FedTCS: Federated Learning with Time-based Client Selection to Optimize Edge Resources

Saira Bano[1,2], Nicola Tonellotto[1], Pietro Cassarà[2,3] and Alberto Gotta[2,3]

[1]Department of Information Engineering, University of Pisa, Pisa, Italy
[2]Information Science and Technology Institute "A. Faedo", National Research Council, Pisa, Italy
[3]CNIT - National Inter-University Consortium for Telecommunications, Parma, Italy

Abstract

Client sampling in federated learning (FL) is a significant problem, especially in massive cross-device scenarios where communication with all devices is not possible. In this work, we study the client selection problem using a time-based back-off system in federated learning for a MEC-based network infrastructure. In the FL paradigm, where a group of nodes can jointly train a machine learning model with the help of a central server, client selection is expected to have a significant impact in FL applications deployed in future 6G networks, given the increasing number of connected devices. Our timer settings are based on an exponential distribution to obtain an expected number of clients for the FL process. Empirical results show that our technique is scalable and robust for a large number of clients and keeps data queues stable at the edge.

Keywords

Federated Learning, Clients Selection, Mobile Edge Computing (MEC) framework

1. Introduction

Recently, mobile telecommunications is facing a digital transformation triggered by the more frequent use of artificial intelligence (AI) based services and massive connection requests from a plethora of smart devices, autonomous vehicles and industrial IoT-based systems. In this scenario, a report published by CISCO [1] estimates that the number of IoT devices will reach to 20 billion by 2023. The massive communication of these devices are generating ever-increasing distributed traffic as they are integrated into many AI-based application scenarios such as computer vision [2] or autonomous vehicles [3]. The growth of these AI applications continues to drive the development of wireless networks, and 6G is expected to bring the evolution of mobile devices from *"connected things"* to *"connected intelligence"* [4]. One of the fundamental goals of 6G is to create a holistic system in which communication, computation and control are jointly orchestrated to achieve unprecedented levels of reliability, energy efficiency and sustainability [5]. In this 6G ecosystem, machine learning (ML) and AI play a critical role particularly through their deployment at the edge of wireless networks, close to end users,

AI6G'22: First International Workshop on Artificial Intelligence in beyond 5G and 6G Wireless Networks, July 21, 2022, Padua, Italy

saira.bano@phd.unipi.it (S. Bano); nicola.tonellotto@unipi.it (N. Tonellotto); pietro.cassara@isti.cnr.it (P. Cassarà); alberto.gotta@isti.cnr.it (A. Gotta)
0000-0002-3704-4133 (P. Cassarà); 0000-0002-8134-7844 (A. Gotta)

and by using the user data to train AI models. However, the challenges of trustworthiness and scalability, especially user's privacy and security, are one of the most important requirements for 6G smart services and applications, for which the General Data Protection Regulation (GDPR) guidelines must be met and the direct transmission or collection of user data is prohibited [6].

To address the problem of privacy, Google has proposed Federated Learning (FL) [7], an approach to address distributed learning problems where a shared global model can be learned using locally generated models from remote clients without sharing private data. This approach has two advantages: first, it decouples the need for a shared data repository for learning, and second, it preserves user privacy. In the process of FL, clients willing to participate in the learning process collaboratively execute training tasks together, which are orchestrated by edge or cloud entities. In the FL, each participating client must download the current global shared model, improve it through training with its local data, and summarise the changes in the form of a light- focused update in the form of weights or gradients. These client-generated updates are sent to the entity which is in charge of aggregating these model updates into the global model. The updated model is then sent back to the clients, that replace their local model with the updated model.

In this paper, we focus on the implementation of an FL-based protocol suitable for network infrastructure relying on Mobile Edge Computing (MEC). In particular, we address the problem of client selection in the FL process. In the process of FL, even if an increasing number of clients can generate a more accurate shared global model, the amount of data generated to update the global model by these clients can lead to overflows and instability of the data queues at the edge of the MEC-based network infrastructure. One of the solution to prevent this overflow and to keep the data queues stable is to select a subset of clients to send their models to the edge. There are many techniques that are proposed in the literature for the client selection, such as in [8], in which the authors try to overcome the challenge of selecting clients with heterogeneous resources. In the proposed system, random clients receive the MEC operator's request to participate in the FL, then these clients inform the operator about their resources. Finally, the MEC operator selects only those clients that can complete the tasks within a certain time frame. In [9], the authors proposed the biased client selection technique and showed that selecting clients that have higher losses of their learning model can improve the speed of error convergence, which significantly reduces the communication overhead. However, the above techniques are difficult to implement in many application scenarios, especially in the case of mobile nodes.

In this paper, we propose a new protocol called FedTCS, i.e., Federated Learning with Time-based Client Selection. In this protocol, the control agent at the edge of the MEC-based infrastructure selects the clients involved in the FL process to enable more efficient updating of the shared global model. We assume that the buffers for storing the model updates generated by the clients are placed at the edge. We propose a time-based scheme protocol for selecting clients involved in the FL process to avoid buffer overflows in the storage at the edge caused by excessive data flow due to the increasing number of clients. We use Kafka[1], a distributed event streaming platform used to process data streams. Precisely, clients store their local updates in the Kafka broker, from where the entity in charge of averaging the updates can read these

[1] https://kafka.apache.org/

updates to generate the global model. After averaging, the global model is again stored in the broker so that clients can download the global model once it is available.

The main contributions in this paper are summarised below:

- We propose a probabilistic time-based client selection scheme for the FL procedure based on exponentially distributed timers. Through simulation-based analysis, we show that our scheme allows to maintain queue stability even as the number of clients increases, thus avoiding the implosion of model updates at the edge.
- We propose a client selection scheme to deal with stragglers of edge devices, by using an equal selection probability during the FL procedure.
- We provide a preliminary performance analysis to prove the scalability of the proposed selection scheme, while keeping the expected number of clients optimal according to the size of the queue.

The remainder of this article is organized as follows: in Section 2, we present our proposed system for the client selection based on exponentially distributed timers; the experimental setup and numerical results for FL are discussed in Section 3. Finally, in Section 4 we provide the conclusions and future developments.

2. Time-based Clients Selection

Federated learning can involve communications between numerous clients and the model averaging entity, resulting in excessive latency and network congestion. We propose a client selection approach that reduces congestion and latency while keeping buffer queues stable at the edge. Our approach allows computing the average number of transmitting clients to maximise the time-averaged accuracy of the federated learning procedure.

In this work, a two-tier distributed network infrastructure based on the MEC architecture is used to make an optimal selection from a large number of clients for FL, as shown in Figure 1. In this distributed architecture, the server FL is located in the cloud, while a control agent is placed at the edge for efficient client selection. Since there may be many edge systems serving different areas, we have a control agent (CA) on each of these systems that are closer to the users. This control agent uses a system based on ACK (short message in the form of an acknowledgment) to send a short message to all clients and the server. This ACK generated by CA is triggered by the arrival of the first message from any client in the given round of FL, which is explained in more detail later in the algorithm. This edge-based system also makes use of the Kafka broker, which acts as a buffer to allow asynchronous reception of model updates from clients. It also controls the flow of updates to and from clients and makes them available to the aggregation server in the cloud. The goal of this work is to select a subset of clients such that the flow of information in the form of client updates toward the edge is reduced and does not saturate the Kafka buffer at the edge. For this purpose, in this work we propose a timer-based algorithm, in which the optimal number of clients is selected according to the size of the queue and the whole algorithm works as follows:

1. We assume that all clients train their model on locally available data and all the clients are available for all rounds of FL. Before the FL task starts, the server broadcasts a

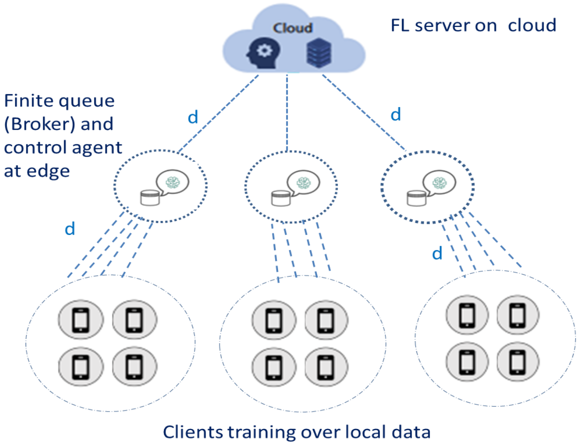

Cloud
FL server on cloud
Finite queue (Broker) and control agent at edge
d
d
d
d
Clients training over local data

Figure 1: Two-Tier Distributed Architecture for Clients Selection, where $d_i = d$ i.e. the delay of all clients from the edge

configuration message in the form of a tuple (R, μ, T) and sends the initial model parameters to all the K clients, where R is the incremental FL round number, T is the time interval, and μ is the exponential distribution parameter.

2. Each client *i* receives the tuple from the server after a delay of $2d_i$. In this work, due to space constraints, we assume that all clients have heterogeneous training times but homogeneous delays $d_i = d$ from the edge and each edge has the same delay d from the server.
3. After receiving the request from the server, each client *i* extracts a random exponentially distributed timer t_i also called a backoff timer, based on the received parameter, i.e., the interval size, and delays its training until the timer expires. Each backoff timer value is between $[0, T]$ according to a truncated exponential distribution.
4. The client with the least backoff timer and training time sends its model parameter to the edge. Upon receiving the model parameter, the CA at the edge as described above is triggered and sends an ACK message to both the server and to all the clients connected to the edge via a *control topic* managed by Kafka.
5. When clients send their model updates after their backoff timer and training have finished, some control parameters are also appended to the message header, namely the round number, the training time, and their timer value (R, Tr_i, t_i). These parameters are used by the server to adjust the T for the next round and to check whether a received message belongs to the current round or not.
6. Any client *i* that receives the ACK before it has finished its training or when the timer has not yet expired will suppress training to save computational resources. This is due to the fact that the sum of the client i's training time Tr_i and its timer t_i is larger than the sum of *minimum* training time, timer, and the round-trip delay $2d$ for receiving an ACK:

$$Tr_i + t_i > Tr_{min} + t_{min} + 2d.$$

7. After receiving an ACK, the server reads the messages from the broker at the edge and

calculates the average number of clients based on the timer values it receives from each client for a given round of FL

8. After evaluating the expected number of clients $E(X)$, it performs global aggregation based on the FedProx algorithm [10], which achieves higher accuracy compared to FedAvg for heterogeneous networks. This updated model is then distributed to all the K clients.
9. The server also sends T and μ for the next round along with the updated model, based on the average number of clients it wants to receive and the maximum latency it can experience in a round.
10. All the above steps form a complete round of FL. This procedure continues until desired accuracy of the global model is reached.

This mechanism of sending the model updates based on probabilistic based timers requires a very low computational complexity at every IoT edge device.

2.1. Exponential Distributed Time-based Client Selection

Suppose that for a random variable X with a probability density function (PDF), a mean (1/μ) is given, then the PDF of the random variable T with the truncated exponential distribution on the right-hand side of T is given as follows:

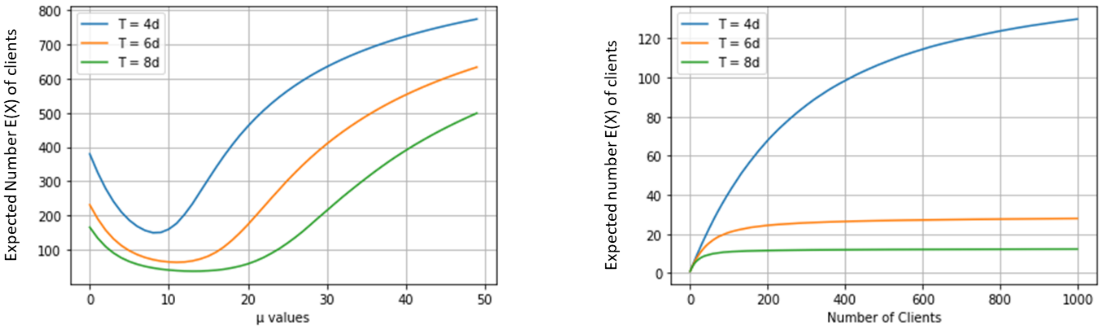
$$f(t_i;\mu,T) = \begin{cases} \frac{\mu}{T}\frac{1}{e^{\mu}-1}e^{\frac{\mu t_i}{T}} & \text{if } 0 \le t_i \le T \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where T is known and sent by the server. The exponential distribution has often been used to estimate the mean of the desired population belonging to a particular group. The densities of truncated distributions are significant in modelling such populations[11]. In order to find the values of timer t_i for each client i, we used the inverse cumulative distributive function (CDF) sampling theorem and calculated t_i from the following expression:

$$t_i(u;\mu,T) = \frac{T}{\mu}\log\big(u(e^{\mu}-1)+1\big) \tag{2}$$

where u belongs to the uniform distribution $\in [0,1]$. The FL server adjusts these μ and T values based on the number of clients expected for a given number of rounds and the desired accuracy, while keeping the queue stable. However, as we increase the values of μ, the weight of density shifts towards T, resulting in a dense timer setting [12]. In order to find the $E(X)$ for K = 1000 clients, we change the timer and μ values as shown in Figure 2a.

The results in Figure 2a show the expected number of clients for T equal to $4d$, $6d$, and $8d$, where d is the delay between edge and client devices. For simplicity, we chose the same delay between each device $i \in K$ devices and the edge when formulating the problem. The results show that for different values of T, the optimal number of clients is almost reached at μ = 10. Further increase in μ leads to higher number of clients and thus the lower number of clients suppress sending their model updates by discarding their training. For μ > 10 larger timer values are closer to T and when μ < 10 the timer values are closer to zero. In both cases, we have a large number of clients because the timer values are distributed in a narrow range. So,



(a) $E(X)$ vs. μ for T = $4d$, $6d$, $8d$, and number of clients K = 1000

(b) $E(X)$ vs. number of clients K for μ = 10, and T = $4d$, $6d$, $8d$

Figure 2: Expected number $E(X)$ of selected clients for model updates given exponentially distributed timers

for the given scenario, if we set the μ parameter close to or equal to 10, our queue will stay below the overflow. Also, all clients whose training time is close to or equal to the minimum timer will complete their training and send their model updates before receiving the ACK from the edge. Only the clients whose training time is greater suppress both training and model updates. So it is not feasible to set the time equal to the delay time, as this would completely ignore the straggler devices.

We repeat the experiment by fixing the value of the parameter μ and changing the values of T. From Figure 2b, we can see that for the smaller values of T, the expected number of clients in the queue $E(X)$ increases with the total number of clients. However, when we move to the larger values of T, $E(X)$ remains stable even with a larger number of clients. This is because for larger values of T, clients will receive the ACK before their timer expires, so a large number of clients will suppress their training.

3. Experimental Environment and Results for Federated Learning

In the process of federated learning, each client sends its model updates to the server, which aggregates them in each round of communication. However, when the number of clients is large, the cost and overhead of communication between clients and edge can be a significant bottleneck. In this paper, we propose to use a time-based technique to select only a small subset of clients and communicate only the updates of this subset to the server. Our goal is to aggregate only a subset of client updates, where the subset of clients selected in each round is different and depends on the exponential timer, so that the final model update looks like an aggregation of all client updates.

We run the experiments with 50 user nodes participating in the FL process. Increasing the number of nodes would be possible if a more powerful test environment were available. To select a different number of clients using a probabilistic time-based client method based on

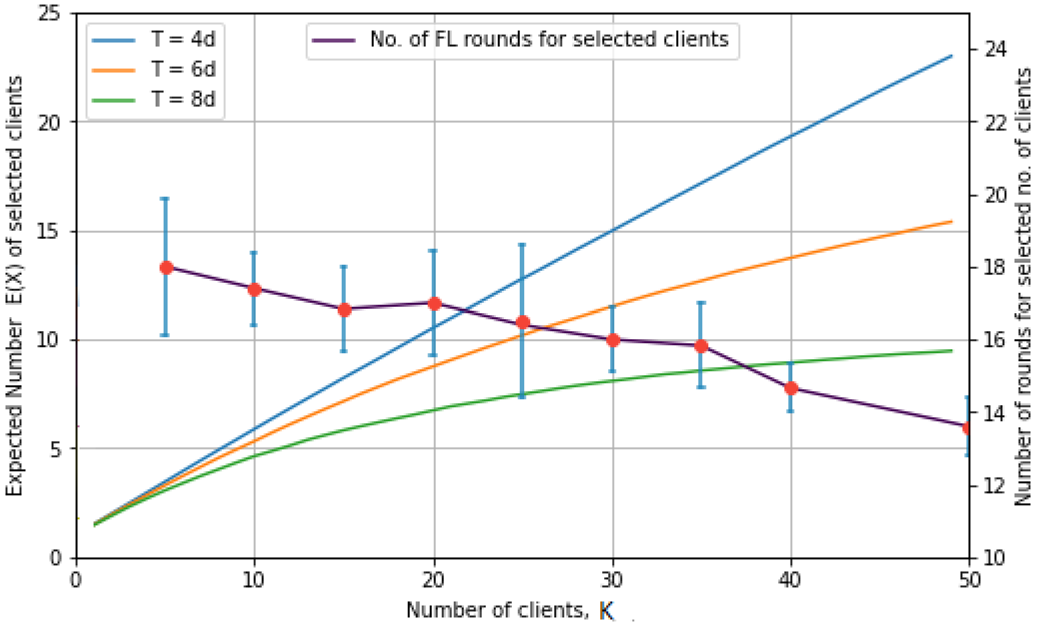


Figure 3: Expected number $E(X)$ of selected clients for model updates with exponentially distributed timers with μ = 10 and intervals of size T =$4d$, $6d$, and number of FL rounds for given number of clients K

exponentially distributed timers, we set the timer parameters such as μ and T for 5, 10, 15, 20, 30, and 40 clients to test how changing the number of clients affects the accuracy. Furthermore, during the FL process we assigned weights to all clients that corresponded to the number of samples that each client used for training. Each experiment is repeated 10 times to find the average number of rounds for the selected clients. We then calculate the confidence interval for the specified rounds and for the selected number of clients, as shown in Figure 3.

3.1. Dataset Distribution

We started with the MNIST dataset[2] for convenience. The MNIST dataset contains 60,000 training examples and 10,000 test examples. The image has a fixed dimension of 28x28 pixels with a value between 0 and 9. Each image is converted into an array of 784 features. We require each client to randomly select images from a different subset of the training data that are not independently and identically distributed (non-iid). For the training dataset of the experiment, each user selects 1200 images at random, without repetitions.

3.2. Experimental Results

We use for the MNIST classification a neural network consisting of three layers, with 200 neurons in the first two layers and 10 output neurons in the third layer, with a learning rate of

[2] http://yann.lecun.com/exdb/mnist/

0.01, a batch size of 32, and each client performs 10 local epochs before sending its model updates. As shown in Figure 3, the number of FL rounds decreases as we increase the number of clients. With 5 clients, we have a maximum number of rounds of 20, and with 50 clients, 13 rounds to achieve 97% accuracy. In these experiments, we distribute the updated model to all clients and not only to the clients that participated in the aggregation process, so whichever clients we select have the same updated global model. This also reduces the communication cost selecting only a subset of clients that deliver their local parameters and saves the computational resources of the rest of the ensemble leading to a fast convergence in a small number of rounds. Also, the communication cost from the server to distribute the model to all clients is compensated by a smaller number of rounds. If we look at Figure 3, the intersection of the two lines shows the optimal working point at which we expect $E(X)$ clients selected for the FL process based on above-discussed parameters μ and T, and the relative number of rounds for a given number of clients to achieve the desired accuracy, which in these tests was up to 97%. After repeating each test more than 10 times, we also defined a confidence interval of 95% for the number of rounds to achieve the desired accuracy. According to the empirical results, our approach improves learning efficiency and enables more uniform and fair performance among clients.

4. Conclusion and Future Works

In this work, we have investigated a technique for selecting clients in the FL process by using the probabilistic timers for a large number of clients. The proposed technique is robust and scalable for a large number of clients. It controls the number of updates to be received to control the queue length at the edge by adjusting the parameters for the average number of clients and the latency for each round. We also investigate the relationship between adjusting the parameters for the exponential distribution and the convergence time for FL. In future work, we will refine the methods and details of the client selection technique by also considering the heterogeneous delays between edge and client devices and analyzing the heterogeneous computational constraints between devices. We will also try to implement the timers with different probability models to see which distribution is more effective. While the focus of this research is on the impact of client sampling on the global optimization goal of FL, future developments could investigate the impact of the client selection technique on the performance of individual users, especially in the context of heterogeneity.

Acknowledgments

This research was partially supported by TEACHING project funded by the EU Horizon 2020 research and innovation programme under GA n. 871385, by the Italian Ministry of Education and Research in the framework of the CrossLab project (Departments of Excellence), and by the University of Pisa in the framework of the PRA 2020 program (AUTENS project, Sustainable Energy Autarky).

References

[1] Cisco annual internet report, 2021. URL: https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html.

[2] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[3] D. Bacciu, S. Akarmazyan, E. Armengaud, M. Bacco, G. Bravos, C. Calandra, E. Carlini, A. Carta, P. Cassarà, M. Coppola, et al., Teaching-trustworthy autonomous cyber-physical applications through human-centred intelligence, in: 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), IEEE, 2021, pp. 1–6.

[4] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, Y.-J. A. Zhang, The roadmap to 6g: Ai empowered wireless networks, IEEE Communications Magazine 57 (2019) 84–90. doi:10.1109/MCOM.2019.1900271.

[5] Y. Shi, K. Yang, T. Jiang, J. Zhang, K. B. Letaief, Communication-efficient edge ai: Algorithms and systems, IEEE Communications Surveys & Tutorials 22 (2020) 2167–2191.

[6] K. B. Letaief, Y. Shi, J. Lu, J. Lu, Edge artificial intelligence for 6g: Vision, enabling technologies, and applications, IEEE Journal on Selected Areas in Communications 40 (2021) 5–36.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.

[8] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019-2019 IEEE international conference on communications (ICC), IEEE, 2019, pp. 1–7.

[9] Y. J. Cho, S. Gupta, G. Joshi, O. Yağan, Bandit-based communication-efficient client selection strategies for federated learning, in: 2020 54th Asilomar Conference on Signals, Systems, and Computers, IEEE, 2020, pp. 1066–1069.

[10] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, V. Smith, On the convergence of federated optimization in heterogeneous networks, arXiv preprint arXiv:1812.06127 3 (2018) 3.

[11] M. F. M. Al-Athari, Estimation of the mean of truncated exponential distribution, Journal of Mathematics and Statistics 4 (2008) 284.

[12] J. Nonnenmacher, E. W. Biersack, Optimal multicast feedback, in: Proceedings. IEEE INFOCOM'98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98, volume 3, IEEE, 1998, pp. 964–971.