have $E\left\|\sum_{k \in C_r} p_r^k \triangle\right\|^2 \leq \sum_{k \in C_r} \left(p_r^k\right)^2 E\|g_k^r - \tilde{g}_k^r\|$. Thus, by using the previous result and the Assumption 3, we can get the upper bound. □

Lemma 2. *During each communication round, the expected divergence in weights between $\boldsymbol{w}_r$ and $\boldsymbol{w}_{r+1}$ has an upper bound, represented as*

$$\frac{L}{2} E\|\boldsymbol{w}_{r+1} - \boldsymbol{w}_r\|^2 \leq L \sum_k \left(p_r^k\right)^2 Q_r^k \sum_q \eta_q^2 \sigma_L^2 + L \cdot E\left\|\sum_k p_r \tilde{G}_k^r\right\|^2$$

Proof. Firstly, $\triangle = \boldsymbol{w}_{r+1} - \boldsymbol{w}_r = \sum_{k \in C^r} p_r^k \left(-G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right)$. And we have the summation of gradients which is denoted as $G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right) = \sum_{q=0}^{Q_{r-\tau_r^k}^k-1} \eta_q g_k^r\left(\boldsymbol{w}_{r-\tau_r^k,q}^k\right)$. Therefore,

we get the following equations:

$$
\begin{aligned}
E\left\|\triangle\right\|^2 &= E\left\|\sum_{k\in C^r} p_r^k G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2 \\
&= E\left\|\sum_{k\in C^r} p_r^k\left(G_k^r - \tilde{G}_k^r + \tilde{G}_k^r\right)\right\|^2 \\
&\leq 2E\left\|\sum_{k\in C^r} p_r^k\left(G_k^r - \tilde{G}_k^r\right)\right\|^2 + 2E\left\|\sum_{k\in C^r} p_r^k \tilde{G}_k^r\right\|^2 \\
&\leq 2\sum_k \left(p_r^k\right)^2 Q_r^k \sum_q \eta_q^2\sigma_L^2 + 2E\left\|\sum_{k\in C^r} p_r^k \tilde{G}_k^r\right\|^2 .
\end{aligned}
$$

We introduce the zero term to obtain the first equation. And the first inequality is obviously obtained, while the second inequality is derived from Lemma 1. After multiplying both sides by $\frac{L}{2}$, we get the desired result. □

Lemma 3. *The divergence expectation between the global weight $\boldsymbol{w}_r$ in round r and the local weight $\boldsymbol{w}_{r-\tau_r^k}^k$ in round $r-\tau_r^k$ has the upper bound, represented as:*

$$
\begin{aligned}
&E\left\|\boldsymbol{w}_r - \boldsymbol{w}_{r-\tau_r^k}^k\right\|^2 \leq 2\Gamma_{\max,k}\sum_{t=r-\tau_r^k}^{r-1} \\
&\left[\sum_{m\in C_t}\left(p_t^m\right)^2 J_t^m \sum_{j=0}^{J_t^m-1}\eta_j^2\sigma_L^2 + 2E\left\|\sum_{m\in C_t} p_t^m \tilde{G}_m^t\left(\boldsymbol{w}_t^m\right)\right\|^2\right]
\end{aligned}
$$

where $\Gamma_{\max,k}$ is maximum staleness for client k, J_t^m is the local epochs for client m at round t, η_j is the learning rate.

Proof. We can directly have the following equations:

$$
\begin{aligned}
&E\left\|\boldsymbol{w}_r - \boldsymbol{w}_{r-\tau_r^k}^k\right\|^2 = E\left\|\sum_{t=r-\tau_r^k}^{r-1}\left(\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\right)\right\|^2 \\
&\leq \tau_r^k \sum_t E\left\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\right\|^2 \quad \triangleright Cauchy - Schwarz \\
&= \tau_r^k \sum_t E\left\|\sum_{m\in C_t} p_t^m G_m^t\left(\boldsymbol{w}_t^m\right)\right\|^2 \\
&\quad \Downarrow Lemma2 \\
&\leq 2\tau_r^k \sum_t\left[\sum_m\left(p_t^m\right)^2 J_t^m \sum_j \eta_j^2\sigma_L^2 + E\left\|\sum_m p_t^m \tilde{G}_m^t\right\|^2\right] \\
&\quad \Downarrow \tau_r^k \leq \Gamma_{\max,k} \\
&\quad \leq 2\Gamma_{\max,k}\sum_{t=r-\tau_r^k}^{r-1} \\
&\quad \left[\sum_{m\in C_t}\left(p_t^m\right)^2 J_t^m \sum_{j=0}^{J_t^m-1}\eta_j^2\sigma_L^2 + 2E\left\|\sum_{m\in C_t} p_t^m \tilde{G}_m\left(\boldsymbol{w}_t^m\right)\right\|^2\right]
\end{aligned}
$$

□

Lemma 4. *Taking into account the staleness, the divergence expectation between the gradient $g\left(\boldsymbol{w}_r\right)$ and the accumulated gradient $\tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)$ has an upper bound, represented as:*

$$
\begin{aligned}
&3L^2\underbrace{\sum_k\left(p_r^k\right)^2 E\left\|\boldsymbol{w}_r - \boldsymbol{w}_{r-\tau_r^k}^k\right\|^2}_{Lemma3} + 3\sum_k\left(p_r^k\right)^2\sigma_G^2 \\
&+ 3\sum_k\left(p_r^k\right)^2 L^2\left[5Q_{r-\tau_r^k}^k\eta_0^2\left(\sigma_L^2 + 6Q_{r-\tau_r^k}^2\sigma_G^2\right)\right. \\
&\left.+30Q_{r-\tau_r^k}^k\eta_0^2\left\|\nabla f\left(\boldsymbol{w}_{r-\tau_r^k}\right)\right\|^2\right]
\end{aligned}
$$

where η_0 is the initial learning rate.

Proof. We have $\triangle = E\left\|g\left(\boldsymbol{w}_r\right) - \sum_{k\in C_r} p_r^k\tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2$, then

$$
\begin{aligned}
\triangle &= E\left\|\sum_{k\in C_r} p_r^k g\left(\boldsymbol{w}_r\right) - \sum_{k\in C_r} p_r^k\tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2 \\
&\leq \sum_{k\in C_r}\left(p_r^k\right)^2 E\left\|g\left(\boldsymbol{w}_r\right) - \tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2 \\
&= \sum_k\left(p_r^k\right)^2 E\left\|g\left(\boldsymbol{w}_r\right) - g\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right. \\
&\quad + g\left(\boldsymbol{w}_{r-\tau_r^k}^k\right) - G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right) \\
&\quad \left.+G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right) - \tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2 \\
&\leq \sum_k\left(p_r\right)^2\left[3E\left\|g\left(\boldsymbol{w}_r\right) - g\left(w_{r-\tau_r^k}^k\right)\right\|^2\right. \qquad (1) \\
&+ 3E\|g\left(w_{r-\tau_r^k}^k\right) - G_k^r\left(w_{r-\tau_r^k}^k\right\|^2 \\
&\left.+\ 3E\left\|G_k^r\left(W_{r-\tau_r^k}^k\right) - \tilde{G}_k^r\left(W_{r-\tau_r^k}^k\right)\right\|^2\right] \\
&\leq 3L^2\underbrace{\sum_k\left(p_r^k\right)^2 E\left\|\boldsymbol{w}_r - \boldsymbol{w}_{r-\tau_r^k}^k\right\|^2}_{Lemma3} + 3\sum_k\left(p_r^k\right)^2\sigma_G^2 \\
&+ 3\sum_k\left(p_r^k\right)^2\underbrace{E\left\|G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right) - \tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2}_{\triangle_1}
\end{aligned}
$$

The second inequality is obtained by using Cauchy-Schwarz $\left(1^2+\cdots+1^2\right)^2\left(X_1^2+\cdots+X_n^2\right) \geqslant \left(X_1+\cdots+X_n\right)^2$ and the final inequality is achieved based on L-smoothness and bounded global divergence.

$$
\begin{aligned}
\triangle_1 &= E\left\|G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)-\tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2 \\
&= E\left\|G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)-\sum_{q=0}^{Q_{r-\tau_r^k}^k-1}\eta_q\tilde{g}_k^r\left(\boldsymbol{w}_{r-\tau_r^k,q}^k\right)\right\|^2 \\
&\text{Setting learning rates } \eta_0 \geqslant \eta_1 \geqslant \cdots \geqslant \eta_{Q_{r-\tau_r^k}^k} \geqslant \frac{1}{Q_{r-\tau_r^k}^k} \\
&\leq E\left\|G_k^r(\boldsymbol{w}_{r-\tau_r^k}^k)-\sum_{q=0}^{Q_{r-\tau_r^k}^k-1}\frac{1}{Q_{r-\tau_r^k}^k}\tilde{g}_k^r\left(\boldsymbol{w}_{r-\tau^k(r),q}^k\right)\right\|^2 \\
&= \frac{1}{Q_{r-\tau_r^k}^k}\sum_q E\left\|g_k^r(\boldsymbol{w}_{r-\tau_r^k}^k)-\tilde{g}_k^r(\boldsymbol{w}_{r-\tau_r^k,q}^k)\right\|^2 \\
&\leq \frac{L^2}{Q_{r'}}\sum_q E\left\|\boldsymbol{w}_{r-\tau_r^k}^k-W_{r-\tau_r^k,q}^k\right\|^2 \\
&\leq L^2\left[5Q_{r-\tau_r^k}^k\eta_0^2\left(\sigma_L^2+6Q_{r-\tau_r^k}^2\sigma_G^2\right)\right. \\
&\left.+30Q_{r-\tau_r^k}^k\eta_0^2\left\|\nabla f\left(\boldsymbol{w}_{r-\tau_r^k}\right)\right\|^2\right]
\end{aligned}
$$

where the final inequality is obtained by using the proof in page 14 of the work [4]. □

Eventually, when the staleness has an upper bound denoted as $\forall\tau_r^k \leq Q_{\max}$, we have the theoretical convergence guarantee and obtain an ergodic convergence rate.

Theorem 1. *Given an initial learning rate η_0 satisfying $\eta_0 \geqslant \frac{1}{Q_r^k}$, the global model iterates of our proposed algorithm achieves the convergence rate:*

$$
\mathcal{O}\left(\frac{K^{\frac{1}{2}}}{Q_{\max}^{\frac{1}{2}}R^{\frac{1}{2}}}\right)+\mathcal{O}\left(\frac{\Gamma_{\max}^2}{RQ_{\max}}\right)+\mathcal{O}\left(\frac{K^2Q_{\max}^{\frac{1}{2}}}{R}\right)+\mathcal{O}\left(\sigma_G^2\right) \tag{2}
$$

where $\Gamma_{\max}$ is the maximum staleness, K is the largest number of clients to be selected in each round, $Q_{\max}$ is the maximum local epochs, and R is the total communication rounds. The final term, $\mathcal{O}\left(\sigma_G^2\right)$, is introduced by potentially unbalanced and biased client contributions.

Proof. Using L-smoothness, we have the following smoothness inequality:

$$
\begin{aligned}
E\left[f\left(\boldsymbol{w}_{r+1}\right)\right] \leq f\left(\boldsymbol{w}_r\right) &+ \underbrace{\left\langle g\left(\boldsymbol{w}_r\right), E\left[\boldsymbol{w}_{r+1}-\boldsymbol{w}_r\right]\right\rangle}_{\triangle} \\
&+ \underbrace{\frac{L}{2}E\left\|\boldsymbol{w}_{r+1}-\boldsymbol{w}_r\right\|^2}_{Lemma2}
\end{aligned} \tag{3}
$$

where

$$
\begin{aligned}
\triangle &= \left\langle g\left(\boldsymbol{w}_r\right), E\left[\sum_{k\in C_r}p_r^k-G_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right]\right\rangle \\
&= -E\left[\left\langle g\left(\boldsymbol{w}_r\right), \sum_{k\in C_r}p_r^kG_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\rangle\right] \\
&\Downarrow \langle a,b\rangle = \frac{1}{2}\left(\|a\|^2+\|b\|^2-\|a-b\|^2\right) \\
&- \frac{1}{2}\left\|g\left(\boldsymbol{w}_r\right)\right\|^2-\frac{1}{2}E\left\|\sum_{k\in C_r}p_r^k\tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2 \\
&+ \underbrace{\frac{1}{2}E\left\|g\left(\boldsymbol{w}_r\right)-\sum_{k\in C_r}p_r^k\tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2}_{Lemma4}
\end{aligned} \tag{4}
$$

Therefore, after introducing the Lemma 2, Lemma 3, and Lemma 4 into Eq. 3, we get the $E\left[f\left(\boldsymbol{w}_{r+1}\right)\right]-f\left(\boldsymbol{w}_r\right)$ as follows:

$$
\begin{aligned}
&- \frac{1}{2}\left\|g\left(\boldsymbol{w}_r\right)\right\|^2-\frac{1}{2}E\left\|\sum_{k\in C_r}p_r^k\tilde{G}_k^r\left(\boldsymbol{w}_{r-\tau_r^k}^k\right)\right\|^2 \\
&+ 3L^2\Gamma_{\max,k}\sum_{t=r-\tau_r^k}^{r-1} \\
&\left[\sum_{m\in C_t}\left(p_t^m\right)^2J_t^m\sum_{j=0}^{J_t^m-1}\eta_j^2\sigma_L^2+2E\left\|\sum_{m\in C_t}p_t^m\tilde{G}_m^t\left(\boldsymbol{w}_t^m\right)\right\|^2\right] \\
&+ \frac{3}{2}\sum_k\left(p_r^k\right)^2\sigma_G^2 \\
&+ \frac{3}{2}\sum_k\left(p_r^k\right)^2L^2\left[5Q_{r-\tau_r^k}^k\eta_0^2\left(\sigma_L^2+6Q_{r-\tau_r^k}^2\sigma_G^2\right)\right. \\
&\left.+45Q_{r-\tau_r^k}^k\eta_0^2\left\|\nabla f\left(\boldsymbol{w}_{r-\tau_r^k}\right)\right\|^2\right] \\
&\frac{L^2}{2}\sum_k\left(p_r^k\right)^2Q_r^k\sum_q\eta_q^2\sigma_L^2+\frac{L^2}{2}\cdot E\left\|\sum_k p_r\tilde{G}_k^r\right\|^2
\end{aligned} \tag{5}
$$

Subsequently, we sum up the above inequality from round $r=0$ to round $r=R$ and get

$$
E\left[f\left(\boldsymbol{w}_R\right)\right]-f\left(\boldsymbol{w}_0\right)=\sum_{r=0}^{R}E\left[f\left(\boldsymbol{w}_{r+1}\right)\right]-f\left(\boldsymbol{w}_r\right)
$$

Computing this summation leads to the final ergodic convergence rate. The proof is done. □

The obtained Theorem 1 gives us four insights into the implementation and application of our proposed algorithm. Firstly, our algorithm guarantees convergence under any staleness as the second term decreases with the increase of the communication round. Secondly, it ensures a fast convergence when the local epochs of clients are changed and different. In the extreme case, the first term $\mathcal{O}\left(\frac{K^{\frac{1}{2}}}{Q_{\max}^{\frac{1}{2}}T^{\frac{1}{2}}}\right)$ decreases

accordingly with the increase of Q. Even though the third term increases with Q, the rate gradually decreases when the communication round T increases. Thirdly, Eq. 2 shows that involving more clients in each round of the training process leads to a slower convergence rate. Specifically, choosing a large K can significantly increase the first and third terms of Eq. 2. In particular, the third term $\mathcal{O}\left(\frac{K^2 Q_{\max}^{\frac{1}{2}}}{T}\right)$ has a higher growth rate. This meets the practical scenario because when more clients are involved in training, the data heterogeneity and staleness situation introduce more challenges in the model update and aggregation. Finally, increasing the maximum local update $Q_{\max}$ to allow clients to perform more local updates on the model may not always lead to a better convergence rate. This is attributed to the fact that the number of clients K cooperates with $Q_{\max}$ to determine the convergence rate simultaneously. For example, a large $Q_{\max}$ leads to the lower values of the first two terms of Eq. 2. However, the third term may increase significantly due to the K^2 multiplied by $Q_{\max}$.

In addition, Theorem 1 also exposes the importance of selecting proper K and $Q_{\max}$ for the model training under the federated paradigm that suffers from staleness concern. With a constant local update, involving fewer clients in the training leads to a better convergence rate. However, substantially increasing the local epochs does not guarantee a faster convergence rate, as the increase in K^2 within the third term can exacerbate the convergence when $Q_{\max}$ is large. This analysis suggests that optimizing the practical application of our algorithm requires a balanced consideration of model quality, the number of local updates, and client participation.

V. Performance Evaluation and Comparisons

With our new paradigm of democratic FL, the algorithms need to be designed on both the clients and the server to make adequate decisions on whether invitations from the other side should be accepted or declined. By providing a greater degree of autonomy to both clients and the server, we wish to balance convergence speed and autonomy. The design of these algorithms affects the convergence behaviour of the entire FL training session, and the time it takes for the model to converge should be empirically evaluated in our comparison of algorithms.

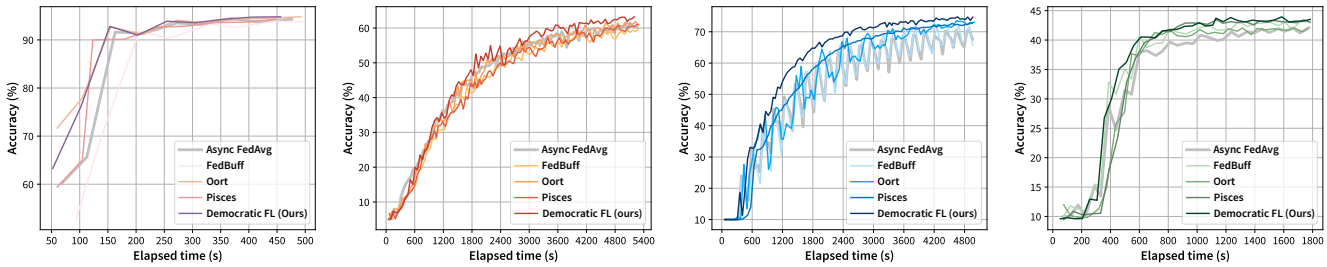
In our experimental setup, we established a pool of 100 clients, from which a subset of 20 clients were selected for participation in each training round. Due to the asynchronous mode of operation, only 10 clients were actively involved in the aggregation process during each round. For experiments involving the `CIFAR-10` dataset paired with the `ResNet-18` model and the `CINIC-10` dataset with the `VGG-16` model, we utilized a NVIDIA RTX A4500 GPU, boasting 20 GB of CUDA memory and operating on CUDA version 11.7. In addition, for the `MNIST` and `Federated Extended MNIST` datasets, training was executed on an Apple M2 Max CPU, featuring 96 GB of unified memory. The data distribution across clients adhered to a non-independent and identically distributed (non-i.i.d.) scheme, orchestrated via a Dirichlet process with a concentration parameter set to 1.

In our performance comparison, we have comprehensively tested several asynchronous algorithms across four distinct datasets. Asynchronous algorithms are characterized by their variability since they guarantee only a minimum number of clients joining aggregations per round, leading to somewhat fluctuating convergence curves. Particularly notable were the performance of Asynchronous FedAvg and FedBuff. These two algorithms are fundamentally simplistic and differ primarily in their aggregation methodologies. While the FedAvg algorithm employs a weighted average based on the number of data samples, FedBuff aggregates based on the number of participating clients. Observations from the `MNIST` and `FedMNIST` datasets, which utilize the relatively uncomplicated `LeNet-5` models, revealed that the performances of FedBuff and FedAvg were nearly identical.

Oort and Pisces slightly outperformed FedAvg and FedBuff, likely because they select clients based on data quality rather than at random. Oort calculates the sum of squares over each sample's loss and, considering clients' training time, has devised sophisticated methods for adding or removing clients from the pool. Pisces altered the computation of model loss for client selection with metrics like data utility and staleness, which indicates the age of the clients' models relative to the current global model round. Both Oort and Pisces maintain a blacklist for subpar clients, excluding them permanently from selection. democratic FL also employs a similar "blacklist" concept. However, it not only excludes underperforming clients but also those who refuse to join the server's training sessions, thus protecting their autonomy. Democratic FL selects clients based on the highest statistical utility and shortest training and communication time, which has proven beneficial for the algorithm based on performance comparisons with its competitors.

The performance gap between democratic FL and its competitors was more pronounced with the more complex `CIFAR-10` dataset using `ResNet-18`. Here, FedAvg and FedBuff fared the worst, with very similar convergence curves. Pisces showed slightly faster convergence than FedAvg and FedBuff but still exhibited an unstable trend, which is always happening in asynchronous settings. Surprisingly, democratic FL demonstrated an accuracy rate which grows much faster than Oort, despite having similar curves.

Our hypothesis considered a scenario where clients could decline participation in FL training. When servers send global models to these non-participatory clients, a substantial amount of network bandwidth cost is incurred, leading to a longer training session. Democratic FL proactively sends invitations to preferred clients and accepts requests from those willing to join before training begins, effectively pre-selecting available clients. In our experimental setup, available clients were set to 90% of the total, with server-invited clients and those wishing to join training each accounting for 80% of the total. Decreasing these parameters would increase the advantages of democratic FL, and in real-world settings, values of these



(a) Training LeNet-5 on MNIST (b) Training LeNet-5 on FedMNIST (c) Training ResNet-18 on CIFAR-10 (d) Training VGG-16 on CINIC-10

Fig. 1. DEMOCRATIC FEDERATED LEARNING vs. its competitors: a performance comparison with the MNIST, FedMNIST, CIFAR-10 and CINIC-10, and a mild non-i.i.d. Dirichlet data distribution.

parameters would not be as high as in our experiments.

Furthermore, democratic FL incorporates an urgent notification mechanism that enables the server to modify the number of training epochs for the clients. When the server receives a certain number of model updates, which is typically set to be less than the minimum number of client updates required for server aggregation, it sends a notification to the remaining clients which are still in training. This notification instructs them to return their model updates after completing their current epoch, rather than completing the pre-established number of epochs. This mechanism further reduces the time required for a client to complete a round of training on the client side. Moreover, as the server has already received a substantial portion of the model updates of client, for instance, 70% of the minimum number required for server aggregation, the precision of the aggregated model is not compromised even if the remaining updates have fewer epochs than initially specified. On the contrary, the overall progress of each federated learning round is expedited by the slower clients forgoing the full number of prescribed training epochs.

Regarding the CINIC-10 dataset with VGG-16, the performance of the five asynchronous algorithms was similar. Democratic FL consistently had the fastest convergence rate, followed by Pisces, FedBuff, Oort, and lastly, FedAvg. We speculate that FedAvg's subpar performance could be due to it aggregating fewer data samples per round in asynchronous settings compared to the synchronous FL. FedBuff, designed for asynchronous mode and using client counts as weighted averages, is less affected by asynchronous conditions. While Oort and Pisces also consider client selection, accelerating convergence further, they do not account for client *rights*, with clients fully controlled by the server. Democratic FL, on the other hand, offers clients the freedom to follow their own will, actively joining or declining FL training, making it more prepared for practical implementation.

VI. CONCLUDING REMARKS

In this paper, we proposed *democratic federated learning*, a new approach that substantially redesigned the way that clients and the server interact with each other in federated learning. With democratic federated learning, both clients and the server are on equal footing with the same privileges, and have the freedom to decline the other party's requests. Our experimental results across various datasets illustrate that democratic FL surpasses existing asynchronous methods in both convergence speed and model accuracy. The client selection algorithm central to democratic FL effectively harnesses client autonomy and computational resources, leading to more efficient training rounds. Our proposed algorithm encourages more practical deployment of federated learning, as only motivated and high-quality clients would participate in FL training sessions.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. Artificial Intelligence and Statistics (AISTATS)*, 2017.

[2] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, and S. Guo, "A Survey of Incentive Mechanism Design for Federated Learning," *IEEE Transactions on Emerging Topics in Computing*, 2021.

[3] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to Smartphones: Incentive Mechanism Design for Mobile Phone Sensing," in *Proc. International Conference on Mobile Computing and Networking (MobiCom)*, 2012, pp. 173–184.

[4] H. Yang, X. Zhang, P. Khanduri, and J. Liu, "Anarchic Federated Learning," in *Proc. International Conference on Machine Learning (ICML)*. PMLR, 2022, pp. 25 331–25 363.

[5] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient Federated Learning via Guided Participant Selection," in *Proc. Symposium on Operating Systems Design and Implementation (OSDI)*, 2021, pp. 19–35.

[6] Z. Jiang, W. Wang, B. Li, and B. Li, "Pisces: Efficient Federated Learning via Guided Asynchronous Training," in *Proc. ACM Symposium on Cloud Computing (SoCC)*, 2022.

[7] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards Flexible Device Participation in Federated Learning," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2021, pp. 3403–3411.

[8] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," in *Proc. IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[9] D. Avdiukhin and S. Kasiviswanathan, "Federated Learning under Arbitrary Communication Patterns," in *Proc. International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 425–435.

[10] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous Federated Optimization," in *Proc. NeurIPS Workshop on Optimization for Machine Learning (OPT)*, 2020.

[11] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated Learning with Buffered Asynchronous Aggregation," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2022, pp. 3581–3607.

[12] N. Su and B. Li, "How Asynchronous can Federated Learning Be?" in *Proc. International Symposium on Quality of Service (IWQoS)*, 2022.

[13] S. Horváth and P. Richtárik, "A better alternative to error feedback for communication-efficient distributed learning," *arXiv preprint arXiv:2006.11077*, 2020.