A Traffic Model Based Approach to Parameter Server Design in Federated Learning Processes

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, Stefano Savazzi, *Member, IEEE*, and Monica Nicoli, *Senior Member, IEEE*

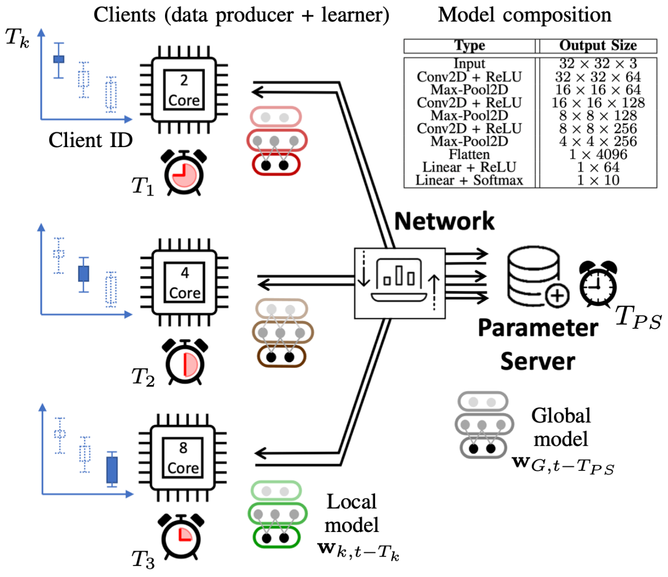
***Abstract*—This letter proposes a model to describe the data traffic generated by a Federated Learning (FL) process in a wireless network with asynchronous Parameter Server (PS) orchestration and heterogeneous clients. The model accounts for the local update processes implemented by individual clients and it is used to enforce requirements on the PS design, namely to regulate the interval among consecutive global model updates. PS requirements are validated on realistic pools of resource-constrained wireless edge devices, typically found in Internet-of-Things (IoT) setups. Numerical results show that the proposed policy is effective when devices have unbalanced resources, namely, different sample distributions and computational capabilities. It permits an accuracy gain of up to 15-17% on average with respect to typical asynchronous PS designs.**

***Index Terms*—Federated learning over networks, traffic modelling, edge devices, computing.**

I. Introduction

Federated Learning (FL) enables resource-constrained edge devices to cooperate over a network for training a shared Machine Learning (ML) model. It protects data ownership by ensuring that the observations used for training never leave the device responsible for its production. As depicted in Fig. 1, FL alternates the computation at each device of local model parameters, i.e., the weights of deep neural network layers, with the communication to a Parameter Server (PS) that fuses the local models and returns a global model [1]. Different FL implementations [2] and enablers [3] emerged in the past few years. Most applications call for geographically distributed [4] and heterogeneous clients with different temporal alignments. In many cases, an asynchronous orchestration of the FL process is also a prerequisite, especially in next generation networks.

Current state-of-the-art on asynchronous FL strategies mainly have the following limitations. First, in vanilla algorithms, the PS updates the global model as soon as a local model is received [5], with no regard to client-specific resource constraints. This can lead, for example, to biased updates from faster clients. Secondly, the update at the clients is not optimized as the number of local epochs is usually fixed and not



Type	Output Size
Input	$32 \times 32 \times 3$
Conv2D + ReLU	$32 \times 32 \times 64$
Max-Pool2D	$16 \times 16 \times 64$
Conv2D + ReLU	$16 \times 16 \times 128$
Max-Pool2D	$8 \times 8 \times 128$
Conv2D + ReLU	$8 \times 8 \times 256$
Max-Pool2D	$4 \times 4 \times 256$
Flatten	1×4096
Linear + ReLU	1×64
Linear + Softmax	1×10

Fig. 1. FL system with heterogeneous and asynchronous clients. Optimized T_{PS} takes into account the local model update completion time of the clients. Top-right corner: model composition for the experiments.

tuned according to the type of traffic or the quantity/quality of the data [6]. The letter proposes a moment-matching approximation to represent the traffic generated by clients engaged in an asynchronous FL process. The model is validated through a real FL prototype consisting of physically separated clients implementing distributed training over a wireless network, using the Message Queuing Telemetry Transport (MQTT) protocol. Besides adapting and formalizing the moment matching technique to the context of FL, the letter provides the necessary requirements on the time interval among consecutive global model updates, namely the server response time T_{PS}. This is used by the PS to decide whether to update the global model or not, depending on the backlog of local models retained by the clients. The proposed requirements account for the traffic type, the local data size, quality, and the channel impairments affecting the FL local round time.

The letter is organized as follows. Sect. II introduces the proposed traffic model for asynchronous FL. The model uses the moment-matching approximation and permits to categorize the traffic of each client using the dispersion index (D) metric. Requirements in Sect. III exploit the proposed model to define operational points that regulate the clients and the PS behavior, while Sec. IV describes a practical policy for T_{PS} selection that fulfills the proposed requirements. The policy is validated through a real-time FL platform prototype.

Manuscript received 15 October 2022; revised 6 December 2022, 23 January 2023, and 14 March 2023; accepted 18 April 2023. Date of publication 3 May 2023; date of current version 12 July 2023. The associate editor coordinating the review of this letter and approving it for publication was Z. Yang. *(Corresponding author: Bernardo Camajori Tedeschini.)*

Bernardo Camajori Tedeschini is with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy (e-mail: bernardo.camajori@polimi.it).

Stefano Savazzi is with IEIIT Institute, Consiglio Nazionale delle Ricerche (CNR), 20133 Milan, Italy (e-mail: stefano.savazzi@ieiit.cnr.it).

Monica Nicoli is with the Dipartimento di Ingegneria Gestionale (DIG), Politecnico di Milano, 20133 Milan, Italy (e-mail: monica.nicoli@polimi.it).

Digital Object Identifier 10.1109/LCOMM.2023.3272844

II. FL Data Traffic Model

We consider a FL system composed of one PS and a set of K clients $\mathcal{K} = \{1, \ldots, K\}$, each with its own private dataset $\mathcal{S}_k$ of size $S_k = |\mathcal{S}_k|$. As depicted in Fig. 1, the aim of the FL process is to obtain a global ML model, of size G, that minimizes a loss function $\mathbf{w}_G = \operatorname{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w})$ with $\mathcal{L} = \frac{1}{K}\sum_{k=1}^{K} \mathcal{L}_k(\mathbf{w}, \mathcal{S}_k)$ and $\mathcal{L}_k$ being the local costs measured by clients using the data batches $\mathcal{S}_k$. The FL process requires the clients to produce local models through optimization, typically via supervised and gradient-based methods. Each client k performs $M(k)$ local epochs before exchanging the local model with the PS, which is in charge of the global model update.

In asynchronous FL, the PS produces a new instance of the global model $\mathbf{w}_{G,t}$ at time $t = nT_{PS}, n = 1, \ldots, N_{FL}$ [4]:

$$\mathbf{w}_{G,t} = (1 - \epsilon_t)\mathbf{w}_{G,t-T_{PS}} + \frac{\epsilon_t}{\sum_{l=1}^{K} S_l} \sum_{k=1}^{K} S_k \mathbf{w}_{k,t-T_k}, \quad (1)$$

where $\mathbf{w}_{k,t-T_k}$ represents the k-th local model available at time $t-T_k$, $T_k \geq 0$ is the time interval required by the client k to produce an updated local model, while T_{PS} regulates the time span between two global model updates. $n \leq N_{FL}$ is the index of the federated rounds. Finally, ϵ_t controls the stability of the update. Considering that clients may have different computing capabilities and datasets, the associated network traffic can vary significantly depending on T_k. A client-specific characterization is thus proposed to model the local model update process and identify the corresponding traffic pattern.

For the exchange of the NN model parameters among the clients and the PS, we propose to employ the MQTT protocol [7] as it enables a real-time exchange of the model parameters and allows the monitoring of the client training time required for T_{PS} tuning. The time required by a client k to implement a local round can be broken down into the time span to download the weights (T_{down}), train the new model for $M(k)$ epochs using local data batches (T_{train}), encrypt, compress and upload the weights, i.e., to the MQTT broker, (T_{up}):

$$T_k = T_{\text{down}}(k) + M(k) \cdot T_{\text{train}}(k) + T_{\text{up}}(k). \quad (2)$$

These quantities can be computed locally by each client, through standard time measurement functions, and permit to separate the contribution of computing capabilities (T_{train}) from possible channel disturbances affecting uplink (UL) and downlink (DL) communications ($T_{\text{up}}, T_{\text{down}}$).

Based on the above assumption, we introduce a model to approximate the probability density function $p_{T_k}(T_k)$ of the traffic pattern generated by each client k. A moment-matching approximation is employed which divides the process into three categories: Bernoulli, Poisson, and Pascal [8]. We classify the traffic into one of these categories by matching the first two moments defined respectively as:

$$\begin{aligned} A(k) &= \mathbb{E}_n[T_k], \\ \sigma^2(k) &= \mathbb{E}_n[(T_k - A(k))^2],\ n \in \{1, \ldots, N_{FL}\}. \end{aligned} \quad (3)$$

The Dispersion Index (D), also called Variance to Mean Ratio (VMR), is:

$$D(k) = \frac{\sigma^2(k)}{A(k)}, \quad \forall k \in \mathcal{K}. \quad (4)$$

According to the moment-matching technique, we can obtain a Poisson traffic by setting $D(k) = 1$, i.e., by imposing a regular traffic pattern. On the contrary, burst-traffic, i.e., Pascal, and smooth traffic, i.e., Bernoulli, are obtained with $D(k) > 1$ and $D(k) < 1$, respectively. Based on the above metrics, in the following, we give upper and lower bounds on the characteristics of the PS, especially regarding the T_{PS}.

III. Minimal Requirements on Clients and T_{PS}

The choice of the server response time T_{PS} is underpinned by the local model update process running on each client, therefore by the number $M(k)$ of epochs that directly reflects on the dispersion index $D(k)$ in (4). Low values of $M(k)$ correspond to frequent contributions of the clients to the global model, at the expense of an increased communication overhead, and possibly non-informative local model updates. Conversely, large $M(k)$ forces the client to implement many local epochs and possibly produce a biased local model (penalized by overfitting).

Optimal $M(k)$ should be bounded as $M_L(k) < M(k) < M_U(k)$. The lower bound $M_L(k)$ sets the minimum $M(k)$ such that the client local model can improve the FL process while satisfying the communication overhead constraints. The upper bound $M_U(k)$ is the maximum $M(k)$ before the client starts overfitting. Note also that $M_L(k)$ is limited by UL and DL maximal communication efficiency η_{MAX} [bit/sec/Hz] dedicated to the link between the PS and the clients, with bandwidth B. Being $T_{FL} = N_{FL}T_{PS}$ the FL training duration, it is:

$$\frac{T_{FL}}{A(k)} G \leq \eta_{MAX} B T_{FL}. \quad (5)$$

This leads to the following:

$$M_L(k) = \mathbb{E}_n \left[\frac{1}{T_{\text{train}}(k)} \left(\frac{G}{\eta_{MAX} B} - (T_{\text{down}}(k) + T_{\text{up}}(k)) \right) \right]. \quad (6)$$

Note that $M_U(k)$ depends mainly on the size of training data and the local model, since more data (or small sized models) require more local epochs for overfitting. For client k, and $\mathbf{w}_{k,m}$ being the local model observed at local epoch $m \in \{1, \ldots, M(k)\}$, $M_U(k)$ is assigned as:

$$M_U(k) = \underset{m}{\operatorname{argmin}}\, \mathcal{L}_k(\mathbf{w}_{k,m}, \mathcal{S}_k^{\text{val}}), \quad (7)$$

where $\mathcal{L}_k(\mathbf{w}_{k,m}, \mathcal{S}_k^{\text{val}})$ is the validation loss computed by client k on the validation dataset $\mathcal{S}_k^{\text{val}}$ at epoch m.

As shown in the next section, the choice of $M(k)$ affects T_{PS} and can be used to set practical bounds on global model updates. On one hand, small $M(k)$ such that $M(k) \ll M_L(k)$ might result in $T_{PS} \gg A(k)$ thus exceeding the constraint on the spectral efficiency, with negligible effect on the FL process and accuracy. On the other hand, performing sporadic

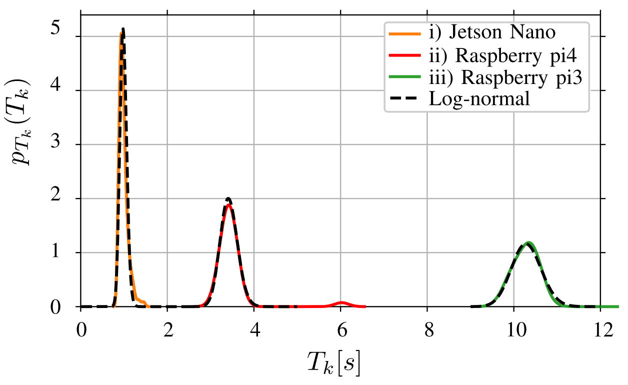


Fig. 2. In orange, red and green the probability density function $p_{T_k}(T_k)$ of a client with hardware ARM-Cortex-A57 SoC, GPU: 128-core Maxwell (*Jetson Nano* model, i), ARM-Cortex-A72 SoC (*Raspberry pi4*, ii), ARMv8-Cortex-A53 SoC (*Raspberry pi3*, iii), respectively. With dotted black line we represent the log-normal distribution that fits the real probability density function of T_k. $M(k)$ is set to 2 and the model size S is 51 KB.

global model updates, namely $M(k) > M_U(k)$, might produce biased local models when $T_{PS} \ll A(k)$. These could negatively contribute to the FL process by either slowing down convergence, reducing the accuracy [9] or possibly preventing the device to complete the local round [10].

IV. PS Design Principles and Validation

This section proposes a policy to regulate the PS response time T_{PS} based on the knowledge of the client dispersion index $D(k)$, the dataset S_k size and possible conditions on local overfitting. The proposed policy is validated in two scenarios where clients are characterized by different traffic patterns, namely varying computing capabilities, and non Independent and Identical Distributed (non-IID) datasets. Validation is based on a FL platform prototype.

A. FL Network Platform and Traffic Modelling

Fig. 2 provides a validation of the proposed client-specific traffic modelling approach based on moment matching. We consider a realistic pool of resource-constrained devices equipped with: i) CPU ARM-Cortex-A57 and GPU 128-core Maxwell (*Jetson Nano* model [11], orange), ii) CPU ARM-Cortex-A72 SoC (*Raspberry pi4*, red) and iii) CPU ARMv8-Cortex-A53 SoC (*Raspberry pi3*, green). For each client, we collected measurements of local round times T_k to obtain the sample probability functions $p_{T_k}(T_k)$. Notice that each client is connected via WLAN to a router which forwards the MQTT packets to the PS. The traffic parameters T_k and $D(k)$, are computed directly by clients at the end of each local round and then sent through a dedicated connection to the PS. The measured statistics $p_{T_k}(T_k)$ are thus reliable and realistic as they are independent from the PS hardware or from the FL processing. As evident from Fig. 2, the local round time distributions are well approximated by log-normal (dashed lines) with mean and standard deviation of 1/0.07, 3.4/0.2 and 10/0.3 for clients i), ii) and iii), respectively.

The goal of the following tests is to analyze the impact of client heterogeneity on PS response time T_{PS}. Based on experiments in Fig. 2, we simulate different execution times of the local rounds according to the log-normal model.

Algorithm 1 T_{PS} Policy

```
1: procedure POLICY($\mathcal{S}_k^{\text{train}}, \mathcal{S}_k^{\text{val}}$)            ▷ Run on client $k$
2:     Initialize $\mathbf{w}_{k,0}$, $M(k) \leftarrow 1$                              ▷ Epoch 0
3:     Train local model using $\mathcal{S}_k^{\text{train}}$
4:     Compute $D(k)$ with (4)
5:     Compute $M_L(k)$ with (6), $M_U(k)$ with (7)
6:     Compute performance metric: $P = \mathcal{P}(\mathbf{w}_{k,M_U(k)}, \mathcal{S}_k^{\text{val}})$
7:     $M^*(k) \leftarrow \max(\mathcal{F}[D(k), P], M_L(k))$
8:     $T_k^* \leftarrow T_{\text{down}}(k) + M^*(k)T_{\text{train}}(k) + T_{\text{up}}(k)$
9:     Return $T_{PS}^*(k) \leftarrow A^*(k) = \mathbb{E}_n[T_k^*]$
10: end procedure
```

B. Client-Specific Policy for the PS Response Time

The choice of the PS response time T_{PS} must take into account both the traffic model of Sec. II and the requirements of Sec. III. The optimal server time T_{PS}^* corresponds to a value $M^*(k)$ bounded by $M_L(k)$ and $M_U(k)$. The main idea, shown in Algorithm 1, is that each client computes its own optimal $M^*(k)$:

$$M^*(k) = \max(\mathcal{F}[D(k), P], M_L(k)), \tag{8}$$

where $\mathcal{F}$ is a policy function. Function $\mathcal{F}$ takes as input the local accuracy P and the traffic type $D(k)$. It can be written analytically as:

$$\mathcal{F}[D(k), P] = Q_k(\gamma P) - C \cdot D(k), \tag{9}$$

where $Q_k(\gamma P) = \{m : \mathcal{P}(\mathbf{w}_{k,m}, \mathcal{S}_k^{\text{val}}) = \gamma P\}$ is the number of epochs that corresponds to a validation accuracy of γP, and $\mathcal{P}(.)$ is the cross-entropy accuracy function. $P = \mathcal{P}(\mathbf{w}_{k,M_U(k)}, \mathcal{S}_k^{\text{val}})$ is obtained at local epoch $M_U(k)$, $C > 0$ is a constant (see Sec. IV-C) and $0 < \gamma < 1$ is a hyper-parameter. Optimal $M^*(k)$ is bounded as $M^*(k) \geq M_L(k)$ from (8), and $M^*(k) \leq M_U(k)$ which follows from (9), since $C \cdot D(k)$ is a positive term and $Q_k \leq M_U(k)$ as $\gamma P < P$.

By replacing $M^*(k)$ in (2), each client derives the PS time $T_{PS}^*(k) = A^*(k)$ using the device-specific parameters $M_U(k)$, $M_L(k)$ and $D(k)$, as well as the training $\mathcal{S}_k^{\text{train}}$ and validation datasets $\mathcal{S}_k^{\text{val}}$, respectively, as inputs. The device returns the $T_{PS}^*(k)$ value to the PS which makes a final decision for T_{PS}. The traffic statistics, the upper and lower bounds, $M_U(k)$ and $M_L(k)$, are obtained independently by each client during an initial training stage using $M(k) = 1$. The log-normal model parameters (3) are computed by means of consecutive time measurements T_k, that account for global model download, local training and model upload steps as in (2). After the training stage, we obtain the performance metric P and apply the policy function $\mathcal{F}$ in (8) using P and VMR $D(k)$.

Fig. 3 shows an example of local training, with loss and validation accuracy P for varying local epochs. Notice that few epochs are typically sufficient to improve the local model without incurring in overfitting. Cross-entropy function $\mathcal{P}$ in FL also follows a negative exponential behavior, namely $\mathcal{P} \approx a - e^{-bm}$, for $m < M_U$, while for such case $\gamma = 0.5$ is found as reasonable (see Sec. IV-C). With the proposed policy we avoid the overfitting region, transferring at the same time a great portion of local information. The term $C \cdot D(k)$ in (9)

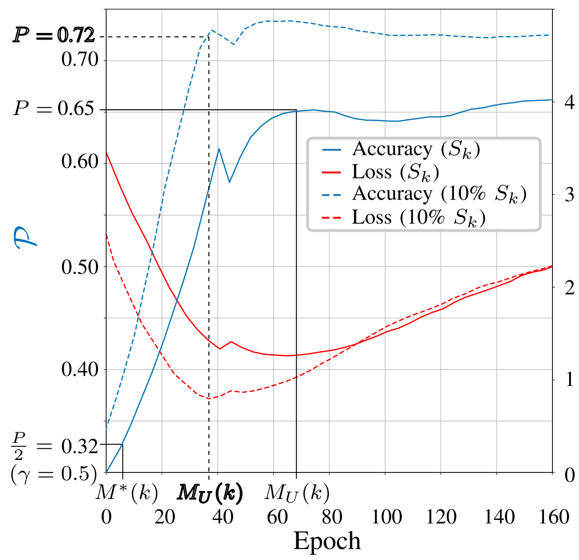


Fig. 3. Example of validation accuracy (blue) and loss (red) during local model training on a client. $M_U(k)$ and $M^*(k)$ values are highlighted together with accuracy P and γP ($\gamma = 0.5$) respectively. Solid and dotted lines are obtained with 100% and 10% of the training dataset (of size S_k). Note that overfitting is expected, as the local training process uses few training samples.

accounts for the traffic variance. Intuitively, a high variance, as a result of a client with varying computing resources, might increase the probability of observing high local training intervals (slow client). For such scenario, choosing a low value of $M(k)$ allows the PS to process the client local model updates more often and compensate for this effect.

The PS collects the optimal $T^*_{PS}(k)$ computed by Algorithm 1 and derives the response time to be used for all clients. Considering the previous analysis, this is obtained as:

$$T^*_{PS} = \min_k T^*_{PS}(k), \tag{10}$$

with the requirements on the efficiency already satisfied by $T^*_{PS}(k)$ since $M_L(k) \leq M^*(k) \leq M_U(k), \forall k$.

In the following, we explore two scenarios in detail. In the first one, the clients are homogeneous as featuring the same VMR $D(k)$, $M^*(k)$ and $T_{\text{train}}(k)$. In the second scenario, the clients are heterogeneous and organized into two clusters (C_1 and C_2). Clients in each cluster $k \in C_i$ have similar computing power/capabilities, namely $A(k) = A_i$, $\sigma^2(k) = \sigma_i^2$ and VMR $D(k) = D_i$, $\forall k \in C_i$ corresponding to the same processing unit, and TPU, if any. Accuracy improvements obtained by following the policy (10) are assessed in both cases.

C. Experimental Assessment

For the experiments, we consider the CIFAR10 [12] dataset using the full validation data and a local training set of $S_k = 500$ images for each of the $K = 9$ clients. Policy validation is based on a real-time FL platform prototype featuring physically separated clients (here *Jetson Nano* devices). The approach adopted for modelling the client heterogeneity is twofold. First, we used an additive delay whose log-normal distribution fits the observed completion times in Fig. 2. Second, the training data is non-IID distributed. To simplify the analysis, the data size is the same for all clients (as typical in FL). The prototype consists of devices connected via WLAN to the PS, thus permits to quantify the impact

TABLE I
HOMOGENEOUS CLIENTS AND NON-IID DATA. OPTIMAL T^*_{PS} AND $M^*(k)$ FOR VARYING $A(k)$, $D(k)$, η/η_{MAX} (%), AND CORRESPONDING ACCURACY IMPROVEMENT (%) W.R.T. VANILLA ASYNCHRONOUS FL

	$A(k)/D(k)$	T^*_{PS}	$\eta/\eta_{MAX}[\%]$	$M^*(k)$	[%]
Bernoulli	2/0.03	12	84	6.00	7.1
	10/0.0064	59	17	5.90	0.7
	20/0.0033	136	7	6.80	12.5
	30/0.0021	90	11	3.00	6.0
	40/0.0016	6	168	0.15	3.0
	80/0.0008	6	168	0.07	1.8

(a)

Poisson	2/4.45	9	112	4.50	10.6
	10/0.89	42	24	4.20	9.6
	20/0.44	84	12	4.20	11.3
	30/0.29	76	13	2.53	3.5
	40/0.22	52	19	1.30	1.7
	80/0.11	49	20	0.60	0.6

(b)

Pascal	2/20.53	3	337	1.50	7.9
	10/4.10	15	67	1.50	6.3
	20/2.05	28	36	1.40	3.3
	30/1.36	48	21	1.60	1.8
	40/1.02	66	15	1.65	0.7
	80/0.51	132	7	1.65	2.5

(c)

of communication impairments. The adopted FL algorithm is FedAvg [9] while Adam [13] is used as local optimizer with default hyper-parameters. The ML model is described in Fig. 1 and consists of 10^5 parameters with size $G = 2.53$ MB. Loss and performance metrics are the categorical cross-entropy and categorical accuracy, respectively. For each considered case, the validation accuracy is evaluated after $T_{FL} = 800$ seconds. Considering the communication with the PS, the bandwidth is $B = 40$ MHz while the max. efficiency dedicated to FL is set to $\eta_{MAX} = 0.05$ bit/sec/Hz. MQTT publishing uses Quality of Service (QoS) level 2 as this permits re-transmissions in exchange for larger $T_{\text{up}}(k)$ and $T_{\text{down}}(k)$.

As previously described, the FL process starts with a client local training to find $M_U(k)$ and subsequently $M^*(k)$ and $T^*_{PS}(k)$ according to Algorithm 1, with $\gamma = 0.5$. $C = 0.2$ adapts the VMR $D(k)$ contribution in (9) to the considered traffic types. Fig. 3 shows the validation loss/accuracy versus the local epochs that are used to retrieve $M_U(k)$ and $M^*(k)$ from (7) and (9). For clients with training data size S_k, the optimal $M^*(k)$ is 6 epochs. Setting now the training size to 10%, we observe shifts in the overfitting region (around $M_U(k)$) according to the bias-variance of the model: now $M^*(k) = 4$.

In Table I we consider at first homogeneous clients, i.e., clients with the same computing power and traffic distribution, but with non-IID data (80% of the local samples are drawn from the same class, chosen randomly). Traffic parameters are set to vary within the set $A(k) \in [2, 80]$ s and $\sigma^2(k) \in [0.001, 100]$. To evaluate the proposed policy, we choose $T_{PS} \in [1, 400]$ s and obtain the empirical optimal T_{PS} for each type of traffic. Notice that for the proposed example, setting $\eta_{MAX} = 0.05$, a client with $A(k) \leq 80$ s would require $M_L(k) = 1$. Table I summarizes the results for all experiments

TABLE II

CLIENTS ORGANIZED IN TWO CLUSTERS (C_1 AND C_2) AND NON-IID DATA (SEE TABLE I). THE RATIO η/η_{MAX} (%) REFERS TO THE WORST CLUSTER CASE, I.E., LOWER $M^*(k)$

		A_i/D_i	T^*_{PS}	$\eta/\eta_{MAX}[\%]$	$M^*(k)$	[%]
Bernoulli	C_1	8/0.0084	55	18	6.87	7.4
	C_2	10/0.0050			5.50	
	C_1	4/0.014	25	40	6.25	15.9
	C_2	20/0.0022			1.25	
	C_1	2/0.033	14	72	7.00	6.0
	C_2	40/0.0016			0.35	
	C_1	1/0.061	6	168	6.00	27.1
	C_2	80/0.0008			0.07	

(a)

Poisson	C_1	8/1.13	37	27	4.62	14.4
	C_2	10/0.91			3.70	
	C_1	4/2.15	20	50	5.00	17.5
	C_2	20/0.41			1.00	
	C_1	2/4.22	7	144	3.50	3.3
	C_2	40/0.19			0.17	
	C_1	1/8.83	4	253	4.00	26.4
	C_2	80/0.11			0.05	

(b)

Pascal	C_1	8/5.11	17	59	2.12	7.7
	C_2	10/4.23			1.70	
	C_1	4/11.12	9	112	2.25	3.8
	C_2	20/2.15			0.45	
	C_1	2/22.07	4	253	2.00	21.2
	C_2	40/1.18			0.10	
	C_1	1/43.92	3	337	3.00	22.9
	C_2	80/0.54			0.03	

(c)

grouped based on the traffic type (Bernoulli, Poisson, Pascal), mainly determined by the value of $\sigma^2(k)$. For each experiment, we highlight the traffic parameters $A(k)$, $D(k)$ (obtained for $M(k) = 1$), the resulting optimal T^*_{PS}, the corresponding $M^*(k)$ and the fraction (%) of utilized bandwidth w.r.t η_{MAX}. Last column quantifies the accuracy improvement w.r.t. a vanilla asynchronous strategy where the PS updates the global model as soon as a local model is available.

Considering the Bernoulli distribution (Table I.a) and $D(k) \approx 0$, the optimal values of T_{PS} and $M(k)$, obtained empirically, are in-line with the model (9), that gives $M^*(k) = 6$ for $\gamma = 0.5$ and $C = 0.2$. On the other hand, when $A(k) > 20$, namely the clients being all very slow, it is more convenient to keep the T_{PS} as low as possible, rather than following the policy (waiting the end of the optimized round). Increasing $D(k)$, namely using Poisson and Pascal traffic, this behaviour is less evident as the optimal $M^*(k)$ (4 and 1, respectively) is now in line with the policy and maintained for all $A(k)$. To summarize, the policy is effective for the prediction of the optimal values of $M^*(k)$ and can be used to tune the T^*_{PS} when $A(k) \ll T_{FL}$ (see cases highlighted in green).

Table II analyzes a more general case where the clients belong to clusters C_1 and C_2 and have different local learning completion times. Cluster C_2 contains much slower clients compared with C_1: the example is thus useful to verify the proposed policy for T_{PS} and whether the PS should specifically follow any cluster C_i, or not. To achieve that, we vary $A(k) = A_{i=1,2}$ of both clusters within the set $[1, 80]$ s, $T_{PS} \in [1, 400]$ s and $\sigma^2_{i=1,2} \in [0.001, 100]$ s^2. Cluster C_1 contains 5 clients while the remaining 4 clients belong to C_2. The results highlighted in blue show that the optimized T_{PS} should be set to follow the optimal number of local rounds $M^*(k)$ of the faster clients. For example, this can be seen in the extreme case of $(A_1 = 1, A_2 = 80)$ where the cluster C_2 almost does not affect the choice of T^*_{PS}. For all the considered cases, the use of an underestimated value of $M^*(k)$ should be preferred to prevent overfitted local models. In other words, it is more beneficial to update more often the global model following the faster clients, $k \in C_1$, as opposed to wait for the slower clients, $k \in C_2$, to complete their round. To conclude, we observe that designing T_{PS} based on the knowledge of the client-specific traffic is able to outperform asynchronous FL strategies. Optimization of T_{PS} is particularly critical for the case of two clusters. Observed accuracy gain increases from 5.1% (single cluster), to 15-17% on average.

V. CONCLUSION

The letter proposed a stochastic traffic model to describe the clients' behavior in FL processes. The model is based on the moment-matching approximation and it is verified with practical resource-constrained devices communicating with a Parameter Server (PS) using MQTT transport. Traffic characterization is used to develop a policy for the selection of the PS response time T_{PS} in asynchronous FL. The policy satisfies spectral efficiency constraints and avoids FL overfitting impairments. Results obtained in real setups show that an accuracy increase of up to 15-17% is possible when clients exhibit different local model completion times.

REFERENCES

[1] P. Kairouz et al., "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.
[2] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.
[3] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.
[4] B. C. Tedeschini et al., "Decentralized federated learning for healthcare networks: A case study on tumor segmentation," *IEEE Access*, vol. 10, pp. 8693–8708, 2022.
[5] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-IID data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 15–24.
[6] Z. Wang et al., "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6961–6978, Sep. 2022.
[7] (2021). *MQTT V3.1 Protocol Specification*. [Online]. Available: https://mqtt.org
[8] R. I. Wilkinson, "Theories for toll traffic engineering in the USA," *Bell Syst. Tech. J.*, vol. 35, no. 2, pp. 421–514, Mar. 1956.
[9] H. B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, Apr. 2017, pp. 1273–1282.
[10] K. Bonawitz et al., "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, vol. 1, 2019, pp. 374–388. [Online]. Available: https://tinyurl.com/34bwmd5m
[11] *Jetson Nano Developer Kit*. Accessed: Mar. 1, 2021. [Online]. Available: https://tinyurl.com/52mdbjzh
[12] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.
[13] D. P. Kingma et al., "Adam: A method for stochastic optimization," May 2015, *arXiv:1412.6980*.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement