Federated Learning-Based Client Scheduling for Low-Latency Wireless Communications

Wenchao Xia[£], Wanli Wen[†], Kai-Kit Wong[‡], Tony Q. S. Quek[†], Jun Zhang[£], and Hongbo Zhu[£]
[£]Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications, Nanjing 210003, China, e-mail: {xiawenchao, zhangjun, zhuhb}@njupt.edu.cn
[†]Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore 487372, e-mail: {wanli_wen, tonyquek}@sutd.edu.sg
[‡]Department of Electronic and Electrical Engineering, University College London, London WC1E 7JE, U.K., e-mail: kai-kit.wong@ucl.ac.uk

***Abstract*—Motivated by the ever-increasing demands for massive data processing and intelligent data analysis at the network edge, federated learning (FL), a distributed architecture for machine learning, has been introduced to enhance edge intelligence without compromising data privacy. Nonetheless, due to the large number of edge devices (referred to as clients in FL) with only limited wireless resources, client scheduling, which chooses only a subset of devices to participate in each round of FL, becomes a more feasible option. Unfortunately, the training latency can be intolerable in the iterative process of FL. To tackle the challenge, this article introduces update-importance based client scheduling schemes to reduce the required number of rounds. Then latency-based client scheduling schemes are proposed to shorten the time interval for each round. We consider the scenario where no prior information regarding the channel state and the resource usage of the devices is available, and propose a scheme based on the multi-armed bandit theory to strike a balance between exploration and exploitation. Finally, we propose a latency-based technique that exploits update importance to reduce the training time. Computer simulation results are presented to evaluate the convergence rate with respect to the rounds and wall-clock time consumption.**

I. Introduction

Driven by the surge of mobile devices and the unprecedent explosion of data, a new computation paradigm, widely known as edge computing, has emerged [1]. By processing data via an access point (AP) at the network edge, edge computing avoids the long propagation latency required by sending data to a remote cloud. Conventionally, centralized machine learning (ML) tools are brought in for action at the edge AP to analyze a large amount of data and extract useful information for different tasks, such as classification, prediction, and detection. Nevertheless, limited by communication resources and storage capacities, as well as other issues such as data privacy and security concerns, sending user data to the edge AP is often considered impractical. Proposed by Google [2], the concept of federated learning (FL) has recently attracted much attention. The main idea of FL is to train ML models based on datasets distributed across multiple devices and send the model updates to an FL server (e.g., the AP) for aggregation.

As shown in Fig. 1, training in FL for wireless networks typically involves an iterative process and each iteration (also called a communication round) includes four main steps:

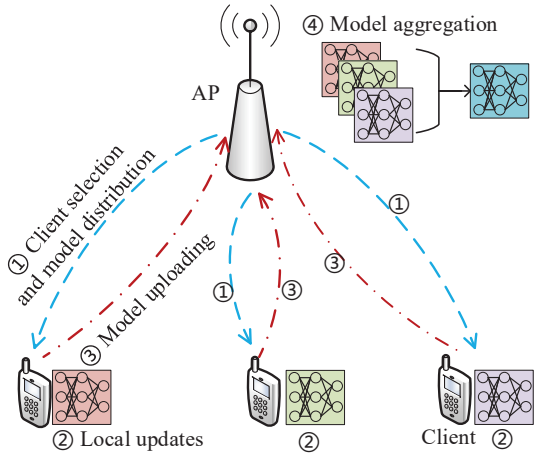


Fig. 1. An FL system illustrating the four main steps in one communication round of the training process.

① An edge AP chooses a subset of clients and distribute the model parameters (e.g., gradients and model weights) to the participating clients;
② Each participant performs local updates by training the downloaded model with its own dataset;
③ Each participant uploads its updated model parameters to an aggregator (i.e., the edge AP), and
④ The aggregator aggregates the model parameters, e.g., by weighted averaging. The aggregated results are then sent back to the clients for the next round of FL iteration.

According to the above process, only the model parameters are uploaded to the AP instead of the raw data, and therefore the FL implementation alleviates the requirement for network bandwidth and also enhances data privacy.

With the rapid development of Internet of Things (IoT), smart cities and many emerging applications, more and more devices (e.g., sensors, cameras, and smart phones) are involved in wireless networks and these devices will be the clients in wireless FL. The challenge is that an edge AP may not have enough channels for all the clients to download/upload their model updates simultaneously due to the limited spectrum

resource [3]. To overcome this, client scheduling, which selects a subset of the clients for local updates in each round of training, can be very effective. Client scheduling can also reduce the communication cost caused by the high-dimensionality of model parameters as well as network congestion. An efficient client scheduling scheme should take into account not only the learning performance (e.g., test accuracy and loss) but the wall-clock time consumption of the training process, achieving low latency needed for 5G and Beyond 5G networks.

To come up with a high-performance model with a low-latency training process, client scheduling should address the following challenges. The first challenge is the heterogeneity of clients, which is also referred to as systems heterogeneity [4]. Specifically, the computational and communication capabilities of different clients can be very different because of the diversity of hardware (e.g., CPU and memory), network connectivity (e.g., 3G, 4G, and 5G), and battery level [4]. The systems heterogeneity causes different completion times among the clients and fast clients have to wait for the slow clients, referred to as stragglers, before model aggregation.

The second challenge is the statistical heterogeneity. The non-independent and non-identical distributions (non-i.i.d.) and unbalanced prosperities of distributed data lead to drift of local updates and result in instability and slow convergence.

Additionally, dynamic wireless environments cannot guarantee communication quality and time-varying activity characteristics of the clients fails to provide stable computing power. Such environmental dynamics is highly unpredictable. Note also that the available computing resource on each client varies over time because a client can execute other processing tasks (e.g., playing games) while performing the local updates.

Moreover, obtaining channel state information (CSI) of the wireless channel and knowledge of the computing resource usage for each client causes considerable overhead and may even be impractical, especially when the number of the clients is very large. These challenges can cause a slow and unstable convergence process if not properly addressed.

In the literature, most research focused on having a faster convergence rate for client scheduling by reducing the number of communication rounds to achieve a certain level of test accuracy. However, this does not necessarily reduce the wall-clock time consumption of the whole training process, since the time interval per round is also a key factor. Note that the time interval per round in synchronous FL training depends on the time consumed by the stragglers since the AP cannot perform model aggregation until it receives all the local updates from the chosen clients. Recognizing this fact, this article investigates client scheduling with an emphasis on the wall-clock time consumption of the whole training process, which includes the latency incurred by both wireless transmission and local computation on the clients. Also, different from the existing works assuming the availability of prior information regarding the CSI and computing resource usage [3, 5, 6], we consider a more practical scenario without the prior information, and utilize the multi-armed bandit (MAB) tool to estimate the statistical information online. Our overarching aim is to achieve a fast convergence speed with respect to the wall-clock time by performing client scheduling in (1) reducing the required number of communication rounds and also (2) shortening the average time interval for each round.

The remainder of this article is organized as follows. We begin by introducing two client scheduling schemes based on update importance that aim to reduce the required number of communication rounds. Then we analyze latency-based client scheduling schemes in both cases with and without prior information. Afterwards, we propose a client scheduling scheme which jointly considers update importance and latency in order to reduce the required number of communication rounds and the average time interval per round simultaneously. Finally, simulation results are presented before we conclude the article.

II. Update Importance Based Scheduling

In this article, synchronous update is considered for FL in wireless networks. To reduce the number of communication rounds required by a certain level of test accuracy, we need to choose the clients whose local updates can help accelerate the convergence and reduce the required rounds. The local updates of such clients, we think, are more important. To describe the update importance, we introduce two metrics, i.e., update staleness and update drift. The former aims to quantify the staleness of the local updates whereas the latter can be regarded as a direct response to the heterogeneity of distributed datasets of the clients.

A. Update Staleness

Stale updates have an undesirable impact on the learning performance of wireless FL. The staleness of local updates slows down convergence and can even make the training process diverge [7]. Motivated by the notion of age-of-information [8], age-of-update (AoU) was proposed to measure the staleness associated with the local updates of each client [9].

The update rule is well known to be: $a_k(t+1) = (a_k(t)+1)(1-s_k(t))$, where $a_k(t)$ is the age of the local update of client k in round t and $s_k(t) \in \{0,1\}$ is an indicator, which equals 1 if client k receives the aggregated model parameters in round t and 0 otherwise. The main idea behind the update staleness based client scheduling schemes is to keep the local updates as fresh as possible, by minimizing a certain function of the AoUs in each round, e.g., $\sum_k \log_2(1+a_k)$. Generally, a client with a larger AoU has priority to be selected to participate in the current communication round if its wireless channel is available.

Fig. 2 presents an example of AoU based client scheduling where at most two clients are chosen from four clients in each round. A client cannot be selected and its AoU increases by one if its channel gain is lower than a threshold, as shown in Fig. 2(a). This is because poor connection will fail to provide stable transmission. Figs. 2(b) and 2(c) illustrate the utility of AoUs and the selected clients in each round, respectively.

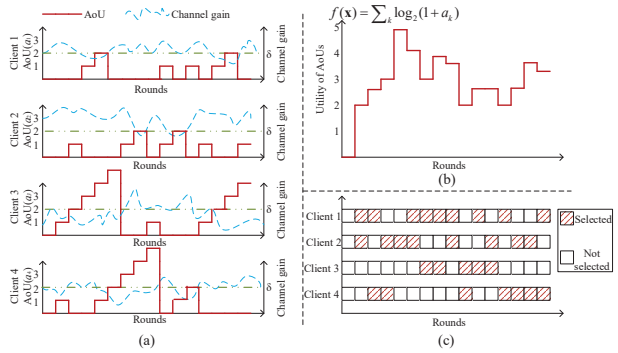


Fig. 2. An example of AoU based client scheduling where at most two clients are chosen from four clients in each round. (a) The AoUs of different clients, (b) the utility of AoUs, and (c) the scheduling results in each round.

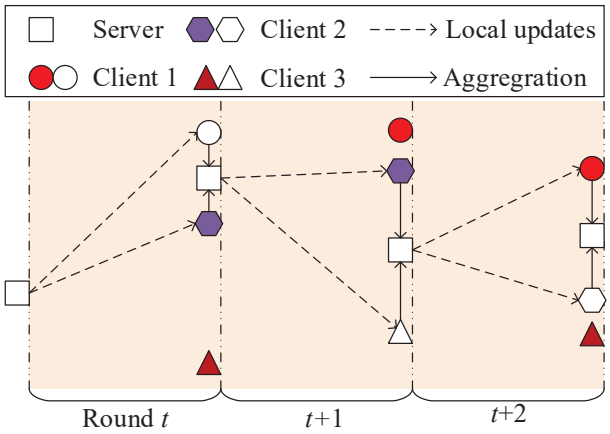


Fig. 3. An example of weight divergence based client scheduling where two clients are chosen from three clients in each round. Different shapes represent the weights of different clients and the solid ones suggest that the corresponding clients are selected for local updates in the next round.

B. Update Drift

The local data is usually non-i.i.d. and each client usually has a unique distribution of data. The skewness of the local data distribution from the population distribution causes the drift of the local updates and degrades the training performance of FL [10]. To quantitatively characterize the update drift, we introduce a new concept *weight divergence*, in terms of the Manhattan distance or the Euclidean distance of the weights of a local model and the global model [11]. Generally, the weight divergence increases as the distributed data becomes more non-i.i.d. For a client, a larger weight divergence suggests that the client should participate in more rounds to reduce the weight divergence. In addition to the weight divergence, similar concepts such as dissimilarity [4], parameter divergence, and gradient diversity [5] can be used to quantify the update drift of distributed datasets of the clients.

Fig. 3 shows an example of client scheduling based on the weight divergence. In each round, the weight divergence of each client is computed after the local updates are uploaded. Based on the report of the weight divergence, the AP uses a certain scheduling policy to make a decision about which clients are selected and informs the selected clients to upload their weights. In this example, two clients are chosen in each round. It is worth pointing out that the update drift based client scheduling schemes can reduce the required communication rounds, but they do not necessarily decrease the time consumption of the whole training process, because they ignore the time consumed by a client to finish each communication round.

III. Latency Based Client Scheduling

In wireless FL, the clients are typically large in number and have unstable network connections due to dynamic wireless environments. In addition, ML models especially deep neural network models often have hundreds of millions of parameters. Insisting on directly training such huge models can result in the systematic exclusion of clients with restricted bandwidth or limited network access, as well as unacceptable delay. Motivated by these facts, it is important to take into account the time consumption for each round while performing client scheduling, rather than just the number of the required communication rounds. The latency includes the communication and computation delay and the communication delay includes the model distribution and model upload time. Here, it is assumed that the delay incurred by model aggregation is negligible because the AP has relatively stronger computation capability compared to the clients and the computational complexity of model aggregation is considered low. In the following, we will analyze two scenarios: one with prior information and the other without prior information available.

A. Prior Information Available

Since the wireless communication environment as well as the activity characteristics of the clients is dynamic, prior information such as the CSI between the clients and AP and the computing resource usage of each client would be very useful [6]. If such information is collected, then the AP can choose a subset of the clients with low latency to participate in the wireless FL training in each round, thereby shortening the average time interval of the training rounds. However, this approach focuses only on communication latency and ignores the effect of distributed data of the clients on the convergence rate and therefore may increase the number of communication rounds. To deal with this challenge, we propose a latency based client scheduling scheme with a fairness constraint.

The proposed latency based client scheduling scheme with a fairness constraint is a mixed timescale method which has a short-term optimization objective in each round and a long-term fairness constraint across all communication rounds. In particular, the proposed scheme aims to minimize the time interval in each round, i.e., $\min_{\mathcal{S}(t)} \max_{k\in\mathcal{S}(t)} \tau_k(t)$, where $\mathcal{S}(t)$ denotes the subset of the clients chosen in round t and $\tau_k(t)$ denotes the time consumed by client k in round t, which is the sum of the three main components: the model distribution time, the local update time, and the model upload time. Note that the time consumption of each round is determined by the slowest client because the AP cannot

perform model aggregation until all the selected clients finish uploading data. Meanwhile, a long-term fairness constraint $\liminf \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[s_k(t)] \geq c_k$ is introduced to "tell" each client that how many communication rounds they should be involved in, where $\mathbb{E}$ denotes the expectation operator, T is the number of communication rounds, and the fairness factor $c_k \in [0, 1)$ denotes the minimum fraction of communication rounds that client k should participate in. Note that the virtual queue technique can be adopted to meet the fairness constraint [12].

The fairness constraint enables a fair participation among the clients and avoids slow clients but with important training data being neglected due to the pursuit of low delay. Furthermore, the fairness constraint can let the clients with important training data participate in more communication rounds by setting a larger fairness factor. Note that the fairness factors c_k's should be carefully designed as they play an important role in improving the learning performance, but how to design the fairness factors is still an open problem. One heuristic solution is to choose them proportional to the dataset size, i.e., a client with a larger dataset will be set with a larger fairness factor.

B. Prior Information Unavailable

Obtaining prior information, such as the CSI and the computing resource usage of the clients, often comes with a high cost and may even be impossible if the number of clients is too massive. Without the prior knowledge, the AP cannot know which clients have a shorter delay in advance and hence would be unable to make proper decisions about client scheduling. To deal with, instead of estimating the instantaneous information in each round as the prior information, we attempt to learn the related statistical information online during the training process, to aid decision making. The MAB theory is useful in shedding light on the scheduling problems [12].

MABs are a special class of reinforcement learning problems that handle decision making in uncertain environments. Specifically, in the model, a player is in charge of making a decision of which arms of the bandits are to be pulled in each round. Each arm pull is regarded as an action. As a result of an action, a reward is observed. The MAB problems aim to maximize the cumulative reward obtained in a sequence of actions. However, due to the lack of prior information, the player has to learn some statistical information online while performing the actions, to help make decisions in the future. Therefore, the player should balance between the exploitation of actions that did well in the past and the exploration of actions that might return higher rewards in the future [13]. Some classic algorithms, such as ϵ-greedy, upper confidence bound (UCB), etc, can collect and use statistical information at the same time to strike a balance between exploration and exploitation. As shown in Fig. 4, the AP and clients in wireless FL can be regarded as the player and arms, respectively. An action is to choose a subset of the arms (referred to as a super arm) to pull. Furthermore, the original objective function, i.e., minimizing the latency of the whole training process, can be recast as the maximization of the latency reduction. Here, the latency reduction is viewed as the reward of an action.

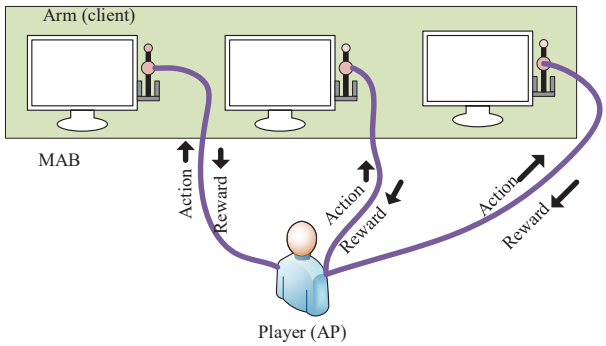


Fig. 4. An example of MAB. A player (i.e., the AP) is responsible for making a decision about which arms (i.e., the clients) are chosen to be pulled in each round. A reward is observed at the AP after performing an action.

For easy of illustration, we take an orthogonal frequency-division multiple-access (OFDMA) system with N wireless channels as an example, in which a channel is allocated to at most one client such that at most N clients are selected in each communication round and there is no interference among the clients. The model distribution time as well as the model upload time, relying on the model size and the wireless channel between the AP and clients, is i.i.d. over time according to a certain distribution and its expectation is unknown in advance. Similarly, the time for local updates is assumed to be i.i.d. over time according to a certain distribution whose expectation is also unknown a priori. Instead of estimating the three main components independently, i.e., the model distribution time, the local update time, and the model upload time, we collect the current average value $\bar{\tau}_k(t)$ of the consumed time $\tau_k(t)$ of client k. A smaller value of $\bar{\tau}_k(t)$ suggests that client k have a larger reward and be chosen with a larger probability. The same arm can be observed in different actions and we collect and exploit information about the reward of one certain arm from the operations of different actions, in order to make better decisions in the future. Then, with the observed reward, the classic MAB algorithms (e.g., ϵ-greedy and UCB algorithms) can be adopted. Note that for more complex scenarios, such as with inter-user interference and/or mobility, the algorithms of contextual and/or adversarial MAB problems can be exploited.

IV. Update Importance and Latency Based Scheduling

According to the above analysis, it is found that both the update importance based or latency based client scheduling metrics do not necessarily decrease the latency of the whole training process because the former ignores the time interval of each round and the the latter does not take into account how to reduce the required number of rounds. To address this, we propose a client scheduling scheme which aims to reduce the time consumption per round and also the required number of communication rounds simultaneously by jointly considering the update importance and latency issues of the clients.

The proposed scheme integrates with the MAB theory and learns to schedule clients online without prior information. On one hand, we use the weight divergence to quantify the effect of the clients on the convergence rate. On the other hand, the MAB tool is used to estimate the average value $\bar{\tau}_k(t)$ of the delay $\tau_k(t)$ of client k in round t. Then the scheduling indicator of each client is set as a weighted sum of the weight divergence and the learned average time cost per round. The clients with a larger indicator have higher priority to be chosen as the participants in the next round of FL training.

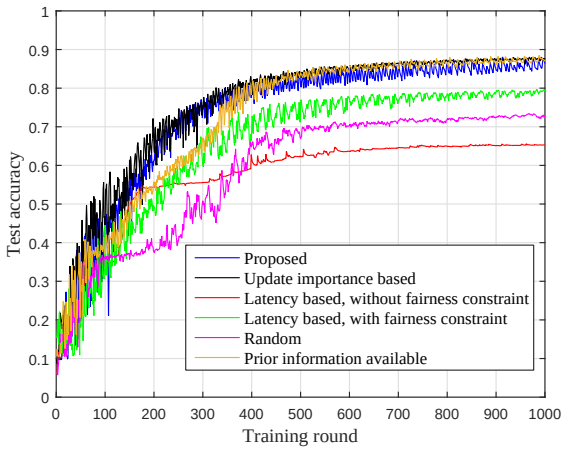
V. Performance Evaluation

To validate the performance of the proposed scheme, we provide some numerical results for wireless FL at the network edge which considers one AP and 20 clients. These clients are assumed to be randomly distributed within the coverage of the AP. OFDMA is adopted and there are five sub-channels, meaning that at most five clients are chosen in each round.

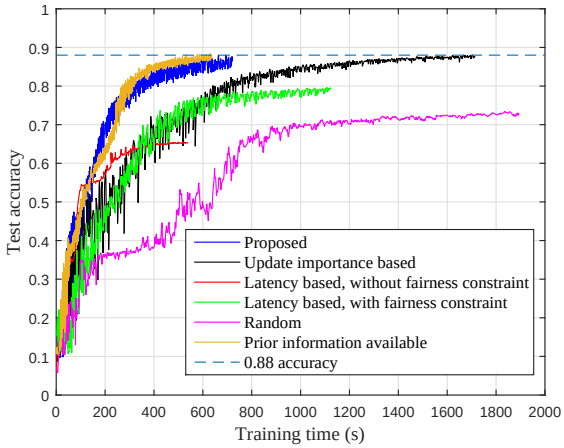
We consider an FL task of classifying handwritten digits using a popular dataset, i.e., the MNIST. We assume that the data of the clients is non-i.i.d. and each client is randomly assigned training samples with only two types of digits. A standard multilayer perceptron model is employed for training. Besides, we introduce five baseline schemes for comparison. The first one is based on the update importance only in which the AP selects the clients according to the weight divergence. The second one is based on the latency-only of each client where the AP selects the fast clients to minimize the time interval in each round. The third one is the second one with an additional fairness constraint. A heuristic solution to setting the fairness factors is adopted, where all the fairness factors are set with the same constant. In addition, a randomized scheme is introduced where the AP selects a subset of the clients randomly. Finally, the proposed scheme with prior information available is also introduced for comparison.

According to the results in Fig. 5(a), the test accuracy, as well as the convergence rate with respect to the number of the communication rounds, of the proposed client scheduling scheme is close to that of the update importance based scheme. However, the latency based client scheduling schemes and the random scheme have lower test accuracy. This is because the update importance based scheme and the proposed scheme take into account the update importance, whereas the other schemes do not. Moreover, we also find that the fairness constraint can help improve the test accuracy of the latency based scheme, as the fairness constraint enables a fair participation of the clients in the training process. It is reasonable to infer that a more carefully designed solution to the fairness factors can bring more improvement of the test accuracy compared to the scheme solely based on the latency issue.

Fig. 5(b) presents the test accuracy versus the wall-clock time consumption. It is easy to find that without prior information, the proposed scheme consumes the least time to achieve the given test accuracy (i.e., 0.88 accuracy). More specifically, the proposed scheme can save more than 50% of time compared to the update importance based scheme. Similar



(a)



(b)

Fig. 5. Comparison of the proposed scheme and the baselines: test accuracy versus (a) the communication rounds and (b) the wall-clock time consumption.

to the observation in Fig. 5(a), the proposed scheme achieves a better test accuracy than the latency based schemes and the randomized scheme. The training latency can be further reduced when prior information is available, but with the high overhead of achieving prior information.

VI. Conclusions and Open Issues

This article studied the issues associated with client scheduling to obtain low-latency in the context of FL for wireless networks, in terms of wall-clock time consumption. To achieve fast convergence for a given service requirement, we performed client scheduling according to two features jointly: (1) reducing the required number of communication rounds and (2) shortening the average time interval per round. The former was realized by performing client scheduling based on update importance and two metrics, i.e., update staleness and update drift. The latter was achieved by performing latency based client scheduling. Both scenarios with and without prior information were considered. Particularly, the MAB theory was introduced to learn the statistical information online in

the scenario without the prior information available. Furthermore, we proposed a scheme which jointly considered update importance and latency. Finally, the numerical results validated the performance of the proposed schemes.

Research on wireless FL is still in its very early stage and there remain open research challenges that deserve future investigation. For example, the first challenge is the availability issue of clients. A client may be unavailable temporarily due to poor channel conditions or exhaustion of computing power. The AP should check carefully the availability of the clients before performing client scheduling, which causes extra overhead. The second challenge is the incentive mechanism. Clients are generally reluctant to take part in the training process due to limited communication and computation resources. How to encourage clients to participate in the training process is a major concern. A possible solution is to introduce financial rewards but this is yet to be explored more closely. Finally, security issue cannot be ignored. Adversarial clients can manipulate/interfere the training result by embedding carefully designed samples into the training dataset (i.e., data-poisoning attacks) or by sending malicious gradient updates (i.e., model-poisoning attacks). These attacks entail defense mechanisms to prevent severe performance loss.

REFERENCES

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, 2018.

[2] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[3] 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," *arXiv preprint arXiv:1909.07972*, 2019.

[4] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[5] J. Ren, Y. He, D. Wen, G. Yu, K. Huang, and D. Guo, "Scheduling in cellular federated edge learning with importance and channel awareness," *arXiv preprint arXiv:2004.00490*, 2020.

[6] Q. Zeng, Y. Du, K. K. Leung, and K. Huang, "Energy-efficient radio resource allocation for federated edge learning," *arXiv preprint arXiv:1907.06040*, 2019.

[7] W. Dai, Y. Zhou, N. Dong, H. Zhang, and E. P. Xing, "Toward understanding the impact of staleness in distributed machine learning," *arXiv preprint arXiv:1810.03264*, 2018.

[8] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE Int. Conf. Computer Commun. (INFOCOM)*, 2012, pp. 2731–2735.

[9] H. H. Yang, A. Arafa, T. Q. Quek, and H. V. Poor, "Age-based scheduling policy for federated learning in mobile edge networks," *arXiv preprint arXiv:1910.14648*, 2019.

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

[11] M. Kamp, L. Adilova, J. Sicking, F. Hüger, P. Schlicht, T. Wirtz, and S. Wrobel, "Efficient decentralized deep learning by dynamic model averaging," in *Proc. Machine Learning Knowledge Discovery Databases*, Cham, 2019, pp. 393–409.

[12] 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.*, pp. 1–1, 2020.

[13] S. Bubeck, N. Cesa-Bianchi *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.