Enhancing Federated Learning with Self-Determining Mechanism in MEC

Ruixin Li*, Chun Wang*, Zibin Zheng†, Huawei Huang†
*School of CSE, Sun Yat-Sen University, China. †SSE, Sun Yat-Sen University, China.
(Corresponding: Huawei Huang) Email: huanghw28@mail.sysu.edu.cn

***Abstract*—Federated learning (FL) has gained substantial attention as a promising solution to the need for client privacy in mobile edge computing (MEC). However, FL suffers from instability of accuracy because of the invalid clients who become stragglers caused by frequent fluctuation of available resources in MEC. To tackle the challenge, most of the frameworks of asynchronous FL allow the parameter server (PS) to schedule clients reasonably. This kind of central server-determining paradigm makes it difficult to select all potentially useful clients because their resources in MEC vary frequently. In this paper, we proposed a new semi-asynchronous FL framework with a self-determining mechanism. This kind of framework fully exploits useful local models to improve global accuracy. Our proposed system has the following notable properties. Firstly, since clients perceive their resource status, thereby the self-determining clients can autonomously determine by themselves whether to participate in FL training according to the resource status. Secondly, compared the experimental results with other baselines, our proposed framework significantly improves the average global test accuracy.**

***Index Terms*—Federated Learning, Self-Determining, Mobile Edge Computing**

I. Introduction

Federated learning (FL) is a distributed machine learning paradigm where a number of clients train a global model collaboratively following the orchestration of a central parameter server without sharing their local private data [1]–[4]. As the most famous FL framework, FedAvg [3] applies data parallelism mode of training to train a data-hungry large model based on the increasing mountains of client data [2].

There are three main phases in the synchronous FL [4], including *Selection Phase*, *Local-Training Phase*, and *Reporting Phase*. In the synchronous FL, the parameter server (PS) and all clients follow the 3 phases mentioned above. In *Selection Phase*, the PS selects several clients and assigns the global model to them. Then, in *Local-Training Phase*, clients that are selected by PS receive the global model and start local training. Finally, in *Reporting Phase*, clients upload local updates to the parameter server. When the PS aggregates local models, the global model update is complete for a round of training. Such a classical learning paradigm seems simple but works. However, FL faces some technical challenges. For example, some devices (i.e., clients) in the MEC environment are not able to afford computing-intensive tasks [5] due to the resource-limited MEC environment; those mobile devices might fail to upload their local updates in *Reporting Phase* due to their mobility.

Although client devices can keep in contact with the PS by exploiting the 5G/B5G network [6], the biggest problem is how to ensure the availability of the resource-limited mobile clients in the MEC environment consisting of highly dynamic uncertainties [7]. These dynamic uncertainties can be caused by the frequent changes of client resources, mainly referring to the computing power of devices and the stability of the network connection. Such uncertainties can lead to the following two kinds of clients: (a) the slow clients, also called *stragglers*, that fail to complete their training tasks before the specific deadline; and (b) the invalid clients that lose contact with the parameter server [1]. The absence of those unavailable clients aggravates clients' drift and degrades the global accuracy of FL [8].

To conquer the problem of the hysteresis above of decision-making, non-synchronous protocols including asynchronous FL [9], [10] and semi-asynchronous FL [11], [12] have been proposed. For the semi-asynchronous FL, the PS follows a series of certain phases but clients just manage themselves without any phases. In this paper, we classify semi-asynchronous FL into two phases for the PS, i.e., *Local-Training Phase* and *Global Update Phase*. Firstly, in *Local-Training Phase*, clients can request the latest global model from the server, then start their local training and upload the local updates when finished. In contrast, the PS in *Local-Training Phase* just keeps the locally updated model in the cache and should not perform aggregation. In *Global Update Phase*, the PS stops both distributing the global model and receiving the local updates as well, so that it concentrates on aggregation for a new global model. Meanwhile, clients are still allowed to participate in local training based on the previously received global model.

The major contributions of our study in this paper can be summarized as follows.

- We propose a new FL framework that can improve the global test accuracy of the semi-asynchronous FL in MEC by applying a self-determining mechanism instead of centralized selection. We then provide a detailed analysis of the advantages in the aspect of convergence performance.
- The experimental results show that our proposed framework outperforms other baselines in terms of converged performance.

TABLE I
SYMBOLS AND NOTATIONS

K	the size of all FL rounds. $\forall k \in [K]$, k denotes round ID.
N	the number of all clients. $\forall i \in [N]$, i denotes client ID.
$\mathcal{S}(M_k, k)$	the set of candidates with the size M_k at round k.
$\zeta_{i,k}$	current network connection quality of client i at round k
$\vartheta_{i,k}$	current computing capability of client i at round k
$d_{i,k}$	currently-observed download rate of client i at round k.
$u_{i,k}$	currently-observed upload rate of client i at round k.
$\gamma_{i,k}$	currently-observed available memory of client i at round k.
$\psi_{i,k}$	currently-observed CPU frequency of client i at round k.
$\phi_{i,k}$	currently-observed remaining battery of client i at round k.
$\Omega_{i,k}$	current status $[\gamma_{i,k}, \psi_{i,k}, u_{i,k}, d_{i,k}, \phi_{i,k}, \zeta_{i,k}, \vartheta_{i,k}]$.
$\mathcal{D}_i$	the set of raw data owned by client i.
y_k	global model of which the version is k (i.e., at round k).
$x_{i,k}$	local model learned by participant i in round k.
$\tau_{i,k}$	Staleness of a local update from client i at round k.
τ_{max}	Maximum of tolerance of staleness for local updates.
$T_d(k)$	a fixed value that represents the time span of an FL round.
$\chi_{i,k}$	a binary variable that indicates whether client i participates in FL or not at round k.

II. SYSTEM DESIGN

A. Problem Formulation

The major symbols and notations used in this paper are given in Table I. This section formulates the problem that we study for semi-asynchronous federated learning. We define $F_{i,k}(x_{i,k})$ (where $i \in [N]$ and $k \in [K]$) to denote the loss value of local model $x_{i,k}$, and $f(x_{i,k}; d_{i,j})$ denote the loss value of each step of gradient descent while utilizing the data sample $d_{i,j} \in \mathcal{D}_i$. Then, Equation (1) shows the loss function of client $i \in [N]$ at round $k \in [K]$.

$$F_{i,k}(x_{i,k}) = \frac{1}{|\mathcal{D}_i|} \sum_{d_{i,j} \in \mathcal{D}_i} f(x_{i,k}; d_{i,j}), \forall i \in [N], k \in [K]. \tag{1}$$

Then, the local model update of client i at round k is written as

$$x_{i,k+1} = x_{i,k} - \eta \nabla F_{i,k}(x_{i,k}), \forall i \in [N], k \in [K], \tag{2}$$

where η denotes learning rate and $\nabla F_{i,k}(x_{i,k})$ denotes the gradient at round $k \in [K]$. According to the local loss function Equation (1), the global loss function and the objective function of FL are defined as

$$F_k(\boldsymbol{y}_k) = \frac{1}{\sum\limits_{i \in \mathcal{S}_k^M} |\mathcal{D}_i|} \sum_{i \in \mathcal{S}_k^M} |\mathcal{D}_i| F_{i,k}(x_{i,k}), \forall k \in [K], \tag{3}$$

$$\min_{k \in [K]} F(\boldsymbol{y}_k) - F(\boldsymbol{y}^*), \tag{4}$$

where $|\mathcal{D}_i|$ denotes the size of the local dataset of client $i \in [N]$, and $\boldsymbol{y}^*$ denotes the optimal global model whose loss value is zero (i.e., the ground-truth labels). Note that, $\mathcal{S}_k^M$ denotes the set of FL participants consisting of a number $M (\in \mathbb{N}^+)$ of clients at round $k \in [K]$. Equation (3) also represents the aggregation of local updates, after which the global Model Update is performed referring to the following equation

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k - \eta \nabla F_k(\boldsymbol{y}_k), \forall k \in [K], \tag{5}$$

where $\boldsymbol{y}_k$ denotes the global model at round k, η denotes the learning rate, and $\nabla F_k(\boldsymbol{y}_k)$ denotes the global gradient calculated by global loss function defined in Equation (3).

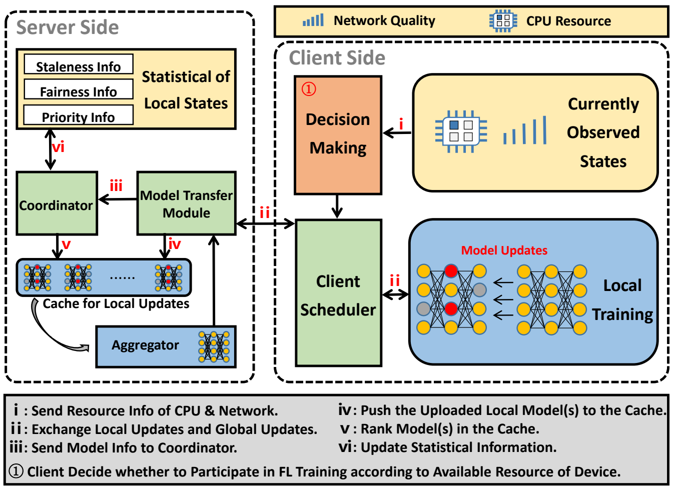

Server Side
Staleness Info
Fairness Info
Priority Info
Statistical of Local States
vi
Coordinator
iii
Model Transfer Module
v
iv
......
Cache for Local Updates
Aggregator
ii
Network Quality
CPU Resource
Client Side
①
Decision Making
i
Currently Observed States
Client Scheduler
ii
Model Updates
Local Training
i : Send Resource Info of CPU & Network.
ii : Exchange Local Updates and Global Updates.
iii: Send Model Info to Coordinator.
iv: Push the Uploaded Local Model(s) to the Cache.
v : Rank Model(s) in the Cache.
vi: Update Statistical Information.
① Client Decide whether to Participate in FL Training according to Available Resource of Device.

Fig. 1. System Overview. The self-determining FL consists of the server and the client. Clients make their own decision on whether to participate in FL training according to their current-observed resource status $\Omega_{i,k}$.

B. System Overview

This subsection shows the design principles of the system, including the brief introduction and coordination of modules. Figure 1 shows that the proposed system consists of 2 main roles, *the parameter server (PS)* and *the client*.

- **Parameter Server (PS)** is a central station for global model updates. Especially, the PS follows the pattern of "selection-after-training" [12] by filtering out the local updates of large staleness.
- **Client** is the basic computing unit of the so-called distributed machine learning in the proposed system. A self-determining client autonomously decides whether to participate in the FL training. After local training, clients report their local updates to the PS.

Figure 1 shows the collaboration between PS and clients in the proposed self-determining FL. According to the aforementioned modules as shown in Figure 1, a client's scheduler sends its local update to PS after local training. Then, the *model transfer* module in PS received model updates and put them into the cache for aggregation.

The information included in the local update is saved into a database by the *coordinator* module. Meanwhile, the coordinator is also able to filter out some less useful local models according to the statistical information (e.g., models with large staleness). After the filtration, the *aggregation* module starts to aggregate local models, and the global update is finished for this round of training.

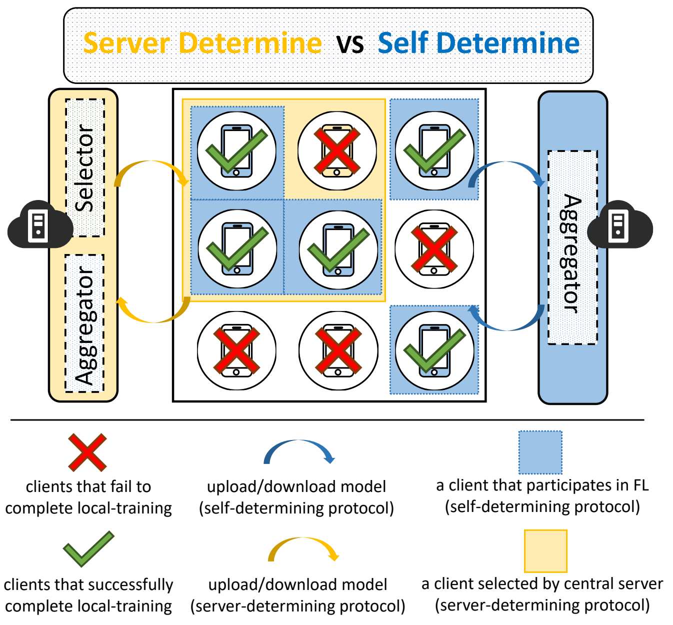

Server Determine vs Self Determine
Selector
Aggregator
Aggregator
clients that fail to complete local-training
upload/download model (self-determining protocol)
a client that participates in FL (self-determining protocol)
clients that successfully complete local-training
upload/download model (server-determining protocol)
a client selected by central server (server-determining protocol)

Fig. 2. Comparison between server-determining protocol and self-determining protocol.

```
Algorithm 1: Self-Determining FL
Input  : $y_0$ ($y_0$ is the initialized global model),
         $\{\mathcal{S}(M_k, k) \in [N]\}$ ($\mathcal{S}(M_k, k)$ is the
         candidates set for round $k \in [K]$, $M_k$ is the
         size of $\mathcal{S}(M_k, k)$ for round $k \in [K]$).
Output: The final global model $y_k$, $k \in [K]$.
1  Initialize $k \leftarrow 0$
2  while $F(w_k) - F(w^*) > \epsilon$ do
3      $k \leftarrow k + 1$
4      Sends $\boldsymbol{y}_{k-1}$ to participants $\mathcal{S}_k^M$
5      repeat
6          Asynchronous local-training based on $\boldsymbol{y}_{k-1}$
7          PS Receives $x_{i,k}$, $\tau_{i,k}$ from participant $i$ (local
            model and staleness info)
8          if $\tau_{i,k} > \tau_{max}$ then
9              Drops the update and notifies client $i$
10             continue;
11         Saves $\tau_{i,k}$ into database
12         Pushes $x_{i,k}$ into the aggregation cache
13     until deadline $T_d(k)$ is met;
14     $\boldsymbol{y}_k \leftarrow$ Updates global model by Equation (5)
15 return the final global model $\boldsymbol{y}_k, k = K$
```

The whole coordination between the PS and the clients is in the reporting phases in FL training. Though PS specifies a fixed deadline and synchronizes the global model after the deadline for each round, each client in the coordination is asynchronous. Consequently, we can classify the clients into *fast*, *normal*, *slow* (i.e., straggler), and *invalid* clients. In the proposed system, PS doesn't identify the type of clients, it only needs the statistical information in their local updates.

C. Self-determining FL

We utilize Figure 2 to illustrate the comparison between the server-determining mechanism and the proposed self-determining mechanism. The self-determining mechanism enables clients to schedule themselves according to their efficiency of training. In Figure 2, a cloud represents the parameter server. Mobile phones with a green check mark represent fast clients or normal clients that complete local training in time. Mobile phones with red cross marks represent invalid slow clients that are not able to complete local training in an FL round. Some invalid clients lost contact because of bad network connection quality.

The core protocol in our proposed system is different from the synchronous FL. At the beginning of the FL round, the PS broadcasts a one-bit start signal to all clients. Then, clients autonomously observe the current resource status and compare the status information with its historical statistics. The comparison results help clients decide whether to start a local training or not.

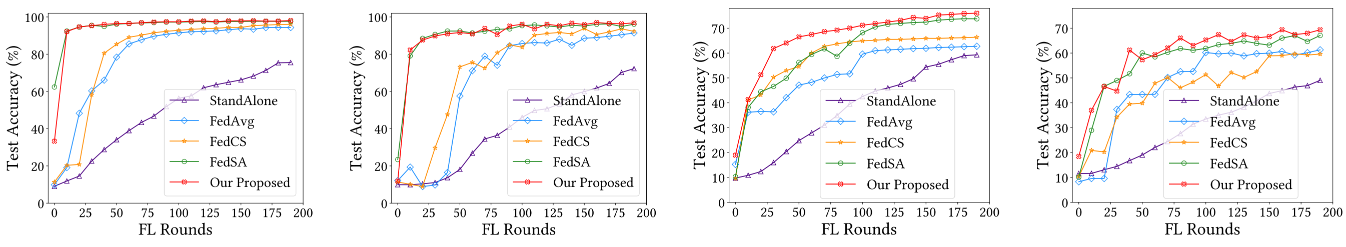
After that, some clients execute local training and upload their updates to PS while others fail. The clients that fail to complete local updates become stragglers and wait for the next *Local-Training Phase*. Some of the clients that lost contact are regarded as invalid clients. The number of rounds that a client i has postponed uploading its updates is recorded as a value of *staleness* $\tau_{i,k}, i \in [N], k \in [K]$. If some of the stragglers still fail to complete their local updates within the tolerance of staleness, they will also turn into invalid clients and restart in the next FL round. In this case, all clients hold their own "timeline" in parallel.

At the end of an FL round, the PS broadcasts a one-bit termination signal to all clients and then starts to aggregate those collected updated models. Moreover, the PS can filter out those models with poor quality according to statistical information ($\tau_{i,k}$, $\forall i \in [N], k \in [K]$) in the local updates reported by clients.

The following paragraph introduces the details of collaboration between the parameter server and self-determining clients.

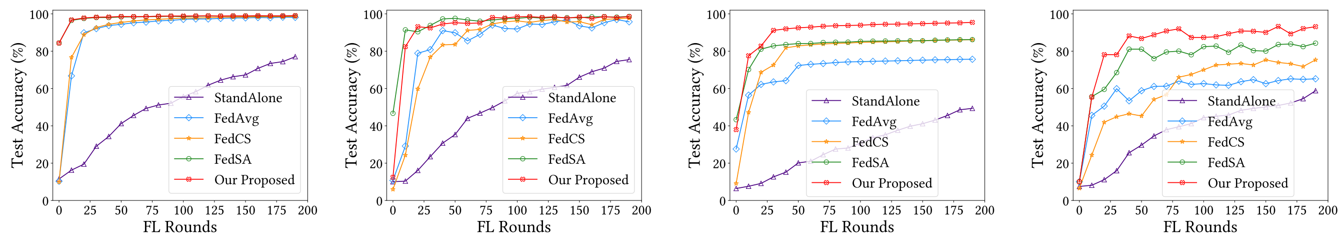
As mentioned in Section II-B, the parameter server in the proposed self-determining protocol is only an auxiliary station for model aggregation. According to Algorithm 1, the parameter server initializes and sends the global model $\boldsymbol{y_k}$ to clients who request it at the beginning of round $k \in [K]$. Then the PS repeats the FL rounds until the loss of the global model converges. Note that, the PS keeps listening to the local updates uploaded from clients until the end of *Local-Training Phase*. If the staleness $\tau_{i,k}$ of the local update is larger than the predefined threshold, the PS drops the model and notifies the relevant client. In *Global Update Phase*, the aggregator of the PS updates the global model by Equation (5).

According to Algorithm 1, clients in the proposed protocol start the *Local-Training Phase* when receiving the start signal $a_{i,k}$ ($i \in [N], k \in [K]$), from the PS. Each client i ($\in [N]$)



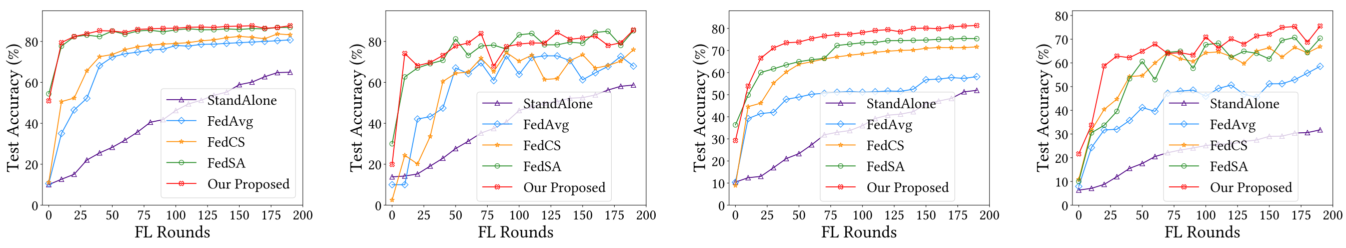
(a) Acc. of CNN (I.I.D.) (b) Acc. of CNN (Non-I.I.D.) (c) Acc. of Linear (I.I.D.) (d) Acc. of Linear (Non-I.I.D.)

Fig. 3. Accuracy of CNN/Linear Model trained based on MNIST using I.I.D or Non-I.I.D. settings.



(a) Acc. of CNN (I.I.D.) (b) Acc. of CNN (Non-I.I.D.) (c) Acc. of Linear (I.I.D.) (d) Acc. of Linear (Non-I.I.D.)

Fig. 4. Accuracy of CNN/Linear Model trained based on EMNIST using I.I.D or Non-I.I.D. settings.



(a) Acc. of CNN (I.I.D.) (b) Acc. of CNN (Non-I.I.D.) (c) Acc. of Linear (I.I.D.) (d) Acc. of Linear (Non-I.I.D.)

Fig. 5. Accuracy of CNN/Linear Model trained based on Fashion-MNIST using I.I.D or Non-I.I.D. settings.

observes the resource status $\Omega_{i,k}$ $(i \in [N], k \in [K])$ at round k. Then, the scheduler in the client makes a decision on whether to participate in the current FL training or not.

III. Performance Evaluation

A. *Experiment Settings*

1) Testbed and Evaluation: We have implemented the system as an emulator for the semi-asynchronous FL using PyTorch 1.10.0 [13]. We have established a testbed where the server is equipped with a CPU of Intel(R) Xeon(R) W-2150B, 64 GB memory, and one GPU of NVIDIA GeForce RTX 3080 Ti with 12 GB memory. Furthermore, we developed an Android APP named *Data Tracer* and recruited around 100 volunteers to collect devices' routine and resource status data.

2) Models and Datasets: For the training models, we chose the Linear Model and CNN model [3]. The Linear Model is composed of 3 linear layers with 512 units, and each layer is activated by a ReLu function. For the training datasets, we applied MNIST [14], EMNIST [15], Fashion-MNIST [16], and CIFAR-10 [17] to the FL training. And we followed the Dirichlet distribution ($\boldsymbol{C}_D = 1.0$) for the Non-I.I.D. local datasets, where D denotes the global dataset.

3) Baselines: We conducted the experiments and evaluated the performances of our proposed framework as well as 4 baselines including Stand Alone, FedAvg [3], FedCS [18] and FedSA [11].

4) Hyper Parameter: In the synchronous FL, there are $K(=200)$ of FL rounds and a total of $N(=100)$ clients for local training. Only a number $M(\leq N)$ of clients participate in local training in a single FL round. For the semi-asynchronous FL, $\tau_{max}(=3)$ denotes the tolerance of staleness of local updates from stragglers.

B. *Convergence of Aggregated Model at FL Server*

We evaluated the global accuracy of each method and the results are shown in Figure 3, Figure 4, and Figure 5. The subfigures of the aforementioned figures represent accuracies on different datasets (EMNIST, Fashion-MNIST, and MNIST). For the analysis of convergence performance, we applied the Linear Model and CNN model as the training models with three datasets including EMNIST, Fashion-MNIST, and MNIST.

We can see in these subfigures that the proposed system outperforms other baselines based on the CNN/Linear Model and different data settings. Moreover, our proposed system achieves higher accuracy while ensuring the stability of convergence in the resource-limited MEC environment. It should be mentioned that the inflection points of convergence of FedSA and the self-determining FL are significantly faster than those of other baselines with synchronous protocols.

These experimental results indicate that the number of rounds to required accuracy is lower and the convergence speed is faster. For detailed analysis, Stand Alone achieves the lowest accuracies because it doesn't update the global model by aggregating local models. FedAvg achieves not bad accuracies in Figure 3(d) compared with FedCS, but accuracies in Figure 4(d) and Figure 5(d) are not promising because of the larger and more complicated datasets.

In summary, our proposed system converges faster and achieves the highest and most stable accuracy out of all methods. In most cases, FedCS and FedSA achieve higher accuracies than FedAvg. However, FedAvg achieves almost the same average accuracy as FedCS in Figure 3(a) and Figure 4(a). This observation indicates that the improved synchronous protocols (e.g., FedCS) are only able to tackle partial cases of FL training. The accuracy gap between FedAvg and the improved synchronous protocols can be narrowed because of the combined impact of datasets, models, and system heterogeneity. Nevertheless, our proposed system achieves the best accuracy in all the above cases of settings.

C. Analysis of Utilization of Local Model

The most important advantage of self-determining protocols is the utilization of local models of all valid clients. As is mentioned above, we can observe in Figure 3, Figure 4 and Figure 5 that the self-determining system achieves a promising global test accuracy in FL training. According to the settings in the experiments, the proposed self-determining system utilizes all potentially valid clients and these clients successfully upload the local updates to the server. Constitutionally, compared with baselines, the self-determining FL gets more training for the global model per unit of time.

IV. Conclusion

We propose a self-determining framework, aiming to improve the global test accuracy of non-synchronous FL. In our proposed framework, clients determine themselves so that the FL system can tackle the frequent changes of resources in MEC. Compared with other state-of-the-art baselines, our proposed framework achieves higher accuracy. In summary, our proposed system is a promising approach for a large-scale FL in MEC.

V. Acknowledgments

This work was supported in part by Key-Area Research and Development Program of Guangdong Province (Grant no. 2021B0101400005).

References

[1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021. [Online]. Available: http://dx.doi.org/10.1561/2200000083

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

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS'17)*, 2017.

[4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba *et al.*, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 374–388. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2019/file/bd686fd640be98efaae0091fa301e613-Paper.pdf

[5] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE Press, 2016, p. 1–6. [Online]. Available: https://doi.org/10.1109/PIMRC.2016.7794955

[6] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, 2019.

[7] H. Huang, R. Li, J. Liu, S. Zhou, K. Lin, and Z. Zheng, "Contextfl: Context-aware federated learning by estimating the training and reporting phases of mobile clients," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022, pp. 570–580.

[8] A. Xu and H. Huang, "Coordinating momenta for cross-silo federated learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, pp. 8735–8743, Jun. 2022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20853

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin *et al.*, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/6aca97005c68f1206823815f66102863-Paper.pdf

[10] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 15–24.

[11] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "Fedsa: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3654–3672, 2021.

[12] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "Safa: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, 2021.

[13] A. Paszke, S. Gross, F. Massa *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[15] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2921–2926.

[16] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[17] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, May 2012.

[18] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE International Conference on Communications*, 2019, pp. 1–7.