Server-side Adaptive Federated Learning over Wireless Mesh Network

Felix Freitag[1], Lu Wei[2], Chun-Hung Liu[3], Mennan Selimi[4], and Luís Veiga[5]

[1] Universitat Politècnica de Catalunya, BarcelonaTech, Spain
felix.freitag@upc.edu
[2] Texas Tech University, Lubbock, TX, USA
luwei@ttu.edu
[3] Mississippi State University, Starkville, MS, USA
chliu@ece.msstate.edu
[4] Max van der Stoel Institute, South East European University, North Macedonia
m.selimi@seeu.edu.mk
[5] INESC-ID Lisboa, Instituto Superior Técnico, University of Lisbon, Portugal
luis.veiga@inesc-id.pt

Abstract. In federated learning, distributed nodes train a local machine learning model and exchange it through a central aggregator. In real environments, these training nodes are heterogeneous in computing capacity and bandwidth, thus their specific characteristics influence the performance of the federated learning process. We propose for such situations the design of a federated learning server that is able to adapt dynamically to the heterogeneity of the training nodes. In experiments with real devices deployed in a wireless mesh network, we observed that the designed adaptive federated learning server successfully exploited the idle times of the fast nodes by assigning them larger training workloads, which led to a higher global model performance without increasing the training time.

Keywords: edge computing, federated learning, machine learning

1 Introduction

Federated learning is a recent paradigm for collaboratively training machine learning models with distributed nodes [1]. Different from centralized training in which all the training data are managed by a single node, in federated learning each node has its own training data to train a local model. The subsequent aggregation of the local models in a central aggregator leads to a new global model. One important advantage of federated learning is that the local training data does not need to leave the node where it is used for training, which preserves the privacy of the data.

Federated learning uses a distributed computing infrastructure, and therefore heterogeneity can be expected in several forms: 1) The quantity and quality of local data at each node may vary. For instance, local data may be acquired

by sensors at or nearby a node, but the local circumstances of each node can lead to a different number of training samples. 2) The computing capacity of the nodes can be different either due to the proper hardware of the nodes or by concurrent executions of other applications at the node, which reduce the available computing resources dedicated to the federated learning process. 3) For exchanging the trained model, some nodes may face limited network bandwidth either due to permanent or dynamic network conditions.

In this paper, we address the federated learning process in a heterogeneous environment with the goal to develop an adaptive design of the federated learning server for exploiting this heterogeneity. Specifically, we propose the federated learning server to determine at each round the workload capacity of each node for model training in a determined time slot. Applying our design, each client receives from the server an individual training instruction that includes the number of samples in each training round. As a consequence, the server is able to exploit the computing capacity of the faster nodes in the training process. In federated learning, this leads to higher-performing global models during the training process compared to the non-adaptive design.

We experiment with the proposed adaptive federated learning server design with distributed computing devices consisting of mini-PCs or Single-Board Computer (SBC). This scenario was chosen as being representative of realistic user environments, where computing devices run as home servers to manage several services while at the same time each server is connected over a network with other servers to collectively perform federated learning. The heterogeneity of the nodes is induced by the different hardware of each of these home servers. In addition, the heterogeneity comes from the fact that in real-world usage these devices will not be dedicated exclusively to the federated learning application but concurrently run other user-oriented services, which results in each training node having a potentially variable computing capacity, thus making our scenario with heterogeneity realistic.

The main contributions of the paper are:

1. We develop the design of an adaptive federated learning server being able to exploit the clients' heterogeneity that leads to an increase in the performance of the global model.
2. We provide the evaluation of the proposed design by experimenting with its implementation on real distributed low-capacity devices connected to a wireless mesh network.

2 Background and related works

Federated learning has raised the interest of the research community as a technique for model training that does not require the sharing of a node's local data. An important area for federated learning is the application to wireless communications such as 5G, where the edge nodes generate valuable data for applications but at the same time, these data must be held private [2, 3].

Federated learning can be considered a distributed computing process requiring computing and communication resources. This translates federated learning into having local computing capacities at the nodes which perform the training, as well as to have the communication capability that allows to transmit the machine learning models between the training nodes and the central aggregator. While powerful edge nodes such as that of the 5G system do have such computing and communication capacities to support federated learning, other edge nodes like SBC and tiny embedded Internet of Things (IoT) devices may have computation and communication challenges [4] [5].

The heterogeneity of the communication capacity in federated learning scenarios was addressed by Jiang et al. in [6]. Their work on the BACombo system considers scenarios of training nodes represented by mobile phones or embedded devices, where the network connectivity of each node is different. The solution proposed in the BACombo federated learning system is to leverage the network between the training nodes for pulling segments of the model updates between clients. The process is supported by a worker selection step at each client that monitors the bandwidth to each of the other workers. It can be observed that in BACombo the heterogeneity is addressed on the client side, i.e., the training nodes are given an extended decision capacity to influence the federated learning process. Differently, in the design that we propose, the decision on the training parameters of each client is made at the server.

Federated learning in edge environments was proposed in several works, as surveyed in [7]. Specific types of edge devices are investigated for instance in the Flower framework, where Android phones, Raspberry Pi, and NVIDIA Jetson were used [8]. The work on Flower proposes a framework that first addresses the hardware heterogeneity of the clients by providing client-specific software implementations. For instance, the federated learning client for Android phones consists of a Java implementation applying a specific TensorFlow Lite Model Personalization support for Android Studio. The federated learning client for the Raspberry Pi and NVIDIA Jetson is implemented in Python. Secondly, in order to address the different hardware capacities of the clients, the Flower architecture proposes a strategy component as part of the federated learning server that assigns to each client a fixed cutoff time for sending back the model based on previous offline observations. The value chosen for this cutoff time results in a trade-off between the training time and the accuracy of the model. Differently, in our design, we propose to apply an adaptive training configuration, where the training parameters for each client are determined by the server at each training round and are based on the online client performance observed in previous rounds.

In the work of Wang et al. [9], an adaptive federated learning approach is proposed which focuses on the frequency of performing global aggregations under resource constraints. The problem addressed is the optimization of resource consumption of federated learning, both related to computation and communication. The work proposes a control algorithm to determine in real-time after how many local training epochs the model data is sent back to the aggregator

node, targeting to minimize a loss function under resource budget constraints. Similar to our approach, in their work the decisions are taken on the server side and are calculated along the training process. The difference is that we propose the server to exploit the faster client's computing capacity and reduce its idle time.

Another work of Wang et al. [10] proposes a communication-efficient compressed federated adaptive gradient optimization framework, FedCAMS, which largely reduces the communication overhead and addresses the adaptivity issue in federated optimization methods. Differently, FedCAMS is evaluated locally and not in a wireless mesh network.

The evaluation is performed by simulations and some experiments in real nodes consisting of 3 Raspberry Pi and 2 laptop computers.

In FedMax [11], a highly-efficient federated learning framework is presented. The heterogeneity of the worker nodes is related to the context of the IoT. Two measures are suggested, which are relaxed worker synchronization for tolerating dropouts of sporadic workers, and similarity-based worker selection, which aims to select a subset of the most efficient workers. FedMax is similar to our approach as it measures the client performance at the server and allows the server to make decisions for each worker, such as assigning a different amount of training load to these workers. However, FedMax is evaluated in the Google Cloud Platform, where the heterogeneity is configured by the number of CPUs per VM having homogeneous communication. In our work, we use a real edge environment with different SBC nodes interconnected over a wireless mesh network with links of different communication capacities.

In our own previous work [4], we demonstrated for the real deployment of federated learning in a wireless mesh network how the heterogeneity of the clients' computing capacity and that of the communication to the server affects the federated learning process. We showed how slower clients, either due to bandwidth or computing limitations, delay the federated learning process. Therefore, in the current paper, we present as a continuation of the situation observed in our earlier work a federated learning server being able to adapt the training load to the capacity of each client.

3 Design of adaptive federated learning

3.1 Server-side adaptive federated learning

We consider a scenario where a server conducts federated learning rounds with a set of heterogeneous clients. This heterogeneity is induced by different bandwidths, different client hardware, and/or different computing capacity usage at each client node.

Federated learning uses a star topology, where a central node, i.e., the federated learning server, orchestrates the training with the registered clients (Figure 1). The role of the federated learning server at the end of a training round is to merge the local models received from the clients into a new global model. For

the next round, this new model is sent out to the clients, along with training parameter values. We propose to add the capacity of adaptation to the federated learning server. Thus, the server determines for each round the learning parameters for each client. Therefore, the interaction between the server and clients not only includes the sending of the new global model to the clients, but also the individual learning parameters.

In order to orchestrate adaptive federated learning, the server takes measurements at each training round. A key metric that the server calculates is the *workload capacity*, which is a client-specific metric. The metric is defined as the number of samples by epochs trained divided by the time that passed between sending the global model to the client and receiving the local model back. The metric contains the time spent for the communication of the model, from the client to the server and back, and the time of the proper model training at the client.

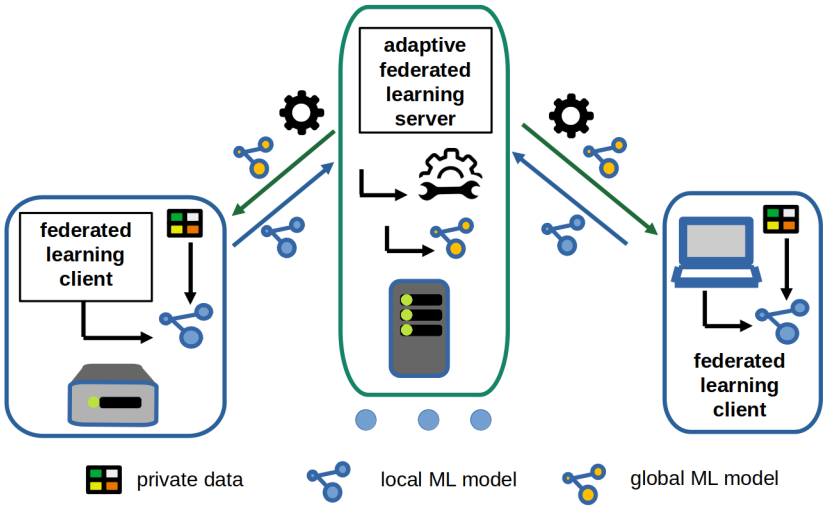

Fig. 1: Overview of the Architecture of server-side adaptive federated learning.

3.2 Federated learning implementation

The federated learning network we use for the experimentation is implemented in Python language. The system is composed of two major components which are the code for the client and the server. In our implementation, the server sends both the model parameters and the learning parameters, which relate to how the training has to be done at the clients. These learning parameters are the learning rate, number of local training epochs, and batch size. It is specific to the adaptive server design that the server makes a distinction between the

different client nodes, i.e., each client can receive a different value of the learning parameters. These data are sent between servers and clients in JSON format over HTTP POST messages. Both the server and the clients implement a REST API.

For both the federated learning server and the client code, we create Docker images in order to instantiate them with Docker containers on the different devices used in the experimentation. The source code of the federated learning network is available on Github[6]. Additional information on the code design can be found in [12].

4 Experimental evaluation of the adaptive federated learning network

4.1 Experimental environment

The objective of the experimentation is to observe the effect of the adaptive server design in federated learning running in a real edge environment.

For conducting the experimentation we use SBC connected to the GuifiSants network[7], a wireless city mesh network. It is a subset of the larger Guifi.net community network[8]. This is the same experimental environment we already used in our previous work [4].

In order to experiment with hardware heterogeneity, we use three types of devices for the clients in the federated learning network. Two of them are SBC, specifically the Minix mini-PC NEO Z83-4 with Intel Atom x5-Z8350 processor and 4GB DDR3 RAM[9] and the PC Engine APU2 with an AMD Embedded G series GX-412TC processor and 4 GB DDR3 RAM[10]. The third type of hardware used for running a federated learning client was a laptop with an i5 processor, hosting also the federated learning server. Figure 2 illustrates the deployment of the three different clients.

With regards to the communication heterogeneity, it can be observed in Figure 2 that the two SBC clients have a rather low capacity link with the server, different to the third client that runs on the same machine as the server.

The federated learning task to be executed in the experiments is to train a 6-layer Convolutional Neural Network (CNN) model with the Chest_X_ray dataset[11]. The CNN model has around $420,000$ parameters.

[6] `https://github.com/eyp/federated-learning-network`
[7] `http://sants.guifi.net/`
[8] guifi.net: Commons Telecommunication Network Open Free Neutral`http://guifi.net/`
[9] `https://minix.com.hk/products/neo-z83-4-pro`
[10] `https://pcengines.ch/apu2e4.htm`
[11] Chest X-Ray Images. `https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia`

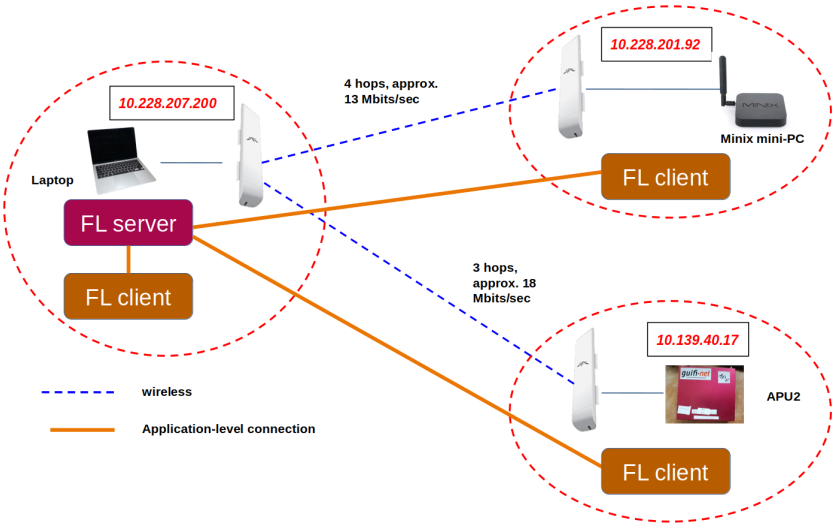

Fig. 2: Testbed nodes within GuifiSants used for the experimentation.

4.2 Server-side adaptive federated learning with heterogeneous clients

In this section, server-side adaptive federated learning is studied experimentally. In this experiment, we use three types of hardware to host three federated learning clients. Specifically, we use a VM with two cores running on a host with an i5 processor, the Minix device and the APU2 device, which in terms of computational capacity results in having a fast, medium, and slow client. We perform 50 training rounds.

Experiment 1: Baseline. In this experiment, we measure different metrics when the clients train with 6 training samples each round along a total of 50 training rounds (Figure 3). Slight variations in the fastest client along the 50 rounds are due to the fact that this client was running on a non-dedicated machine.

Experiment 2: Adaptive server. In this experiment during the first 10 rounds, the server applies its default behavior (without being adaptive). Then, from rounds 11 to 50 the server applies the adaptive behavior. The experimental setting for training was having a minimum number of 6 training samples (which the server can increase due to the adaptive behavior) and inference after training was done with 200 test samples at all clients. Since the number of training samples used by each client varies with the adaptive behavior, the federated averaging algorithm at the server was extended to apply weighted averaging.

Figure 4 shows the obtained results when the adaptive server is used. In Figure 4 it can be seen how the training time of the fast clients after 10 rounds increases to that of the slow clients. This is due to the fact that the server increased the number of training samples for the fast clients according to each

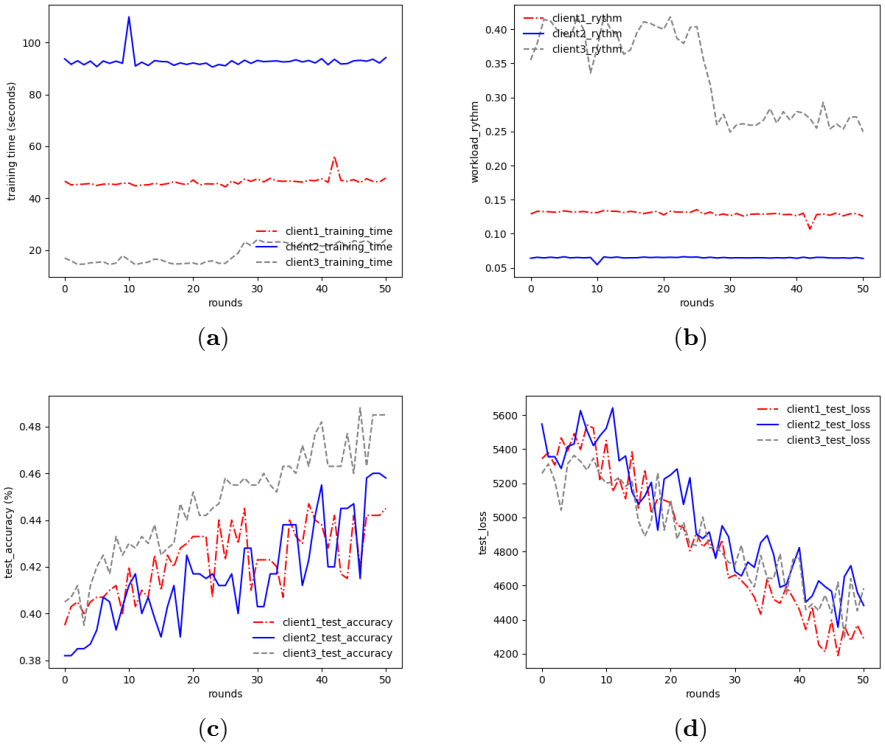

Fig. 3: Baseline. Behavior of three clients.(**a**) Training time. (**b**) Server-measured client rythm. (**c**) Test accuracy. (**d**) Test loss.

client's workload capacity. Specifically, the client running on the Minix mini-PC is trained with about one hundred samples, the client on the APU2 devices with up to around 800 training samples, and the fastest client on the i5 laptop is trained with the original 6 samples. Comparing the accuracy achieved by the adaptive federated learning server in Figure 4c with the accuracy of the baseline training (Figure 3c), it is observed that with the adaptive server it is significantly higher. This is expected since the number of images used for training in the adaptive server configuration is higher. However, since the reference time for adjusting the number of training images for the fast clients is the training time of the slowest client, the overall time for training the model in both baseline and adaptive federated learning is similar.

5 Conclusions

This paper presented a server-side adaptive federated learning design. The experimentation with the developed implementation was conducted in a real en-

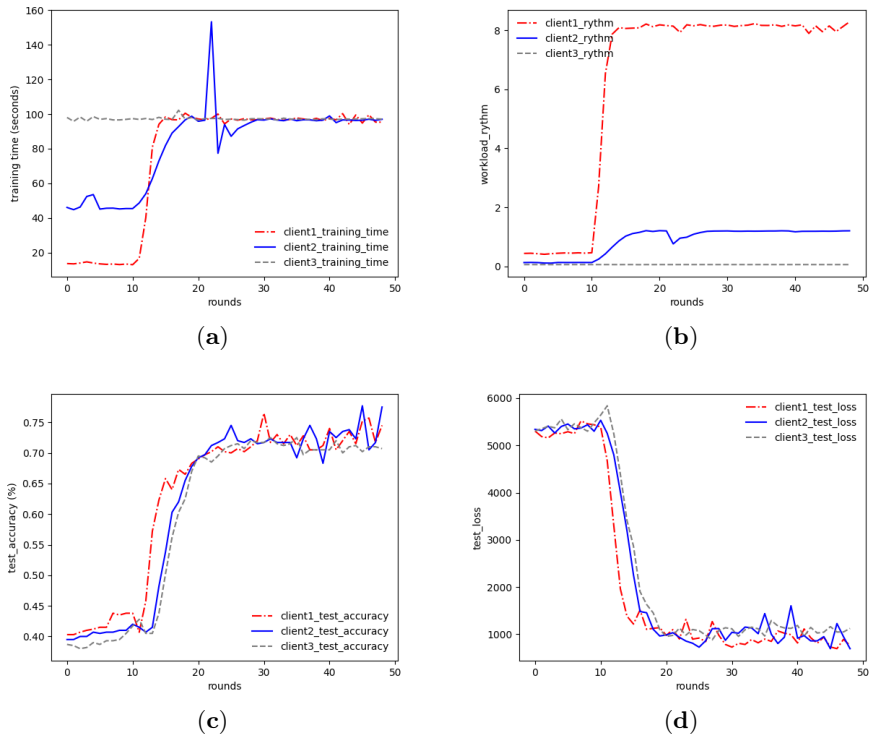

Fig. 4: Adpative server. Behavior of three clients.(**a**) Training time. (**b**) Server-measured client rythm. (**c**) Test accuracy. (**d**) Test loss.

vironment with SBC and wireless network connectivity. It was shown with the adaptive design the server successfully exploited the idle time of the faster clients by assigning them higher training workloads. The better-performing local models of the faster clients improved the overall global model performance by the model aggregation in the federated learning server without increasing the training time.

The results may have interesting applicability in other resource-constrained edge scenarios. Specifically, we aim to extend our results to embedded IoT devices, where the studied design could address critical computing and communication resource constraints. Another line of work is to combine the server-side adaptation with that on the client side.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871582 — NGIatlantic.eu and was partially supported by the Spanish Government under contracts PID2019-106774RB-C21, PCI2019-111851-2 (LeadingEdge CHIST-ERA),

PCI2019-111850-2 (DiPET CHIST-ERA), and by national funds through FCT, Fundação para a Ciência e a Tecnologia, Portugal, under project UIDB/50021/2020. The work of C.-H. Liu was supported in part by the U.S. National Science Foundation (NSF) under Award CNS-2006453 and in part by Mississippi State University under Grant ORED 253551-060702. The work of L. Wei is supported in part by the U.S. National Science Foundation (#2006612 and #2150486).

References

1. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2) (January 2019)
2. Niknam, S., Dhillon, H.S., Reed, J.H.: Federated learning for wireless communications: Motivation, opportunities, and challenges. IEEE Communications Magazine **58**(6) (2020) 46–51
3. Ibraimi, L., Selimi, M., Freitag, F.: Bepoch: Improving federated learning performance in resource-constrained computing devices. In: IEEE Global Communications Conference (GLOBECOM). (2021)
4. Freitag, F., Vilchez, P., Wei, L., Liu, C.H., Selimi, M.: Performance evaluation of federated learning over wireless mesh networks with low-capacity devices. In Rocha, Á., Ferrás, C., Méndez Porras, A., Jimenez Delgado, E., eds.: Information Technology and Systems, Cham, Springer International Publishing (2022) 635–645
5. Llisterri Giménez, N., Monfort Grau, M., Pueyo Centelles, R., Freitag, F.: On-device training of machine learning models on microcontrollers with federated learning. Electronics **11**(4) (2022)
6. Jiang, J., Hu, L., Hu, C., Liu, J., Wang, Z.: Bacombo—bandwidth-aware decentralized federated learning. Electronics **9**(3) (2020)
7. Abreha, H.G., Hayajneh, M., Serhani, M.A.: Federated learning in edge computing: A systematic survey. Sensors **22**(2) (2022)
8. Mathur, A., Beutel, D.J., de Gusmão, P.P.B., Fernandez-Marques, J., Topal, T., Qiu, X., Parcollet, T., Gao, Y., Lane, N.D.: On-device federated learning with flower. (2021)
9. Wang, S., Tuor, T., Salonidis, T., Leung, K.K., Makaya, C., He, T., Chan, K.: Adaptive federated learning in resource constrained edge computing systems. IEEE Journal on Selected Areas in Communications **37**(6) (2019) 1205–1221
10. Wang, Y., Lin, L., Chen, J.: Communication-efficient adaptive federated learning (2022)
11. Xu, H., Li, J., Xiong, H., Lu, H.: Fedmax: Enabling a highly-efficient federated learning framework. In: 2020 IEEE 13th International Conference on Cloud Computing (CLOUD). (2020) 426–434
12. Parareda, E.Y.: Federated learning network: Training distributed machine learning models with the federated learning paradigm. (2021)