A. Appendix

A.1. Experiment Settings

A.1.1 Computing Infrastructure

The simulation experiments are conducted on a computing server with one GPU. The server is equipped with AMD EPYC 7502 32-Core Processor and 1024G memory. The GPU is NVIDIA RTX A4000.

A.1.2 Datasets and Models

AI Benchmark. AI Benchmark [10] is a public dataset that is designed for evaluating the performance of important AI tasks on mobile devices. AI Benchmark provides diverse models' training and inference speed across various devices, including chipsets from Qualcomm, HiSilicon, Samsung, MediaTek, and Unisoc. Figure 8a illustrates the distribution of the computation efficiency across clients in the AI Benchmark. The slowest device would take around 13.3× computational times than the fastest device for the same task. To approach the dynamic availability of devices, such as low-power mode or multi-process running, we design a coefficient w as follows:

$$x \sim \mathcal{N}(1,\, 0.3)$$
$$w = \begin{cases} 1 & x \leq 1 \\ x & 1 \leq x \leq 1.3 \\ 1.3 & x \geq 1.3 \end{cases} \tag{2}$$

In this work, we assign the values from AI Benchmark as base computation time to the clients to emulate real devices, analogous to the usage in FedScale [14]. We also generate the coefficient w every round for each client to simulate the natural disturbance to availability. The local computation time in each round equals the product of w and the base computation time for each client.

MobiPerf. MobiPerf is a public dataset for measuring network performance on mobile devices, which collects the available cloud-to-edge network throughput of over 100k worldwide mobile clients. Figure 8b illustrates the distribution of communication consumption of MobiPerf. Note that the best communication channel can be 200× better than the worst one. We randomly assign a value from MobiPerf to a simulated device every communication round to emulate intermittent connectivity in a real deployment.

CIFAR-10. The CIFAR-10 dataset [13] consists of 60,000 32x32 colour images in 10 classes. There are 50,000 training images and 10,000 test images. We normalize the images by the mean and standard deviation of the dataset. We evaluate the dataset with ResNet-20 [7] model. To emulate the realistic non-iid distribution, we partition the dataset using a Dirichlet distribution, following the previous works [22].

Google Command. The Google Command speech dataset [29] covers 105,829 audio recordings collected from 2,618 clients. The training set includes recordings from 2,112D speakers, the validation set includes 256 speakers, and the test set includes 250 speakers. The data set is composed of 35 common words from the everyday vocabulary, such as "Yes", "No", "Up", and "Down". We evaluate the dataset with VGG11 [26] model and a lightweight model based on one related work [33] for a 35-class keyword spotting task.

For the VGG11-based experiment on Google Speech Commands, we use the Mel-frequency cepstral coefficients (MFCC) method to pre-process the raw audio data. Specifically, a sequence of overlapping Hamming windows is applied to the raw speech signal with a time shift of 10 ms and window size of 25ms. The MFCC is used for training the keyword spotting model.

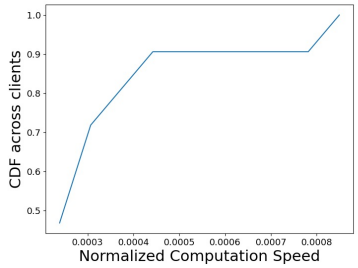
For the lightweight model experiment, to pre-process the raw audio data, a sequence of overlapping Hamming windows is applied to the raw speech signal with a time shift of 10 ms. We calculate the discrete Fourier transform (DFT) with a frame length of 1,024 and compute the Mel-spectrogram with a dimension of 128. The Mel-spectrogram is used for training the keyword spotting model. We follow [33] for this setup.

Reddit. Reddit [1] consists of comments from 1,660,820 users in the Reddit forum. Each client corresponds to a user, whose data are all of their personal posts. Thus it follows the real non-iid data under FL scenarios. In this dataset, we filter the users with less than 20 words in total and restrict to the 30k most frequently used words, as the same settings in the previous work [14]. Then, we train the lightweight Albert [16] model for the next-word-prediction task. The performance is evaluated by the perplexity loss (ppl), which lower is better.

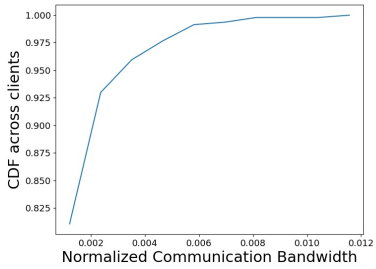
A.1.3 Hyperparameter Settings

We searched for the client learning rate in a range from 10^{-6} to 10^{0}, server learning rate in a range from 10^{-4} to 10^{0}, input batch size in a range from 8 to 256, and total training round in a range from 1000 to 10000. The aggregation goal and aggregation participation target is searched from 30% to 50% of training concurrency per round for `FedBuff` and `TimelyFL`, respectively.

After hyper-parameter searching, we fixed the following hyperparameters: for CIFAR-10 related experiments, the total training round is 2000, and training concurrency is 128 for all setups. The aggregation goal and aggregation participation target is 50% of the training concurrency for both `FedBuff` and `TimelyFL`. For CIFAR-10 with FedAvg related experiments, the batch size is 8, and the client learning rate is 0.8. For CIFAR-10 with FedOpt related ex-



(a) Diverse computation efficiency in AI Benchmark



(b) Diverse communicate efficiency in Mobiperf

Figure 8. Heterogeneous system utility across simulated clients.

periments, the batch size is 10, the client learning rate is 0.03, and the server learning rate is 0.001 with ADAM as server optimizer.

For Google command related experiments with VGG11 model, the total training round is 1000, and training concurrency is 20 for all setups. The aggregation goal and aggregation participation target is 50% of the training concurrency for both `FedBuff` and `TimelyFL`. The batch size is 32, and the client learning rate is 0.01. Under the FedOpt, the server learning rate is 0.001 with ADAM as server optimizer.

For Google command related experiments with the lightweight model, the total training round is 5000, and training concurrency is 106 for all setups. The aggregation goal and aggregation participation target is 50% of the training concurrency for both `FedBuff` and `TimelyFL`. The batch size is 16, and the client learning rate is 0.1 under the FedAvg. Under the FedOpt, the client learning rate is 0.05 for synchrounous FL and `TimelyFL`, and the client learning rate is 0.2 for `FedBuff`. The server learning rate is 0.001 with ADAM as server optimizer for all setups.

Finally, for Reddit related experiments, the total training round is 500, and training concurrency is 20 for all setups. The aggregation goal and aggregation participation target is 50% of the training concurrency for both `FedBuff` and `TimelyFL`. The batch size is 20, and the client learning rate is 0.0005 for SyncFL and `TimelyFL`, and 0.0003 for `FedBuff`. Under the FedOpt, the server learning rate is 0.001 with ADAM as server optimizer.

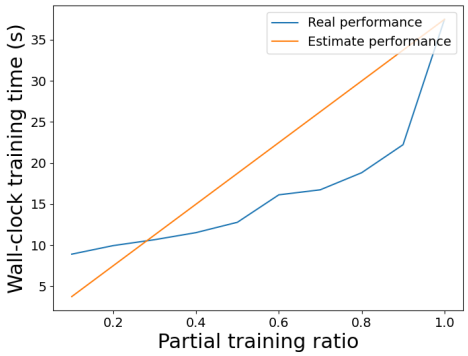


Figure 9. Partial training system performance in on real edge devices.

A.2. System Performance

A.2.1 Partial Training Performance

Due to different parameters and tensor shapes among different layers, the training time (computational time of the forward and backward propagation) is not strictly linear to the trainable layer numbers and varies with the model structures. For simplicity and generality, we define the training time of the partial model as the linear multiplication of the training time of the full model and the training ratio. This linear relationship is verified through our real measurement on a Samsung Galaxy S20 with ResNet-20 model using MNN [11] library. As shown in Figure 9, most of the test results are below the linear straight line (except the ratio is below 0.2), justifying the rationality of our choice.